User Guide

Amazon Virtual Private Cloud

Amazon Virtual Private Cloud: User Guide

Table of Contents

What is Amazon VPC? ... 1
- Features ... 1
- Getting started with Amazon VPC ... 3
- Working with Amazon VPC ... 3
- Pricing for Amazon VPC ... 3

How Amazon VPC works ... 6
- VPCs and subnets ... 7
- Default and nondefault VPCs ... 7
- Route tables ... 8
- Access the internet ... 8
- Access a corporate or home network ... 9
- Connect VPCs and networks ... 9
- AWS private global network ... 10

Plan your VPC ... 11
- Sign up for an AWS account ... 11
- Verify permissions ... 12
- Determine your IP address ranges ... 12
- Select your Availability Zones ... 12
- Plan your internet connectivity ... 13
- Create your VPC ... 13
- Deploy your application ... 13

IP addressing ... 15
- Private IPv4 addresses ... 16
- Public IPv4 addresses ... 16
- IPv6 addresses ... 18
  - Public IPv6 addresses ... 19
  - Private IPv6 addresses ... 19
- Use your own IP addresses ... 21
- Use Amazon VPC IP Address Manager ... 21
- VPC CIDR blocks ... 21
  - IPv4 VPC CIDR blocks ... 22
  - Manage IPv4 CIDR blocks for a VPC ... 23
  - IPv4 CIDR block association restrictions ... 26
  - IPv6 VPC CIDR blocks ... 28

Subnet CIDR blocks 29
Subnet sizing for IPv4 29
Subnet sizing for IPv6 30
Compare IPv4 and IPv6 31
Managed prefix lists 32
Prefix lists concepts and rules 33
Identity and access management for prefix lists 34
Customer-managed prefix lists 35
AWS-managed prefix lists 45
Optimize AWS infrastructure management with prefix lists 47
AWS IP address ranges 49
Download 50
Egress control 51
Geolocation feed 51
Find address ranges 51
Syntax 58
Subscribe to notifications 63
IPv6 support for your VPC 65
Add IPv6 support for your VPC 65
Example dual-stack VPC 70
IPv6 support on AWS 72
Services that support IPv6 72
Additional IPv6 support 81
Learn more 81
Virtual private clouds 82
VPC basics 83
VPC IP address range 83
VPC diagram 83
VPC resources 84
VPC configuration options 85
Default VPCs 86
Default VPC components 87
Default subnets 89
Work with your default VPC and default subnets 90
Create a VPC 94
Create a VPC plus other VPC resources 94

- Create a VPC only 96
- Create a VPC using the AWS CLI 98
- Visualize the resources in your VPC 102
- Add or remove CIDR block 104
- DHCP option sets 106
  - What is DHCP? 107
  - DHCP option set concepts 108
  - Work with DHCP option sets 111
- DNS attributes 116
  - Understanding Amazon DNS 116
  - View DNS hostnames for your EC2 instance 121
  - View and update DNS attributes for your VPC 122
- Network Address Usage 123
  - How NAU is calculated 124
  - NAU examples 125
- Share a VPC subnet 126
  - Shared subnet prerequisites 127
  - Working with shared subnets 127
  - Billing and metering for owner and participants 130
  - Responsibilities and permissions for owners and participants 131
  - AWS resources and shared VPC subnets 133
- Extend a VPC to other Zones 135
  - Subnets in AWS Local Zones 135
  - Subnets in AWS Wavelength 140
  - Subnets in AWS Outposts 143
- Delete your VPC 144
  - Delete using the console 145
  - Delete using the CLI 146
- Generate IaC from console actions 147

Subnets 149
- Subnet basics 149
  - Subnet IP address range 149
  - Subnet types 150
  - Subnet diagram 151
  - Subnet routing 151
  - Subnet settings 151

Subnet security ... 152
Create a subnet ... 152
Add or remove an IPv6 CIDR block from your subnet ... 154
Modify the IP addressing attributes of your subnet ... 155
Subnet CIDR reservations ... 156
Work with subnet CIDR reservations using the console ... 157
Work with subnet CIDR reservations using the AWS CLI ... 158
Route tables ... 159
Route table concepts ... 160
Subnet route tables ... 161
Gateway route tables ... 168
Route priority ... 171
Example routing options ... 173
Create a route table and routes ... 188
Manage subnet route tables ... 190
Replace the main route table ... 194
Associate a route table with a gateway ... 195
Replace or restore the target for a local route ... 195
Advanced routing ... 197
Troubleshoot reachability issues ... 243
Middlebox routing wizard ... 244
Middlebox routing wizard prerequisites ... 244
Redirect VPC traffic to a security appliance ... 244
Middlebox routing wizard considerations ... 247
Middlebox scenarios ... 248
Delete a subnet ... 258
Connect your VPC ... 260
Internet gateways ... 261
Internet gateway basics ... 262
Create an internet gateway ... 264
Delete an internet gateway ... 267
Egress-only internet gateways ... 268
Egress-only internet gateway basics ... 268
Add egress-only internet access to a subnet ... 270
NAT devices ... 272
NAT gateways ... 274

- NAT instances ... 321
- Compare NAT devices ... 333
- Elastic IP addresses ... 336
- Elastic IP address concepts and rules ... 337
- Start using Elastic IP addresses ... 338
- AWS Transit Gateway ... 348
- AWS Virtual Private Network ... 349
- VPC peering connections ... 350

Monitoring ... 352

- VPC Flow Logs ... 353
- Flow logs basics ... 354
- Flow log records ... 357
- Flow log record examples ... 368
- Flow log limitations ... 377
- Pricing ... 379
- Work with flow logs ... 379
- Publish to CloudWatch Logs ... 383
- Publish to Amazon S3 ... 391
- Publish to Amazon Data Firehose ... 399
- Query using Athena ... 407
- Troubleshoot ... 411
- CloudWatch metrics ... 415
- NAU metrics and dimensions ... 416
- Enable or disable NAU monitoring ... 419
- NAU CloudWatch alarm example ... 420
- Billing and usage reports ... 420
- IP address management ... 421
- VPC endpoints ... 422
- Transit gateways ... 423
- Network analysis ... 423
- Traffic mirroring ... 424
- VPC Lattice ... 424
- Cross-account/Region resources ... 425

Security ... 426

- Data protection ... 427
- Internetwork traffic privacy ... 428

Identity and access management ... 428
Audience ... 429
Authenticate with identities ... 429
Manage access using policies ... 432
How Amazon VPC works with IAM ... 435
Policy examples ... 439
Troubleshoot ... 452
AWS managed policies ... 454
Infrastructure security ... 457
Network isolation ... 457
Control network traffic ... 458
Compare security groups and network ACLs ... 459
Security groups ... 460
Security group basics ... 462
Security group example ... 463
Security group rules ... 464
Default security groups ... 469
Create a security group ... 471
Configure security group rules ... 472
Delete a security group ... 474
Associate security groups with multiple VPCs ... 475
Share security groups with AWS Organizations ... 478
Network ACLs ... 484
Network ACL basics ... 485
Network ACL rules ... 487
Default network ACL ... 488
Custom network ACLs ... 490
Path MTU Discovery ... 495
Create a network ACL ... 496
Manage network ACL associations ... 499
Delete a network ACL ... 502
Example: Control access to instances in a subnet ... 503
Resilience ... 506
Compliance validation ... 508
Block public access to VPCs and subnets ... 509
VPC BPA basics ... 510

Assess impact of VPC BPA and monitor VPC BPA ... 516
Advanced example ... 520
Best practices ... 574
Use with other services ... 575
AWS PrivateLink ... 576
AWS Network Firewall ... 577
Route 53 Resolver DNS Firewall ... 579
Reachability Analyzer ... 580
Examples ... 581
Test environment ... 582
Overview ... 582
1. Create the VPC ... 584
2. Deploy your application ... 585
3. Test your configuration ... 586
4. Clean up ... 586
Web and database servers ... 586
Overview ... 586
1. Create the VPC ... 590
2. Deploy your application ... 592
3. Test your configuration ... 592
4. Clean up ... 592
Private servers ... 593
Overview ... 593
1. Create the VPC ... 596
2. Deploy your application ... 597
3. Test your configuration ... 597
4. Clean up ... 597
Tutorials ... 598
Getting started with Amazon VPC using the AWS CLI ... 598
... 598
Prerequisites ... 599
Create a VPC ... 599
Create subnets ... 600
Configure internet connectivity ... 602
Create a NAT Gateway ... 603
Configure subnet settings ... 604

- Create security groups ... 605
- Verify your VPC configuration ... 606
- Deploy EC2 instances ... 607
- Troubleshooting ... 211
- Clean up resources ... 610
- Going to production ... 611
- Next steps ... 612
- Create a VPC with private subnets and NAT gateways using AWS CLI ... 612
  - Prerequisites ... 599
  - Create the VPC and subnets ... 614
  - Create and configure internet connectivity ... 616
  - Create NAT gateways ... 618
  - Create a VPC endpoint for Amazon S3 ... 619
  - Configure security groups ... 620
  - Create a launch template for EC2 instances ... 621
  - Create a load balancer and target group ... 622
  - Create an Auto Scaling group ... 624
  - Test your configuration ... 624
  - Clean up resources ... 610
  - Next steps ... 612

Quotas ... 628

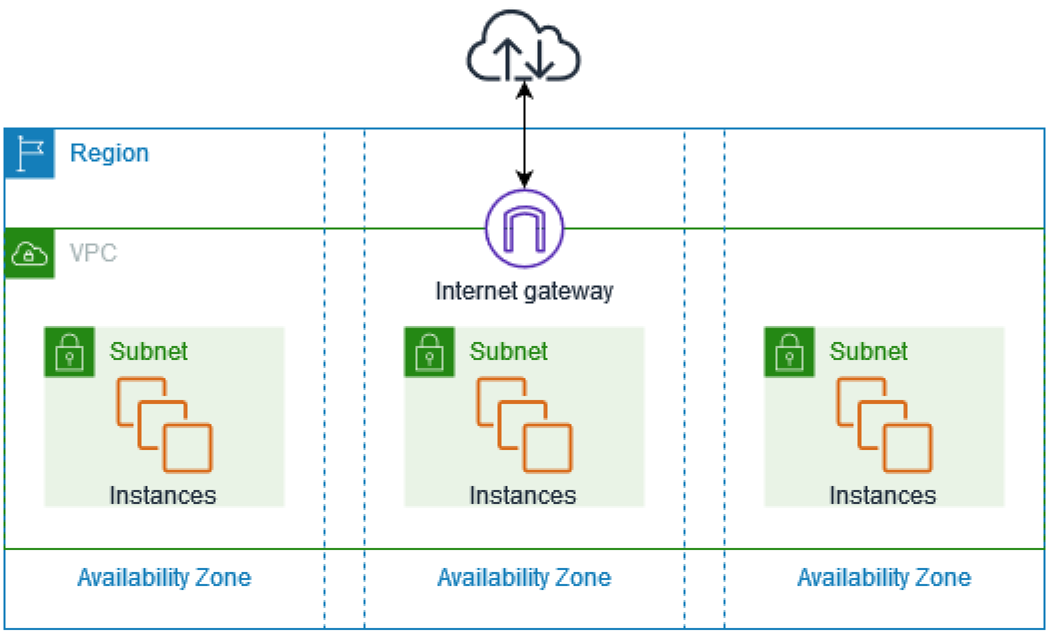
- VPC and subnets ... 628
- DNS ... 629
- Elastic IP addresses ... 629
- Gateways ... 629
- Customer-managed prefix lists ... 630
- Network ACLs ... 631
- Network interfaces ... 632
- Route tables ... 632
- Route servers ... 633
- Security groups ... 635
- VPC subnet sharing ... 636
- Network Address Usage ... 637
- Amazon EC2 API throttling ... 637
- Additional quota resources ... 637

Document history ... 639

What is Amazon VPC?

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

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



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

Features

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

Virtual private clouds (VPC)

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

Subnets

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

IP addressing

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

Routing

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

Gateways and endpoints

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

Peering connections

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

Traffic Mirroring

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

Transit gateways

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

VPC Flow Logs

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

VPN connections

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

Getting started with Amazon VPC

Your AWS account includes a default VPC in each AWS Region. Your default VPCs are configured such that you can immediately start launching and connecting to EC2 instances. For more information, see *Plan your VPC*.

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

Working with Amazon VPC

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

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

Pricing for Amazon VPC

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

Nearly all resources that you launch in your virtual private cloud (VPC) provide you with an IP address for connectivity. The vast majority of resources in your VPC use private IPv4 addresses. Resources that require direct access to the internet over IPv4, however, use public IPv4 addresses.

Amazon VPC enables you to launch managed services, such as Elastic Load Balancing, Amazon RDS, and Amazon EMR, without having a VPC set up beforehand. It does this by using the default

VPC in your account if you have one. Any public IPv4 addresses provisioned to your account by the managed service will be charged. These charges will be associated with Amazon VPC service in your AWS Cost and Usage Report.

Pricing for public IPv4 addresses

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

If you are an existing or new AWS Free Tier customer, you get 750 hours of public IPv4 address usage with the EC2 service at no charge. If you are not using the EC2 service in the AWS Free Tier, Public IPv4 addresses are charged. For specific pricing information, see the *Public IPv4 address* tab in Amazon VPC Pricing.

Private IPv4 addresses (RFC 1918) are not charged. For more information about how public IPv4 addresses are charged for shared VPCs, see Billing and metering for the owner and participants.

Public IPv4 addresses have the following types:

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

The following list shows the most common AWS services that can use public IPv4 addresses.

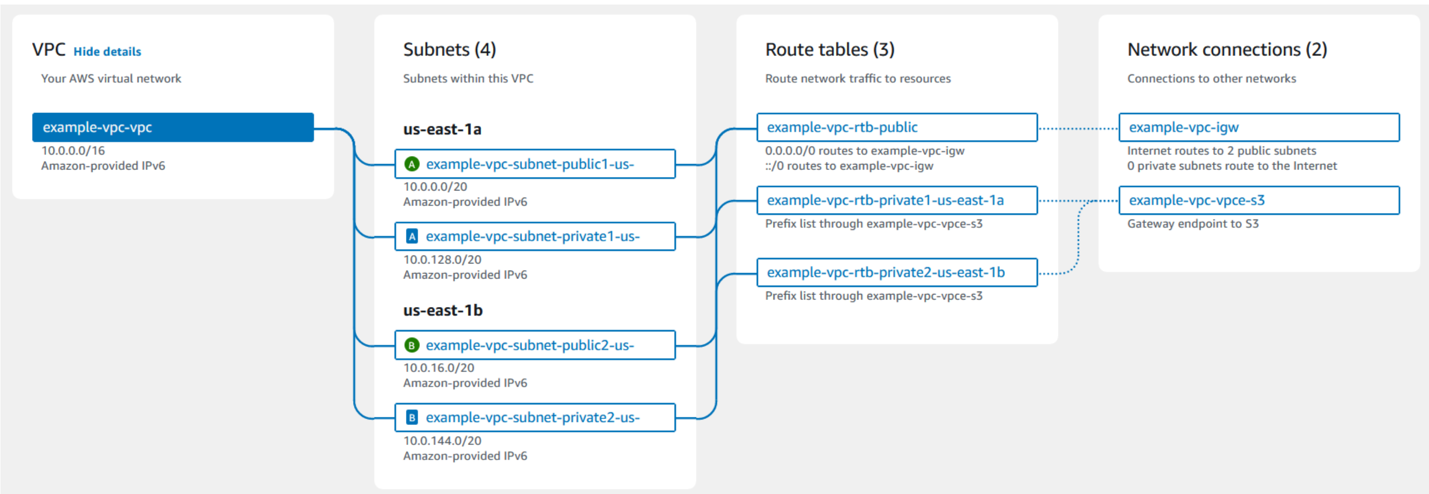
- Amazon AppStream 2.0
- AWS Client VPN
- AWS Database Migration Service
- Amazon EC2
- Amazon Elastic Container Service

- Amazon EKS
- Amazon EMR
- Amazon GameLift Servers
- AWS Global Accelerator
- AWS Mainframe Modernization
- Amazon Managed Streaming for Apache Kafka
- Amazon MQ
- Amazon RDS
- Amazon Redshift
- AWS Site-to-Site VPN
- Amazon VPC NAT gateway
- Amazon WorkSpaces
- Elastic Load Balancing

How Amazon VPC works

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

The following is a visual representation of a VPC and its resources from the **Preview** pane shown when you create a VPC using the AWS Management Console. For an existing VPC, you can access this visualization on the **Resource map** tab. This example shows the resources that are initially selected on the **Create VPC** page when you choose to create the VPC plus other networking resources. This VPC is configured with an IPv4 CIDR and an Amazon-provided IPv6 CIDR, subnets in two Availability Zones, three route tables, an internet gateway, and a gateway endpoint. Because we've selected the internet gateway, the visualization indicates that traffic from the public subnets is routed to the internet because the corresponding route table sends the traffic to the internet gateway.



Concepts

- VPCs and subnets
- Default and nondefault VPCs
- Route tables
- Access the internet
- Access a corporate or home network
- Connect VPCs and networks

- AWS private global network

VPCs and subnets

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

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

Learn more

- *IP addressing*
- *Virtual private clouds*
- *Subnets*

Default and nondefault VPCs

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

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

Learn more

- the section called “Default VPCs”

- the section called “Create a VPC”

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

- Configure route tables

Access the internet

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

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

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

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

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

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

Learn more

- Enable internet access for a VPC using an internet gateway
- Enable outbound IPv6 traffic using an egress-only internet gateway
- Connect to the internet or other networks using NAT devices

Access a corporate or home network

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

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

Learn more

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

Connect VPCs and networks

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

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

Learn more

- Amazon VPC Peering Guide
- Amazon VPC Transit Gateways

AWS private global network

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

Packets that originate in the private global network with a destination in the private global network stay in the private global network and do not traverse the public internet. This is true whether the destination is a private IP address or a public IP address. For example, if EC2 instances in two VPCs communicate using public IP addresses, the traffic stays in the private global network. The destination can be in the same Availability Zone, a different Availability Zone in the same Region, or a different Region, except for the 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%.

Plan your VPC

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

Tasks

- Sign up for an AWS account
- Verify permissions
- Determine your IP address ranges
- Select your Availability Zones
- Plan your internet connectivity
- Create your VPC
- Deploy your application

Sign up for an AWS account

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

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

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

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

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

Verify permissions

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

Determine your IP address ranges

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

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

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

Select your Availability Zones

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

Production environment

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

Development or test environment

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

Plan your internet connectivity

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

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

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

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

Create your VPC

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

Deploy your application

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

Production environment

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

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

Development or test environment

For a development or test environment, you might choose to launch a single EC2 instance. For more information, see Get started with Amazon EC2 in the *Amazon EC2 User Guide*.

IP addressing for your VPCs and subnets

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

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

- An individual IPv4 address is 32 bits, with 4 groups of up to 3 decimal digits, 0-255. For example, 10.0.1.0.
- An IPv4 CIDR block has an IPv4 address followed by a slash and a number from 0 to 32. For example, 10.0.0.0/16 represents 65,536 IPv4 addresses from 10.0.0.0 to 10.0.255.255.
- An individual IPv6 address is 128 bits, with 8 segments of 4 hexadecimal digits. For example, 2001:0db8:85a3:0000:0000:8a2e:0370:7334. It is not necessary to include the leading zeros in a segment. You can also replace consecutive all-zero segments with double colons (::) one time in an address. Therefore, the example address can be compressed as 2001:db8:85a3::8a2e:370:7334.
- An IPv6 CIDR block has an IPv6 address that ends with all-zero segments, with the all-zero segments replaced by a double colon, followed by a slash and a number from 0 to 128. For example, 2001:db8:1234:1a00::/56 represents 2^72 IPv6 addresses from 2001:db8:1234:1a00:0000:0000:0000:0000 to 2001:db8:1234:1aff:ffff:ffff:ffff:ffff.

For more information, see What is CIDR?

Contents

- Private IPv4 addresses
- Public IPv4 addresses
- IPv6 addresses
- Use your own IP addresses
- Use Amazon VPC IP Address Manager
- VPC CIDR blocks
- Subnet CIDR blocks
- Compare IPv4 and IPv6
- Consolidate and manage network CIDR blocks with managed prefix lists

- AWS IP address ranges
- IPv6 support for your VPC
- AWS services that support IPv6

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 primary network interface (for example, eth0) of the instance. Each instance is also given a private (internal) DNS hostname that resolves to the private IP address of the instance. The hostname can be of two types: resource-based or IP-based. For more information, see EC2 instance naming. If you don't specify a primary private IP address, we select an available IP address in the subnet range for you. For more information about network interfaces, see Elastic Network Interfaces in the *Amazon EC2 User Guide*.

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*.

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.

We never advertise the IPv4 address range of a subnet to the internet.

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 that's created for the instance. A public IP address is mapped to the primary private IP address through network address translation (NAT).

> **Note**
>
> AWS charges for all public IPv4 addresses, including public IPv4 addresses associated with running instances and Elastic IP addresses. For more information, see the **Public IPv4 Address** tab on the Amazon VPC pricing page.

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 IP addressing attributes of your subnet.
- Enabling or disabling the public IP addressing feature during instance launch, which overrides the subnet's public IP addressing attribute.
- You can unassign a public IP address from your instance after launch by managing the IP addresses associated with a network interface. For more information, see Manage IP addresses in the *Amazon EC2 User Guide*.

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. 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*.

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

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

If you are using Amazon VPC IP Address Manager (IPAM), you can get a contiguous block of public IPv4 addresses from AWS and use it to allocate sequential Elastic IP addresses to AWS resources.

Using contiguous IPv4 address blocks can significantly reduce management overhead for security access control lists and simplify IP address allocation and tracking for enterprises scaling on AWS. For more information, see Allocate sequential Elastic IP addresses from an IPAM pool in the *Amazon VPC IPAM User Guide*.

IPv6 addresses

As the internet continues to grow, so does the need for IP addresses. The most common format for IP addresses is IPv4. The new format for IP addresses is IPv6, which provides a larger address space than IPv4. IPv6 resolves the IPv4 address exhaustion issue and enables you to connect more devices to the internet. The transition is gradual, but as IPv6 adoption grows, you can simplify your networks and take advantage of IPv6 advanced capabilities for better connectivity, performance, and security.

Many AWS services, such as Amazon EC2, Amazon S3, and Amazon CloudFront, offer either dual-stack (IPv4 and IPv6) or IPv6-only support, allowing resources to be assigned IPv6 addresses and accessed over the IPv6 protocol and simplifying network configuration and management for those customers adopting IPv6. Other services offer limited or partial dual-stack and IPv6-only support.

For more information about services that support IPv6, see AWS services that support IPv6.

Note that some IPv6 addresses are reserved by the Internet Engineering Task Force. For more information about reserved IPv6 address ranges, see IANA IPv6 Special-Purpose Address Registry and RFC4291.

Note

Both public and private IPv6 addressing is available in AWS. AWS defines public IP addresses as those advertised on the internet from AWS, while private IP addresses are not and cannot be advertised on the internet from AWS.

Contents

- Public IPv6 addresses
- Private IPv6 addresses

Public IPv6 addresses

Amazon-provided IPv6 addresses are always advertised on the internet. They are globally unique, and therefore reachable over the internet. You can control whether resources such as EC2 instances are reachable using their IPv6 addresses by controlling routing for your subnets, or by using security groups and network ACLs.

These are some of the ways you can prepare to use public IPv6 addresses for your workloads:

- Create an IPAM with Amazon VPC IP Address Manager and provision an Amazon-owned public IPv6 address range to an IPAM address pool. For more information, see Create IPv6 pools in the *Amazon VPC IPAM User Guide*.
- If you have an IPAM and you own a public IPv6 address range, bring some or all of the public IPv6 address range to IPAM and provision the public IPv6 address range to an IPAM address pool. For more information, see Tutorial: Bring your IP addresses to IPAM in the *Amazon VPC IPAM User Guide*.
- If you don't have an IPAM but you own a public IPv6 address range, bring some or all of the public IPv6 address range to AWS. For more information, see Bring your own IP addresses (BYOIP) to Amazon EC2 in the *Amazon EC2 User Guide*.

When you are prepared to use public IPv6 addresses, you can assign public IPv6 addresses to instances (see IPv6 addresses in the *Amazon EC2 User Guide*), you can allocate a public IPv6 CIDR block to your VPC (see Add or remove a CIDR block from your VPC) and associate the IPv6 CIDR block with your subnets (see Modify the IP addressing attributes of your subnet).

Private IPv6 addresses

Private IPv6 addresses are IPv6 addresses that are not advertised and cannot be advertised on the Internet from AWS.

You can use a private IPv6 address if you want your private networks to support IPv6 and you have no intention of routing traffic from these addresses to the Internet. If you want to connect to the internet from a resource that has a private IPv6 address, you can, but you must route traffic through a resource in another subnet with a public IPv6 address to do so.

There are two types of private IPv6 addresses:

- **IPv6 ULA ranges**: IPv6 addresses as defined in RFC4193. These address ranges always start with “fc” or “fd”, which makes them easily identifiable. Valid IPv6 ULA space is anything under fd00::/8 that does not overlap with the Amazon reserved range fd00::/16.
- **IPv6 GUA ranges**: IPv6 addresses as defined in RFC3587. The option to use IPv6 GUA ranges as private IPv6 addresses is disabled by default and must be enabled before you can use it. For more information, see Enable provisioning private IPv6 GUA CIDRs in the *Amazon VPC IPAM User Guide*.

Note the following:

- Private IPv6 addresses are only available through Amazon VPC IP Address Manager (IPAM). IPAM discovers resources with IPv6 ULA and GUA addresses and monitors pools for overlapping IPv6 ULA and GUA address space.
- When you use private IPv6 GUA ranges, we require that you use IPv6 GUA ranges owned by you.
- Private IPv6 addresses are not and cannot be advertised on the internet by AWS. AWS does not allow direct egress to the public internet from a private IPv6 range even if there is an internet gateway or egress only internet gateway in the VPC. Private IPv6 addresses are automatically dropped at the internet gateway edge ensuring that they are not routed publicly.
- AWS reserves the first 4 subnet private IPv6 addresses and the last one.
- Valid ranges for private IPv6 ULA are /9 to /60 starting with fd80::/9.
- If you have a private IPv6 GUA range allocated to a VPC, you cannot use public IPv6 GUA space that overlaps the private IPv6 GUA space in the same VPC.
- Communication between resources with private IPv6 ULA and GUA address ranges is supported (such as across Direct Connect, VPC peering, transit gateway, or VPN connections).
- You can use private IPv6 addresses with IPv6-only and dual-stack VPC subnets, elastic load balancers and AWS Global Accelerator endpoints.
- There is no charge for private IPv6 addresses.

These are some of the ways you can prepare to use private IPv6 addresses for your workloads:

- Create an IPAM with Amazon VPC IP Address Manager and provision a private IPv6 *ULA* range to an IPAM address pool. For more information, see Create IPv6 pools in the *Amazon VPC IPAM User Guide*.

- Create an IPAM with Amazon VPC IP Address Manager and provision a private IPv6 *GUA* range to an IPAM address pool. The option to use IPv6 GUA ranges as private IPv6 addresses is disabled by default and must be enabled on your IPAM before you can use it. For more information, see Enable provisioning private IPv6 GUA CIDRs in the *Amazon VPC IPAM User Guide*.

When you are prepared to use private IPv6 addresses, you can allocate a private IPv6 CIDR block from an IPAM pool to your VPC (see Add or remove a CIDR block from your VPC) and associate the IPv6 CIDR block with your subnets (see Modify the IP addressing attributes of your subnet).

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*.

Use Amazon VPC IP Address Manager

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

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

VPC CIDR blocks

The IP addresses for your virtual private cloud (VPC) are represented using Classless Inter-Domain Routing (CIDR) notation. A VPC must have an associated IPv4 CIDR block. You can optionally associate additional IPv4 CIDR blocks and one or more IPv6 CIDR blocks. For more information, see IP addressing for your VPCs and subnets.

Contents

- IPv4 VPC CIDR blocks
- Manage IPv4 CIDR blocks for a VPC

- IPv4 CIDR block association restrictions
- IPv6 VPC CIDR blocks

IPv4 VPC CIDR blocks

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

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

RFC 1918 range	Example CIDR block
10.0.0.0 - 10.255.255.255 (10/8 prefix)	10.0.0.0/16
172.16.0.0 - 172.31.255.255 (172.16/12 prefix)	172.31.0.0/16
192.168.0.0 - 192.168.255.255 (192.168/16 prefix)	192.168.0.0/20

Considerations

- You can't specify the following CIDR blocks for your VPCs:
  - 0.0.0.0/8
  - 127.0.0.0/8 (internal host loopback address range)
  - 169.254.0.0/16 (link-local address range)
  - 224.0.0.0/4 (multicast address range)
- When you create a VPC for use with an AWS service, check the service documentation to verify if there are specific requirements for its configuration.
- Some AWS services use the `172.17.0.0/16` CIDR range. Services can experience IP address conflicts if the IP address range is already in use in your network. For example, AWS Cloud9 and Amazon SageMaker AI use `172.17.0.0/16`. To avoid conflicts, don't use this range when

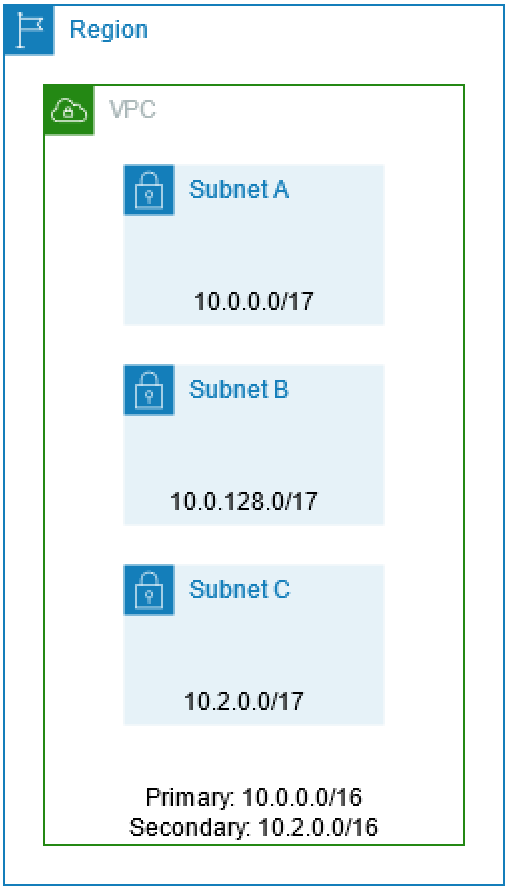
creating your VPC. For more information, see [Can't connect to EC2 environment because VPC's IP addresses are used by Docker] in the *AWS Cloud9 User Guide*.

- You can create a VPC with a publicly routable CIDR block that falls outside of the private IPv4 address ranges specified in RFC 1918. However, for the purposes of this documentation, we refer to *private IP addresses* as the IPv4 addresses that are within the CIDR range of your VPC.
- If you create a VPC using a command line tool or the Amazon EC2 API, the CIDR block is automatically modified to its canonical form. For example, if you specify 100.68.0.18/18 for the CIDR block, we create a CIDR block of 100.68.0.0/18.

Manage IPv4 CIDR blocks for a VPC

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

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



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

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](#).
- 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.
- The CIDR block must not be the same or larger than a destination CIDR range in a route in any of the VPC route tables. For example, in a VPC where the primary CIDR block is `10.2.0.0/16`, you have an existing route in a route table with a destination of `10.0.0.0/24` to a virtual private gateway. You want to associate a secondary CIDR block in the `10.0.0.0/16` range. Because of the existing route, you cannot associate a CIDR block of `10.0.0.0/24` or larger. However, you can associate a secondary CIDR block of `10.0.0.0/25` or smaller.
- The following rules apply when you add IPv4 CIDR blocks to a VPC that's part of a VPC peering connection:
  - If the VPC peering connection is `active`, you can add CIDR blocks to a VPC provided they do not overlap with a CIDR block of the peer VPC.
  - If the VPC peering connection is `pending-acceptance`, the owner of the requester VPC cannot add any CIDR block to the VPC, regardless of whether it overlaps with the CIDR block of the accepter VPC. Either the owner of the accepter VPC must accept the peering connection, or the owner of the requester VPC must delete the VPC peering connection request, add the CIDR block, and then request a new VPC peering connection.
  - If the VPC peering connection is `pending-acceptance`, the owner of the accepter VPC can add CIDR blocks to the VPC. If a secondary CIDR block overlaps with a CIDR block of the requester VPC, the VPC peering connection request fails and cannot be accepted.
- If you're using AWS Direct Connect to connect to multiple VPCs through a Direct Connect gateway, the VPCs that are associated with the Direct Connect gateway must not have overlapping CIDR blocks. If you add a CIDR block to one of the VPCs that's associated with the Direct Connect gateway, ensure that the new CIDR block does not overlap with an existing CIDR block of any other associated VPC. For more information, see Direct Connect gateways in the *AWS Direct Connect User Guide*.
- When you add or remove a CIDR block, it can go through various states: `associating` | `associated` | `disassociating` | `disassociated` | `failing` | `failed`. The CIDR block is ready for you to use when it's in the `associated` state.

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

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

The following is example output.

```
10.0.0.0/16
```

IPv4 CIDR block association restrictions

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

IP address range	Restricted associations	Permitted 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 any of the CIDR blocks associated with the VPC are from the 10.0.0.0/15 range (10.0.0.0 to 10.1.255.255), you cannot add a CIDR block from the 10.0.0.0/16 range (10.0.0.0 to 10.0.255.255). CIDR blocks from the 198.19.0.0/16 range.	Any other CIDR block from the 10.0.0.0/8 range between a /16 netmask and /28 netmask that's not restricted. Any publicly routable IPv4 CIDR block (non-RFC 1918) between a /16 netmask and /28 netmask or a CIDR block between a /16 netmask and /28 netmask from the 100.64.0.0/10 range.
169.254.0.0/16	CIDR blocks from the "link local" block are reserved as described in RFC 5735 and cannot be assigned to VPCs.	

IP address range	Restricted associations	Permitted associations
172.16.0.0/12	CIDR blocks from other RFC 1918* ranges (10.0.0.0/8 and 192.168.0 .0/16). CIDR blocks from the 172.31.0.0/16 range. CIDR blocks from the 198.19.0.0/16 range.	Any other CIDR block from the 172.16.0.0/12 range between a /16 netmask and /28 netmask that's not restricted. Any publicly routable IPv4 CIDR block (non-RFC 1918) between a /16 netmask and /28 netmask or a CIDR block between a /16 netmask and /28 netmask from the 100.64.0. 0/10 range.
192.168.0.0/16	CIDR blocks from other RFC 1918* ranges (10.0.0.0/8 and 172.16.0. 0/12). CIDR blocks from the 198.19.0.0/16 range.	Any other CIDR block from the 192.168.0.0/16 range between a /16 netmask and /28 netmask. Any publicly routable IPv4 CIDR block (non-RFC 1918) between a /16 netmask and /28 netmask or a CIDR block from the 100.64.0.0/10 range between a /16 netmask and /28 netmask.
198.19.0.0/16	CIDR blocks from the RFC 1918* ranges.	Any publicly routable IPv4 CIDR block (non-RFC 1918) between a /16 netmask and /28 netmask or a CIDR block from the 100.64.0.0/10 range between a /16 netmask and /28 netmask.

IP address range	Restricted associations	Permitted associations
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. CIDR blocks from the 198.19.0.0/16 range.	Any other publicly routable IPv4 CIDR block (non-RFC 1918) between a /16 netmask and /28 netmask or a CIDR block between a /16 netmask and /28 netmask from the 100.64.0.0/10 range. You can also associate a CIDR in one of the RFC 1918 ranges, but to do this you have to add that CIDR first when you create the VPC and then add the non-RFC 1918 CIDR.

* RFC 1918 ranges are the private IPv4 address ranges specified in RFC 1918.

IPv6 VPC CIDR blocks

You can associate a single IPv6 CIDR block when you create a new VPC or you can associate up to five IPv6 CIDR blocks from `/44` to `/60` in increments of `/4`. You can request an IPv6 CIDR block from Amazon's pool of IPv6 addresses. For more information, see Add or remove a CIDR block from your VPC.

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

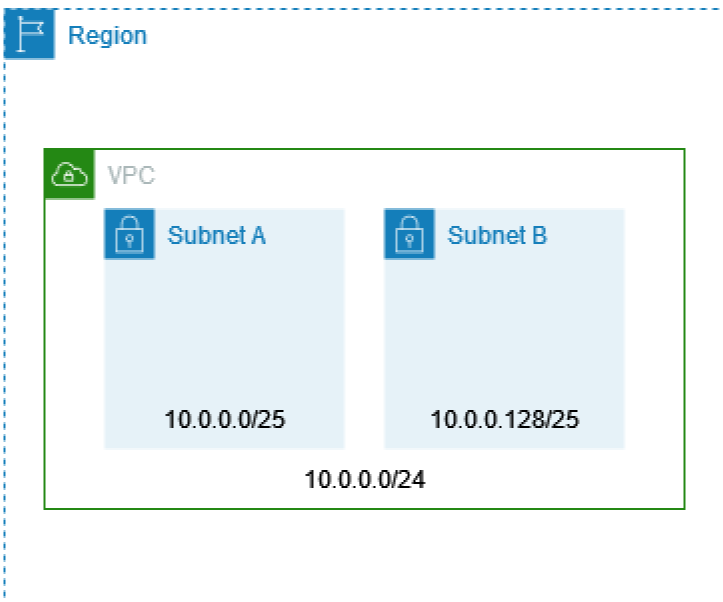
For example, you create a VPC and specify that you want to associate an Amazon-provided IPv6 CIDR block with the VPC. Amazon assigns the following IPv6 CIDR block to your VPC: `2001:db8:1234:1a00::/56`. You cannot choose the range of IP addresses yourself. You can create a subnet and associate an IPv6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

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

Subnet CIDR blocks

The IP addresses for your subnets are represented using Classless Inter-Domain Routing (CIDR) notation. The CIDR block of a subnet can be the same as the CIDR block for the VPC (to create a single subnet in the VPC), or a subset of the CIDR block for the VPC (to create multiple subnets in the VPC). 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 and IPv6 subnet CIDR blocks. You can find tools that suit your needs by searching for terms such as 'subnet calculator' or 'CIDR calculator'. Your network engineering group can also help you determine the IPv4 and IPv6 CIDR blocks to specify for your subnets.

Subnet sizing for IPv4

The allowed IPv4 CIDR block size for a subnet is between a /28 netmask and /16 netmask. The first four IP addresses and the last IP address in each subnet CIDR block are not available for your use, and they cannot be assigned to a resource, such as an EC2 instance. For example, in a subnet with CIDR block `10.0.0.0/24`, the following five IP addresses are reserved:

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

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

If you bring an IPv4 address range to AWS using BYOIP, you can use all of the IP addresses in the range, including the first address (the network address) and the last address (the broadcast address).

Subnet sizing for IPv6

If you've associated an IPv6 CIDR block with your VPC, you can associate an IPv6 CIDR block with an existing subnet in your VPC, or when you create a new subnet. Possible IPv6 netmask lengths are between `/44` and `/64` in increments of `/4`.

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

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

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

- `2001:db8:1234:1a00::3`
- `2001:db8:1234:1a00:ffff:ffff:ffff:ffff`

In addition to the IP address reserved by AWS for the VPC router in the example above, the following IPv6 addresses are reserved for the default VPC router:

- A link-local IPv6 address in the FE80::/10 range generated using EUI-64. For more information about link-local addresses, see Link-local address.
- The link-local IPv6 address `FE80:ec2::1`.

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

Compare IPv4 and IPv6

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

For a list of AWS services that support dual-stack configuration (IPv4 and IPv6) and IPv6-only configurations, see Services that support IPv6.

Characteristic	IPv4	IPv6
VPC size	Up to 5 CIDRs from /16 to /28. This quota is adjustable.	Up to 5 CIDRs from /44 to /60 in increments of /4. This quota is adjustable.
Subnet size	From /16 to /28.	From /44 to /64 in increments of /4.
Address selection	You can choose the IPv4 CIDR block for your VPC or you can allocate a CIDR block from Amazon VPC IP Address Manager (IPAM). For more information, see What is IPAM? in the *Amazon VPC IPAM User Guide*.	You can bring your own IPv6 CIDR block to AWS for your VPC, choose an Amazon-provided IPv6 CIDR block, or you can allocate a CIDR block from Amazon VPC IP Address Manager (IPAM). For more informati on, see What is IPAM? in the *Amazon VPC IPAM User Guide*.

Characteristic	IPv4	IPv6
Internet access	Requires an internet gateway.	Requires an internet gateway. Supports outbound-only communica tion using an egress-only internet gateway.
Elastic IP addresses	Supported. Gives an EC2 instance a permanent, static public IPv4 address.	Not supported. EIPs keep the public IPv4 address of an instance static on instance restart. IPv6 addresses are static by default.
NAT gateways	Supported. Instances in private subnets can connect to the internet using a public NAT gateway or to resources in other VPCs using a private NAT gateway.	Supported. You can use a NAT gateway with NAT64 to enable instances in IPv6-only subnets to communicate with IPv4-only resources within VPCs, between VPCs, in your on-premises networks, or over the internet.
DNS names	Instances receive Amazon-provided IPBN or RBN-based DNS names. The DNS name resolves to the DNS records selected for the instance.	Instance receive Amazon-provided IPBN or RBN-based DNS names. The DNS name resolves to the DNS records selected for the instance.

Consolidate and manage network CIDR blocks with managed prefix lists

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

There are two types of prefix lists:

- **Customer-managed prefix lists** — Sets of IP address ranges that you define and manage. You can share your prefix list with other AWS accounts, enabling those accounts to reference the prefix list in their own resources.
- **AWS-managed prefix lists** — Sets of IP address ranges for AWS services. You cannot create, modify, share, or delete an AWS-managed prefix list.

Contents

- Prefix lists concepts and rules
- Identity and access management for prefix lists
- Customer-managed prefix lists
- AWS-managed prefix lists
- Optimize AWS infrastructure management with prefix lists

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.

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

AWS-managed prefix lists

The following rules apply to AWS-managed prefix lists:

- You cannot create, modify, share, or delete an AWS-managed prefix list.
- Different AWS-managed prefix lists have a different weight when you use them. For more information, see AWS-managed prefix list weight.
- You cannot view the version number of an AWS-managed prefix list.

Identity and access management for prefix lists

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

To see a list of Amazon VPC actions and the resources and condition keys that you can use in an IAM policy, see Actions, resources, and condition keys for Amazon EC2 in the *Service Authorization Reference*.

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

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:GetManagedPrefixListAssociations",
```

```
                "ec2:GetManagedPrefixListEntries",
                "ec2:ModifyManagedPrefixList",
                "ec2:RestoreManagedPrefixListVersion"
            ],
            "Resource": "arn:aws:ec2:us-east-1:123456789012: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.

Customer-managed prefix lists

Customer-managed prefix lists allow you to define and maintain your own sets of IP address ranges, known as prefixes, within AWS. Instead of hardcoding these IP addresses into your various resources, you can create a centralized prefix list and reference it wherever needed. This not only simplifies the management of your IP addresses but also promotes consistency and reusability across your AWS landscape.

One of the standout features of customer-managed prefix lists is the ability to share them with other AWS accounts. By granting access to your prefix lists, you can enable other teams or organizations to leverage your defined IP address ranges in their own resources. This collaborative approach fosters a more cohesive and efficient cloud experience, where IP address management is shared and synchronized.

In the sections that follow, we'll dive deeper into the practical aspects of working with customer-managed prefix lists, including step-by-step guidance on creating, managing, and sharing your IP address ranges.

Tasks

- Work with customer-managed prefix lists

Work with customer-managed prefix lists

This section describes how to work with customer-managed prefix lists.

Contents

- Create a prefix list
- View prefix lists
- View the entries for a prefix list
- View associations (references) for your prefix list
- Modify a prefix list
- Resize a prefix list
- Restore a previous version of a prefix list
- Delete a prefix list
- Share customer-managed prefix lists

Create a prefix list

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

Limitation

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

To create a prefix list using the console

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

8. (Optional) For **Tags**, add tags to the prefix list to help you identify it later.
9. Choose **Create prefix list**.

To create a prefix list using the AWS CLI

Use the create-managed-prefix-list command.

View prefix lists

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

To view prefix lists using the console

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

To view prefix lists using the AWS CLI

Use the describe-managed-prefix-lists command.

View the entries for a prefix list

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

To view the entries for a prefix list using the console

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

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

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

View associations (references) for your prefix list

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

Limitation

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

To view prefix list associations using the console

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

To view prefix list associations using the AWS CLI

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

Modify a prefix list

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

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

Considerations

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

To modify a prefix list using the console

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

2. In the navigation pane, choose **Managed Prefix Lists**.
3. Select the checkbox for prefix list, and choose **Actions**, **Modify prefix list**.
4. For **Prefix list name**, enter a new name for the prefix list.
5. For **Prefix list entries**, choose **Remove** to remove an existing entry. To add a new entry, choose **Add new entry** and enter the CIDR block and a description for the entry.
6. Choose **Save prefix list**.

To modify a prefix list using the AWS CLI

Use the modify-managed-prefix-list command.

Resize a prefix list

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

To resize a prefix list using the console

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

To resize a prefix list using the AWS CLI

Use the modify-managed-prefix-list command.

Restore a previous version of a prefix list

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

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

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

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

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

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

Delete a prefix list

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

Limitation

You cannot delete an AWS-managed prefix list.

To delete a prefix list using the console

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

To delete a prefix list using the AWS CLI

Use the delete-managed-prefix-list command.

Share customer-managed prefix lists

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

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

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

For more information about AWS RAM, see the AWS RAM User Guide. For more information quotas, see Service quotas in the AWS RAM User Guide.

> **Important**
>
> There are no additional charges for sharing prefix lists.

Contents

- Shared prefix list permissions
- Work with shared prefix lists

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.

Work with shared prefix lists

AWS prefix lists provide a convenient way to manage and reference the IP address ranges used by various AWS services. In addition to the AWS-managed prefix lists, you also can create and share your own customer-managed prefix lists with other AWS accounts.

Sharing prefix lists can be particularly useful for organizations with complex networking requirements or those that need to coordinate IP address usage across multiple AWS workloads. By sharing a prefix list, you can ensure consistent IP address management and simplify networking configurations for your collaborators.

This section describes and how to share prefix lists and how to identify and use prefix lists that have been shared with your account.

Contents

- Share a prefix list
- Unshare a shared prefix list
- Identify a shared prefix list
- Identify references to a shared prefix list

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.

> **Important**
>
> - To share a prefix list, you must own it. You cannot share a prefix list that has been shared with you. You cannot share an AWS-managed prefix list.

- To share a prefix list with your organization or an organizational unit in AWS Organizations, you must enable sharing with AWS Organizations. For more information, see [Enable sharing with AWS Organizations] in the *AWS RAM User Guide*.

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

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. 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.

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.

AWS-managed prefix lists

AWS-managed prefix lists are sets of IP address ranges for AWS services. These prefix lists are maintained by Amazon Web Services and provide a way to reference the IP addresses used by various AWS offerings. This can be particularly useful when configuring security groups or other network-level controls within a VPC.

The prefix lists cover a wide range of AWS services, including S3 and DynamoDB, and many others. By using the managed prefix lists, you can ensure that your network configurations are up-to-date and properly account for the IP addresses used by the AWS services you depend on. This can help simplify networking tasks and reduce the administrative overhead of manually maintaining lists of IP addresses.

In addition to the practical benefits, using the managed prefix lists also aligns with AWS security best practices. By relying on the authoritative IP address information provided by AWS, you can minimize the risk of misconfiguration or unexpected connectivity issues. This can be especially important for mission-critical applications or workloads with strict compliance requirements.

Contents

- Available AWS-managed prefix lists
- AWS-managed prefix list weight
- Use an AWS-managed prefix list

Available AWS-managed prefix lists

The following services provide AWS-managed prefix lists.

AWS service	Prefix list name	Weight
Amazon CloudFront	com.amazonaws.global.cloudfront.origin-facing (IPv4) com.amazonaws.global.ipv6.cloudfront.origin-facing (IPv6)	55
Amazon DynamoDB	com.amazonaws.*region*.dynamodb	1

AWS service	Prefix list name	Weight
Amazon EC2 Instance Connect	com.amazonaws.*region*.ec2-instance-connect	2
	com.amazonaws.*region*.ipv6.ec2-instance-connect	2
AWS Ground Station	com.amazonaws.global.groundstation	5
Amazon Route 53	com.amazonaws.*region*.ipv6.route53-healthchecks	25
	com.amazonaws.*region*.route53-healthchecks	25
Amazon S3	com.amazonaws.*region*.s3	1
Amazon S3 Express One Zone	com.amazonaws.*region*.s3express	6
Amazon VPC Lattice	com.amazonaws.*region*.vpc-lattice	10
	com.amazonaws.*region*.ipv6.vpc-lattice	10

To view the AWS-managed prefix lists using the console

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

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

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

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

AWS-managed prefix list weight

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

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

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

Use an AWS-managed prefix list

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

As with customer-managed prefix lists, you can use AWS-managed prefix lists with AWS resources such as security groups and route tables. For more information, see Optimize AWS infrastructure management with prefix lists.

Optimize AWS infrastructure management with prefix lists

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

Resources

- VPC security groups
- Subnet route tables
- Transit gateway route tables
- AWS Network Firewall rule groups
- Amazon Managed Grafana network access control
- AWS Outposts rack local gateways

VPC security groups

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

Important

You can't modify an existing rule to use a prefix list. You have to create a new rule to use a prefix list.

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 **Custom**. Then, in the next field, under **Prefix lists**, choose the ID of the prefix list.
6. Choose **Save rules**.

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

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

Subnet route tables

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

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

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

7. Choose **Save changes**.

To reference a prefix list in a route table using the AWS CLI

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

Transit gateway route tables

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

AWS Network Firewall rule groups

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

Amazon Managed Grafana network access control

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

AWS Outposts rack local gateways

Each AWS Outposts rack provides a local gateway that allows you to connect your Outpost resources with your on-premises networks. You can group CIDRs that you frequently use in a prefix list and reference this list as a route target in your local gateway route table. For more information, see Manage local gateway route table routes in the *AWS Outposts User Guide for racks*.

AWS IP address ranges

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

Considerations

- We publish the IP address ranges for services that customers commonly use to perform egress filtering. We don't publish the IP address ranges for all services.
- Services use their IP address ranges to communicate with other services or to communicate with a customer network.
- The IP address ranges that you bring to AWS through bring your own IP addresses (BYOIP) are not included in the `.json` file. For more information, see Advertise your address range through AWS in the *Amazon EC2 User Guide*.

Some services publish their address ranges using AWS-managed prefix lists. For more information, see the section called “Available AWS-managed prefix lists”.

Contents

- Download the JSON file
- Egress control
- Geolocation feed
- Find the IP address ranges for AWS services
- Syntax for AWS IP address range JSON
- AWS IP address ranges notifications

Download the JSON file

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

The following is an example **curl** command that saves the JSON file to the current directory.

```
curl -O https://ip-ranges.amazonaws.com/ip-ranges.json
```

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

To receive notifications of updates to the JSON file, see AWS IP address ranges notifications.

Egress control

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

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

Geolocation feed

The IP address ranges in `ip-ranges.json` are by AWS Region. However, a Local Zone is not in the same physical location as its parent Region. The geolocation data published in geo-ip-feed.csv accounts for Local Zones. The data follows RFC 8805.

Find the IP address ranges for AWS services

The AWS IP address range JSON file provided by AWS can be a valuable resource for finding the IP addresses of various AWS services and leveraging that information to enhance your network security and access control. By parsing the detailed data contained within this JSON file, you can precisely identify the IP address ranges associated with specific AWS services and Regions.

For example, you can utilize the IP address ranges to configure robust network security policies, setting up granular firewall rules to allow or deny access to certain AWS resources. This information can also be useful for a variety of AWS Network Firewall tasks. This level of control is crucial for protecting your applications and data, ensuring that only authorized traffic can reach the necessary AWS services. Additionally, having this IP intelligence can help you ensure your applications are properly configured to communicate with the right AWS endpoints, improving overall reliability and performance.

Beyond just firewall rules, the `ip-ranges.json` file can also be leveraged to configure sophisticated egress filtering on your network infrastructure. By understanding the destination IP address ranges for different AWS services, you can set up routing policies or leverage advanced network security solutions like to selectively permit or block outbound traffic based on its intended destination. This egress control is essential for mitigating the risk of data leakage and unauthorized access.

It's important to note that the `ip-ranges.json` file is regularly updated, so maintaining an up-to-date local copy is crucial to ensure you have the most accurate and current information. By continuously leveraging the contents of this file, you can efficiently manage network access and security for your AWS-based applications, strengthening your overall cloud security posture.

The following examples can help you filter the AWS IP address ranges to just what you are looking for. On Linux, you can download and use the the jq tool to parse a local copy of the JSON file. The AWS Tools for Windows PowerShell includes a cmdlet, Get-AWSPublicIpAddressRange, that you can use to parse this JSON file. For more information, see the following blog: Querying the Public IP Address Ranges for AWS.

To get the JSON file, see the section called “Download”. For more information about the syntax of the JSON file, see the section called “Syntax”.

Examples

- Get the file creation date
- Get the IP addresses for a specific Region
- Get all IPv4 addresses
- Get all IPv4 addresses for a specific service
- Get all IPv4 addresses for a specific service in a specific Region
- Get all IPv6 addresses
- Get all IPv6 addresses for a specific service
- Get all IP addresses for a specific border group

Get the file creation date

The following example gets the creation date of `ip-ranges.json`.

jq

```
$ jq .createDate < ip-ranges.json
```

```
"2024-08-01-17-22-15"
```

PowerShell

```
PS C:\> Get-AWSPublicIpAddressRange -OutputPublicationDate

Thursday, August 1, 2024 9:22:35 PM
```

Get the IP addresses for a specific Region

The following example filters the JSON file for the IP addresses for the specified Region.

jq

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

{
  "ip_prefix": "23.20.0.0/14",
  "region": "us-east-1",
  "network_border_group": "us-east-1",
  "service": "AMAZON"
},
{
  "ip_prefix": "50.16.0.0/15",
  "region": "us-east-1",
  "network_border_group": "us-east-1",
  "service": "AMAZON"
},
{
  "ip_prefix": "50.19.0.0/16",
  "region": "us-east-1",
  "network_border_group": "us-east-1",
  "service": "AMAZON"
},
...
```

PowerShell

```
PS C:\> Get-AWSPublicIpAddressRange -Region us-east-1

IpPrefix        Region      NetworkBorderGroup        Service
```

```
--------        ------      -------                     -------
23.20.0.0/14    us-east-1   us-east-1                   AMAZON
50.16.0.0/15    us-east-1   us-east-1                   AMAZON
50.19.0.0/16    us-east-1   us-east-1                   AMAZON
...
```

Get all IPv4 addresses

The following example filters the JSON file for the IPv4 addresses.

jq

```
$ jq -r '.prefixes | .[].ip_prefix' < ip-ranges.json

23.20.0.0/14
27.0.0.0/22
43.250.192.0/24
...
```

PowerShell

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

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

Get all IPv4 addresses for a specific service

The following example filters the JSON file for the IPv4 addresses for the specified service.

jq

```
$ jq -r '.prefixes[] | select(.service=="GLOBALACCELERATOR") | .ip_prefix' < ip-
ranges.json
```

```
13.248.117.0/24
15.197.34.0/23
15.197.36.0/22
...
```

PowerShell

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

IpPrefix
--------
13.248.117.0/24
15.197.34.0/23
15.197.36.0/22
...
```

Get all IPv4 addresses for a specific service in a specific Region

The following example filters the JSON file for the IPv4 addresses for the specified service in the specified Region.

jq

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

13.248.124.0/24
99.82.166.0/24
99.82.171.0/24
...
```

PowerShell

```
PS C:\> Get-AWSPublicIpAddressRange -Region us-east-1 -ServiceKey GLOBALACCELERATOR
 | where {$_.IpAddressFormat -eq "Ipv4"} | select IpPrefix

IpPrefix
--------
13.248.117.0/24
99.82.166.0/24
```

```
99.82.171.0/24
...
```

Get all IPv6 addresses

The following example filters the JSON file for the IPv6 addresses.

jq

```
$ jq -r '.ipv6_prefixes | .[].ipv6_prefix' < ip-ranges.json

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

PowerShell

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

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

Get all IPv6 addresses for a specific service

The following example filters the JSON file for the IPv6 addresses for the specified service.

jq

```
$ jq -r '.ipv6_prefixes[] | select(.service=="GLOBALACCELERATOR") | .ipv6_prefix' <
 ip-ranges.json

2600:1f01:4874::/47
2600:1f01:4802::/47
2600:1f01:4860::/47
```

```
2600:9000:a800::/40
...
```

PowerShell

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

IpPrefix
--------
2600:1f01:4874::/47
2600:1f01:4802::/47
2600:1f01:4860::/47
2600:9000:a800::/40
...
```

Get all IP addresses for a specific border group

The following example filters the JSON file for all IP addresses for the specified border group.

jq

```
$ jq -r '.prefixes[] | select(.network_border_group=="us-west-2-lax-1")
 | .ip_prefix' < ip-ranges.json
70.224.192.0/18
52.95.230.0/24
15.253.0.0/16
...
```

PowerShell

```
PS C:\> Get-AWSPublicIpAddressRange | where {$_.NetworkBorderGroup -eq "us-west-2-
lax-1"} | select IpPrefix

IpPrefix
--------
70.224.192.0/18
52.95.230.0/24
15.253.0.0/16
...
```

Syntax for AWS IP address range JSON

AWS publishes its current IP address ranges in JSON format. To get the JSON file, see the section called "Download". The syntax of the JSON file is as follows.

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

syncToken

The publication time, in Unix epoch time format.

Type: String

Example: `"syncToken": "1416435608"`

createDate

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

Type: String

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

prefixes

The IP prefixes for the IPv4 address ranges.

Type: Array

ipv6_prefixes

The IP prefixes for the IPv6 address ranges.

Type: Array

ip_prefix

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

Type: String

Example: `"ip_prefix": "198.51.100.2/24"`

ipv6_prefix

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

Type: String

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

network_border_group

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

Type: String

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

region

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

Type: String

Valid values: `af-south-1` | `ap-east-1` | `ap-east-2` | `ap-northeast-1` | `ap-northeast-2` | `ap-northeast-3` | `ap-south-1` | `ap-south-2` | `ap-southeast-1` | `ap-southeast-2` | `ap-`

southeast-3 | ap-southeast-4 | ap-southeast-5 | ap-southeast-7 | ca-central-1 | ca-west-1 | cn-north-1 | cn-northwest-1 | eu-central-1 | eu-central-2 | eu-north-1 | eu-south-1 | eu-south-2 | eu-west-1 | eu-west-2 | eu-west-3 | il-central-1 | mx-central-1 | me-central-1 | me-south-1 | sa-east-1 | us-east-1 | us-east-2 | us-gov-east-1 | us-gov-west-1 | us-west-1 | us-west-2 | GLOBAL

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

service

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

Type: String

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

Example: "service": "AMAZON"

Range overlaps

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

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

Learn more

This section provides links to additional information for different service codes.

- `AMAZON_APPFLOW` – IP address ranges
- `AMAZON_CONNECT` – Set up your network
- `CHIME_MEETINGS` – Configuring for media and signaling
- `CLOUDFRONT` – Locations and IP address ranges of CloudFront edge servers
- `DYNAMODB` – IP address ranges
- `EC2` – Public IPV4 addresses
- `EC2_INSTANCE_CONNECT` – EC2 Instance Connect prerequisites
- `GLOBALACCELERATOR` – Location and IP address ranges of Global Accelerator edge servers
- `ROUTE53` – IP address ranges of Amazon Route 53 servers
- `ROUTE53_HEALTHCHECKS` – IP address ranges of Amazon Route 53 servers
- `ROUTE53_HEALTHCHECKS_PUBLISHING` – IP address ranges of Amazon Route 53 servers
- `WORKSPACES_GATEWAYS` – PCoIP gateway servers

Release notes

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

Description	Release date
Added the `IVS_LOW_LATENCY` service code.	July 29, 2025
Added the `AURORA_DSQL` service code.	May 21, 2025
Added the `IVS_REALTIME` service code.	June 11, 2024
Added the `MEDIA_PACKAGE_V2` service code.	May 9, 2023
Added the `CLOUDFRONT_ORIGIN_FACING` service code.	October 12, 2021
Added the `ROUTE53_RESOLVER` service code.	June 24, 2021

Description	Release date
Added the EBS service code.	May 12, 2021
Added the KINESIS_VIDEO_STREAMS service code.	November 19, 2020
Added the CHIME_MEETINGS and CHIME_VOICECONNECTOR service codes.	June 19, 2020
Added the AMAZON_APPFLOW service code.	June 9, 2020
Add support for the network border group.	April 7, 2020
Added the WORKSPACES_GATEWAYS service code.	March 30, 2020
Added the ROUTE53_HEALTHCHECK_PUBLISHING service code.	January 30, 2020
Added the API_GATEWAY service code.	September 26, 2019
Added the EC2_INSTANCE_CONNECT service code.	June 26, 2019
Added the DYNAMODB service code.	April 25, 2019
Added the GLOBALACCELERATOR service code.	December 20, 2018
Added the AMAZON_CONNECT service code.	June 20, 2018
Added the CLOUD9 service code.	June 20, 2018
Added the CODEBUILD service code.	April 19, 2018
Added the S3 service code.	February 28, 2017
Added support for IPv6 address ranges.	August 22, 2016
Initial release	November 19, 2014

AWS IP address ranges notifications

AWS publishes its current IP address ranges in JSON format. Whenever there is a change to the AWS IP address ranges, we send notifications to subscribers of the Amazon SNS topic named `AmazonIpSpaceChanged`. For more information about the syntax of the JSON file, see the section called "Syntax".

The payload of the notification contains information in the following format.

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

create-time

The creation date and time.

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

synctoken

The publication time, in Unix epoch time format.

md5

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

url

The location of the `ip-ranges.json` file. For more information, see the section called "Download".

You can subscribe to receive notifications as follows.

To subscribe to AWS IP address range notifications

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.

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

   a. For **Topic ARN**, copy the following Amazon Resource Name (ARN):

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

   b. For **Protocol**, choose the protocol to use (for example, `Email`).

   c. For **Endpoint**, type the endpoint to receive the notification (for example, your email address).

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

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

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

To unsubscribe from AWS IP address ranges notifications

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home.
2. In the navigation pane, choose **Subscriptions**.
3. Select the check box for the subscription.
4. Choose **Actions**, **Delete subscriptions**.
5. When prompted for confirmation, choose **Delete**.

For more information about Amazon SNS, see the *Amazon Simple Notification Service Developer Guide*.

IPv6 support for your VPC

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

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

Considerations

- There is no migration path from IPv4-only subnets to IPv6-only subnets.
- This example assumes that you have an existing VPC with public and private subnets. For information about creating a new VPC for use with IPv6, see the section called "Create a VPC".
- Before you begin using IPv6, ensure that you have read the features of IPv6 addressing for Amazon VPC: Compare IPv4 and IPv6.

Contents

- Add IPv6 support for your VPC
- Example dual-stack VPC configuration

Add IPv6 support for your VPC

The following table provides an overview of the process to enable IPv6 for your VPC.

Contents

- Step 1: Associate an IPv6 CIDR block with your VPC and subnets
- Step 2: Update your route tables
- Step 3: Update your security group rules
- Step 4: Assign IPv6 addresses to your instances

Step	Notes
Step 1: Associate an IPv6 CIDR block with your VPC and subnets	Associate an Amazon-provided or BYOIP IPv6 CIDR block with your VPC and with your subnets.
Step 2: Update your route tables	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	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: Assign IPv6 addresses to your instances	Assign IPv6 addresses to your instances from the IPv6 address range of your subnet.

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.
4. Choose **Actions**, **Edit CIDRs** and then choose **Add new IPv6 CIDR**.
5. Select one of the following options, and then choose **Select CIDR**:

- **Amazon-provided IPv6 CIDR block** – Use an IPv6 CIDR block from Amazon's pool of IPv6 addresses. For **Network Border Group**, choose the group from which AWS advertises IP addresses.
- **IPAM-allocated IPv6 CIDR block** – Use an IPv6 CIDR block from an IPAM pool. Choose the IPAM pool and the IPv6 CIDR block.
- **IPv6 CIDR owned by me** – Use an IPv6 CIDR block from your IPv6 address pool (BYOIP). Choose the IPv6 address pool and the IPv6 CIDR block.

6. Choose **Close**.

To associate an IPv6 CIDR block with a subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Subnets**.
3. Select a subnet.
4. Choose **Actions**, **Edit IPv6 CIDRs** and then choose **Add IPv6 CIDR**.
5. Edit the CIDR block as needed (for example, replace the `00`).
6. Choose **Save**.
7. Repeat this procedure for any other subnets in your VPC.

For more information, see IPv6 VPC CIDR blocks.

Step 2: Update your route tables

When you associate an IPv6 CIDR block with your VPC, we automatically add a local route to each route table for the VPC to allow IPv6 traffic within the VPC.

You must update the route tables for your public subnets to enable instances (such as web servers) to use the internet gateway for IPv6 traffic. You must also update the route tables for your private subnets to enable instances (such as database instances) to use an egress-only internet gateway for IPv6 traffic, because NAT gateways do not support IPv6.

To update the 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 **Subnets**. Select the public subnet. On the **Route table** tab, choose the route table ID to open the details page for the route table.
3. Select the route table. On the **Routes** tab, choose **Edit routes**.
4. Choose **Add route**. Choose `::/0` for **Destination**. Choose the ID of the internet gateway for **Target**.
5. Choose **Save changes**.

To update the route table for a private subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Egress-only internet gateways**. Choose **Create egress only internet gateway**. Choose your VPC from **VPC**, and then choose **Create egress only internet gateway**.

   For more information, see Enable outbound IPv6 traffic using an egress-only internet gateway.
3. In the navigation pane, choose **Subnets**. Select the private subnet. On the **Route table** tab, choose the route table ID to open the details page for the route table.
4. Select the route table. On the **Routes** tab, choose **Edit routes**.
5. Choose **Add route**. Choose `::/0` for **Destination**. Choose the ID of the egress-only internet gateway for **Target**.
6. Choose **Save changes**.

> **Note**
>
> A route table cannot have the same destination (::/0) pointing to both an internet gateway and an egress-only internet gateway simultaneously. If you receive an error message stating "There are existing ipv6 routes with next hop as internet Gateway" when configuring an egress-only internet gateway, you must first remove the existing IPv6 route to the internet gateway before adding the route to the egress-only internet gateway.

For more information, see Example routing options.

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 don't 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.

To update your inbound 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 inbound rules**.
4. For each rule that allows IPv4 traffic, choose **Add rule** and configure the rule to allow the corresponding IPv6 traffic. For example, to add a rule that allows all HTTP traffic over IPv6, choose **HTTP** for **Type** and `::/0` for **Source**.
5. When you are finished adding rules, choose **Save rules**.

Update your outbound security group rules

When you associate an IPv6 CIDR block with your VPC, we automatically add an outbound rule to the security groups for the VPC that allows all IPv6 traffic. 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.

Update your network ACL rules

When you associate an IPv6 CIDR block with a VPC, we automatically add rules to the default network ACL to allow IPv6 traffic. However, if you modified your default network ACL or if you've created a custom network ACL, you must manually add rules for IPv6 traffic. For more information, see [Add and delete rules](#).

Step 4: Assign IPv6 addresses to your instances

All current generation instance types support IPv6. If your instance type does not support IPv6, you must resize the instance to a supported instance type before you can assign an IPv6 address. The process that you'll use depends on whether the new instance type that you choose is compatible with the current instance type. For more information, see Change the instance type in the *Amazon EC2 User Guide*. If you must launch an instance from a new AMI to support IPv6, you can assign an IPv6 address to your instance during launch.

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 (for example, eth0) for the instance. For more information, see Assign an IPv6 address to an instance in the *Amazon EC2 User Guide*.

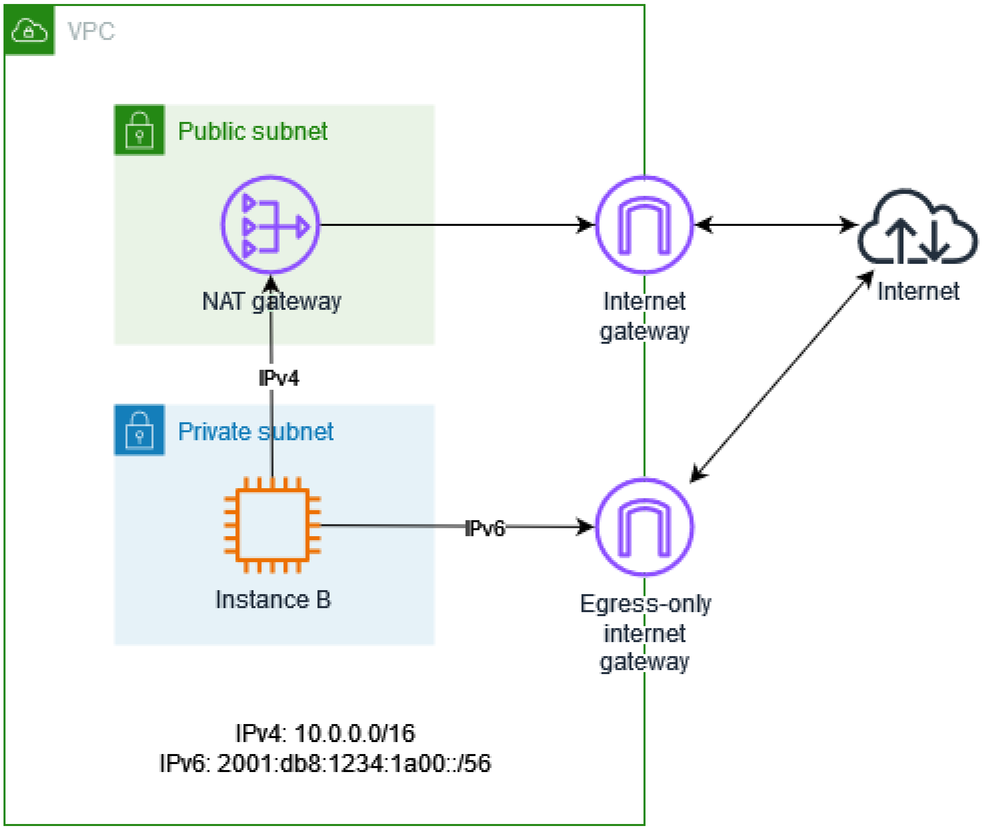
You can connect to an instance using its IPv6 address. For more information, see Connect to your Linux instance using an SSH client in the *Amazon EC2 User Guide*.

If you launched your instance using an AMI for a current version of your operating system, your instance is configured for IPv6. If you can't ping an IPv6 address from your instance, refer to the documentation for your operating system to configure IPv6.

Example dual-stack VPC configuration

With a dual-stack configuration, you can use both IPv4 and IPv6 addresses for communication between resources in your VPC and resources over the internet.

The following diagram represents the architecture of your VPC. Your VPC has a public subnet and a private subnet. The VPC and subnets have both an IPv4 CIDR block and an IPv6 CIDR block. There is an EC2 instance in the private subnet that has both an IPv4 address and an IPv6 address. The instance can send outbound IPv4 traffic to the internet using a NAT gateway and outbound IPv6 traffic to the internet using an egress-only internet gateway.



Route table for public subnet

The following is the route table for the public subnet. The first two entries are the local routes. The third entry sends all IPv4 traffic to the internet gateway. Note that the fourth entry is necessary only if you plan to launch EC2 instances with IPv6 addresses in the public subnet.

Destination	Target
VPC IPv4 CIDR	local
VPC IPv6 CIDR	local
0.0.0.0/0	*internet-gateway-id*
::/0	*internet-gateway-id*

Route table for the private subnet

The following is the route table for the private subnet. The first two entries are the local routes. The third entry sends all IPv4 traffic to the NAT gateway. The last entry sends all IPv6 traffic to the egress-only internet gateway.

Destination	Target
VPC IPv4 CIDR	local
VPC IPv6 CIDR	local
0.0.0.0/0	*nat-gateway-id*
::/0	*egress-only-gateway-id*

AWS services that support IPv6

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

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

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

Services that support IPv6

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

Service name	Dual stack support	IPv6 only support	Public endpoints support IPv6	Private endpoints support IPv6 [1]
AWS Amplify	Yes	No	Yes	
Amazon API Gateway	Yes	No	Yes	Yes
AWS App Mesh	Yes	Yes	Yes	No
AWS AppConfig	Yes	No	Yes	Yes
AWS Application Discovery Service	Yes	No	Yes	Yes
Application Recovery Controller (ARC)	Yes	No	Yes	
Amazon AppStream 2.0	Yes	No	No	No
AWS AppSync[2]	Partial	No	Partial	No
Amazon Athena	Yes	No	Yes	Yes
Amazon Aurora	Yes	No	Yes	No
AWS B2B Data Interchange	Yes	No	Yes	Yes
AWS Backup	Yes	No	Yes	Yes
AWS Batch	Yes	No	Yes	Yes
AWS Billing and Cost Management Data Exports	Yes	No	Yes	Yes
AWS Billing and Cost Management Pricing Calculator	Yes	No	Yes	Yes

Service name	Dual stack support	IPv6 only support	Public endpoints support IPv6	Private endpoints support IPv6 [1]
AWS Billing Conductor	Yes	No	Yes	Yes
AWS Budgets	Yes	No	Yes	Yes
Amazon Braket	Yes	Yes	Yes	Yes
AWS Certificate Manager	Yes	No	Yes	No
Amazon Comprehend	Yes	Yes	Yes	Yes
AWS Clean Rooms	Yes	Yes	Yes	Yes
AWS Clean Rooms ML	Yes	Yes	Yes	Yes
AWS Cloud9	Yes	No	Yes	
AWS Cloud Control API	Yes	No	Yes	Yes
Amazon CloudFront	Yes	Yes	Yes	
AWS CloudHSM	Yes	No	Yes	Yes
AWS CloudTrail	Yes	No	Yes	Yes
Amazon CloudWatch	Yes	Yes	Yes	Yes
Amazon CloudWatch Logs	Yes	Yes	Yes	Yes
Amazon CloudWatch Observability Access Manager	Yes	Yes	Yes	
Amazon CloudWatch Synthetics	Yes	No	Yes	Yes

Service name	Dual stack support	IPv6 only support	Public endpoints support IPv6	Private endpoints support IPv6 [1]
AWS Cloud Map	Yes	Yes	Yes	Yes
AWS Cloud WAN	Yes	No	Yes	Yes
AWS CodeArtifact	Yes	No	Yes	Yes
Amazon Connect Customer Profiles	Yes	No	Yes	Yes
Amazon CodeGuru Profiler	Yes	No	Yes	Yes
AWS Cost Explorer	Yes	No	Yes	Yes
AWS Cost Optimization Hub	Yes	No	Yes	Yes
AWS Elastic Beanstalk	No	No	Yes	Yes
Amazon Cognito	Yes	No	Yes	
Amazon Data Firehose	No	No	Yes	Yes
Amazon Data Lifecycle Manager	Yes	No	Yes	Yes
AWS Database Migration Service	Yes	No	No	No
AWS DataSync	Yes	Yes	Yes	Yes
AWS Deadline Cloud	Yes	No	Yes	Yes
Amazon Detective	Yes	Yes	Yes	
AWS Direct Connect	Yes	Yes	No	Yes

Service name	Dual stack support	IPv6 only support	Public endpoints support IPv6	Private endpoints support IPv6 [1]
Amazon EBS direct APIs	Yes	No	Yes	Yes
Amazon EC2	Yes	Yes	Yes	No
Amazon ECR	Yes	No	Yes	No
Amazon ECS	Yes	No	Yes	Yes
Amazon EFS	Yes	Yes	Yes	Yes
Amazon EKS	Partial	Partial	Yes	Yes
Elastic Load Balancing	Partial	Partial	No	No
Amazon ElastiCache	Yes	Yes	No	No
AWS End User Messaging Social	Yes	No	Yes	No
AWS Entity Resolution	Yes	No	Yes	Yes
AWS Fargate	Yes	No	No	No
Amazon FSx	No	No	Yes	Yes
Amazon GameLift Streams	Yes	No	Yes	No
AWS Global Accelerator	Yes	No	Yes	
AWS Glue	Yes	No	No	Yes
Amazon Managed Grafana [3]	Yes	No	Yes	Yes
AWS Ground Station [4]	Yes	No	Yes	Yes

Service name	Dual stack support	IPv6 only support	Public endpoints support IPv6	Private endpoints support IPv6 [1]
AWS Identity and Access Management (IAM)	Yes	Yes	Yes	Yes
AWS IAM Access Analyzer	Yes	No	Yes	Yes
Amazon Inspector	Yes	Yes	Yes	Yes
AWS IoT	Yes	No	Yes	No
AWS IoT FleetWise	Yes	No	Yes	Yes
AWS IoT SiteWise	Yes	No	Yes	Yes
AWS IoT TwinMaker	Yes	No	Yes	Yes
AWS IoT Wireless	Yes	No	Yes	Yes
AWS Key Management Service	Yes	Partial	Yes	Yes
Amazon Kinesis Data Streams	Yes	No	Yes	No
AWS Lake Formation	No	No	No	Yes
AWS Lambda	Yes	No	Yes	Yes
Amazon Lightsail	Yes	Yes	Yes	Yes
Amazon Macie	Yes	No	Yes	Yes
AWS Mainframe Modernization	Yes	No	Yes	Yes
AWS Network Firewall	Yes	Yes	No	No

Service name	Dual stack support	IPv6 only support	Public endpoints support IPv6	Private endpoints support IPv6 [1]
AWS Network Manager	Yes	No	Yes	Yes
Amazon OpenSearch Service	Yes	No	Yes	No
AWS Organizations	Yes	No	Yes	No
Amazon Personalize	Yes	No	Yes	Yes
Amazon Pinpoint	Yes	No	Yes	No
Amazon Polly	Yes	No	Yes	Yes
AWS Private Certificate Authority	Yes	No	Yes	Yes
AWS Private CA Connector for Active Directory	Yes	No	Yes	Yes
AWS Private CA Connector for SCEP	Yes	No	Yes	Yes
AWS PrivateLink	Yes	Yes	Yes	
Amazon Managed Service for Prometheus	Yes	No	Yes	Yes
Amazon RDS	Yes	No	Yes	No
Recycle Bin	Yes	No	Yes	Yes
AWS re:Post Private	Yes	No	Yes	Yes
AWS Resource Access Manager	Yes	No	Yes	Yes

Service name	Dual stack support	IPv6 only support	Public endpoints support IPv6	Private endpoints support IPv6 [1]
AWS Resource Explorer	Yes	No	Yes	Yes
AWS Resource Groups	Yes	Yes	Yes	Yes
AWS Resource Groups Tagging API	Yes	Yes	Yes	Yes
Amazon Route 53	Yes	Yes	No	
Amazon S3	Yes	No	Yes	No
AWS Secrets Manager	Yes	No	Yes	No
Amazon Security Lake	Yes	No	Yes	Yes
AWS Shield	Yes	Yes	No	Yes
Amazon Simple Email Service	Yes	No	Yes	Yes
Amazon Simple Notification Service	Yes	No	Yes	Yes
Amazon Simple Queue Service	Yes	No	Yes	Yes
Amazon Simple Workflow Service	Yes	No	Yes	Yes
AWS Site-to-Site VPN	Yes	No	Yes	No
AWS Step Functions	Yes	No	Yes	Yes
Amazon Transcribe	Yes	Yes	Yes	Yes
AWS Transfer Family [5]	Yes	No	Yes	Yes

Service name	Dual stack support	IPv6 only support	Public endpoints support IPv6	Private endpoints support IPv6 [1]
AWS Transit Gateway	Yes	No	Yes	No
Amazon Translate	Yes	Yes	Yes	Yes
Amazon Verified Permissions	Yes	No	Yes	Yes
Amazon VPC	Yes	Yes	Yes	No
AWS WAF	Yes	Yes	No	Yes
Amazon WorkSpaces	Yes	No	No	No
AWS X-Ray	Yes	No	Yes	Yes
EC2 Image Builder	Yes	Yes	Yes	Yes

[1] An empty cell indicates that the service does not integrate with AWS PrivateLink.

[2] This entry represents IPv6 support for AWS AppSync GraphlQL and Event API configuration operations, through the AWS AppSync SDK API. IPv6 is not supported for client connections to customer managed AWS AppSync GraphQL and Event APIs.

[3] This entry represents IPv6 support for Grafana *workspace management* operations, such as updating workspaces and workspace permissions. There is no IPv6 support for general Grafana workspace operations, such as creating and editing dashboards or querying data sources.

[4] This entry represents IPv6 support for AWS Ground Station *control plane* operations, such as calling the AWS Ground Station API. IPv6 is not supported by the AWS Ground Station *data plane*, so make sure the resources you are delivering data to (such as Amazon EC2 instances) are accessible over IPv4.

[5] For more details on IPv6 support in AWS Transfer Family, visit Ipv6 limitations.

Additional IPv6 support

Compute

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

Networking and Content Delivery

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

Security, Identity, and Compliance

- Amazon Detective supports IPv6 addresses in its network-related findings and entity profiles.
- AWS Identity and Access Management (IAM) supports IPv6 addresses in IAM identity-based policies.
- Amazon Macie supports IPv6 addresses in personally identifiable information (PII).
- Amazon Security Lake supports IPv6 addresses across all operations on log sources and subscribers.

Management and Governance

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

Learn more

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

Configure a virtual private cloud

Amazon Virtual Private Cloud (VPC) is a fundamental building block, allowing you to provision a logically isolated virtual network within the AWS cloud. By creating your own VPC, you gain full control over the networking environment, including the ability to define IP address ranges, subnets, routing tables, and connectivity options.

Your AWS account contains a default VPC for each AWS Region. This default VPC comes pre-configured with settings that make it a convenient option for quickly launching resources. However, the default VPC may not always align with your long-term networking needs. This is where creating additional VPCs can be advantageous.

Creating additional VPCs offers several advantages over relying on the default VPC that comes provisioned with every new AWS account. With a self-managed VPC, you can architect the network topology to align precisely with your specific requirements, whether that's implementing a multi-tier application, connecting to on-premises resources, or segregating workloads by department or business unit.

In addition, creating multiple VPCs can enable greater security and isolation between your different applications or business units. Each VPC acts as a separate, virtual network, allowing you to apply distinct security policies, access controls, and routing configurations tailored to each environment.

Ultimately, the decision to use the default VPC or create one (or more) custom VPCs should be based on your specific application requirements, security needs, and long-term scalability goals. Investing the time to thoughtfully design your VPC infrastructure can pay dividends in the form of a robust, secure, and adaptable cloud networking foundation.

Contents

- VPC basics
- VPC configuration options
- Default VPCs
- Create a VPC
- Visualize the resources in your VPC
- Add or remove a CIDR block from your VPC
- DHCP option sets in Amazon VPC
- DNS attributes for your VPC

- Network Address Usage for your VPC
- Share your VPC subnets with other accounts
- Extend a VPC to a Local Zone, Wavelength Zone, or Outpost
- Delete your VPC
- Generate infrastructure-as-code from your VPC console actions with Console-to-Code

VPC basics

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

Contents

- VPC IP address range
- VPC diagram
- VPC resources

VPC IP address range

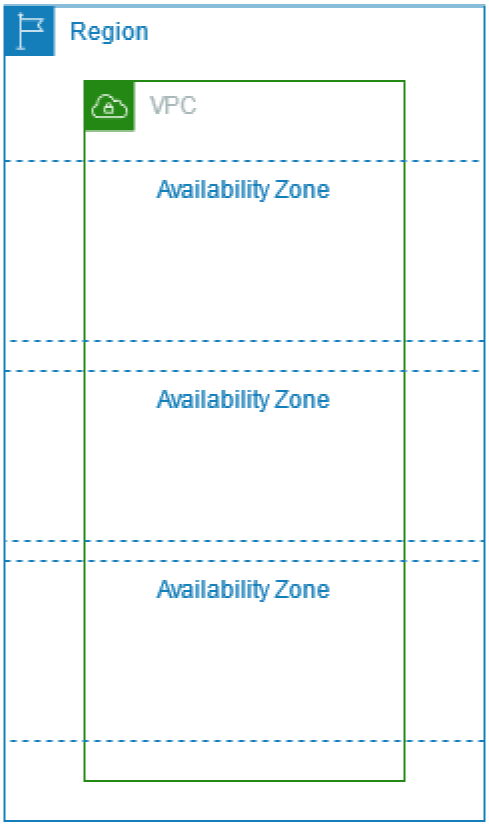
When you create a VPC, you specify its IP addresses as follows:

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

For more information, see IP addressing for your VPCs and subnets.

VPC diagram

The following diagram shows a VPC with no additional VPC resources. For example VPC configurations, see *Examples*.



VPC resources

Each VPC automatically comes with the following resources:

- Default DHCP option set
- Default network ACL
- Default security group
- Main route table

You can create the following resources for your VPC:

- Network ACLs
- Custom route tables
- Security groups
- Internet gateway
- NAT gateways

VPC configuration options

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

Availability Zones

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

CIDR blocks

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

DNS options

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

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

Internet gateway

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

Name

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

NAT gateways

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

Route tables

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

Subnets

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

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

For more information, see Subnets.

Tenancy

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

Default VPCs

When you start using Amazon VPC, you have a default VPC in each AWS Region. A default VPC comes with a public subnet in each Availability Zone, an internet gateway, and settings to enable

DNS resolution. Therefore, you can immediately start launching Amazon EC2 instances into a default VPC. You can also use services such as Elastic Load Balancing, Amazon RDS, and Amazon EMR in your default VPC.

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

You can add subnets to your default VPC. For more information, see the section called "Create a subnet".

Contents

- Default VPC components
- Default subnets
- Work with your default VPC and default subnets

Default VPC components

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

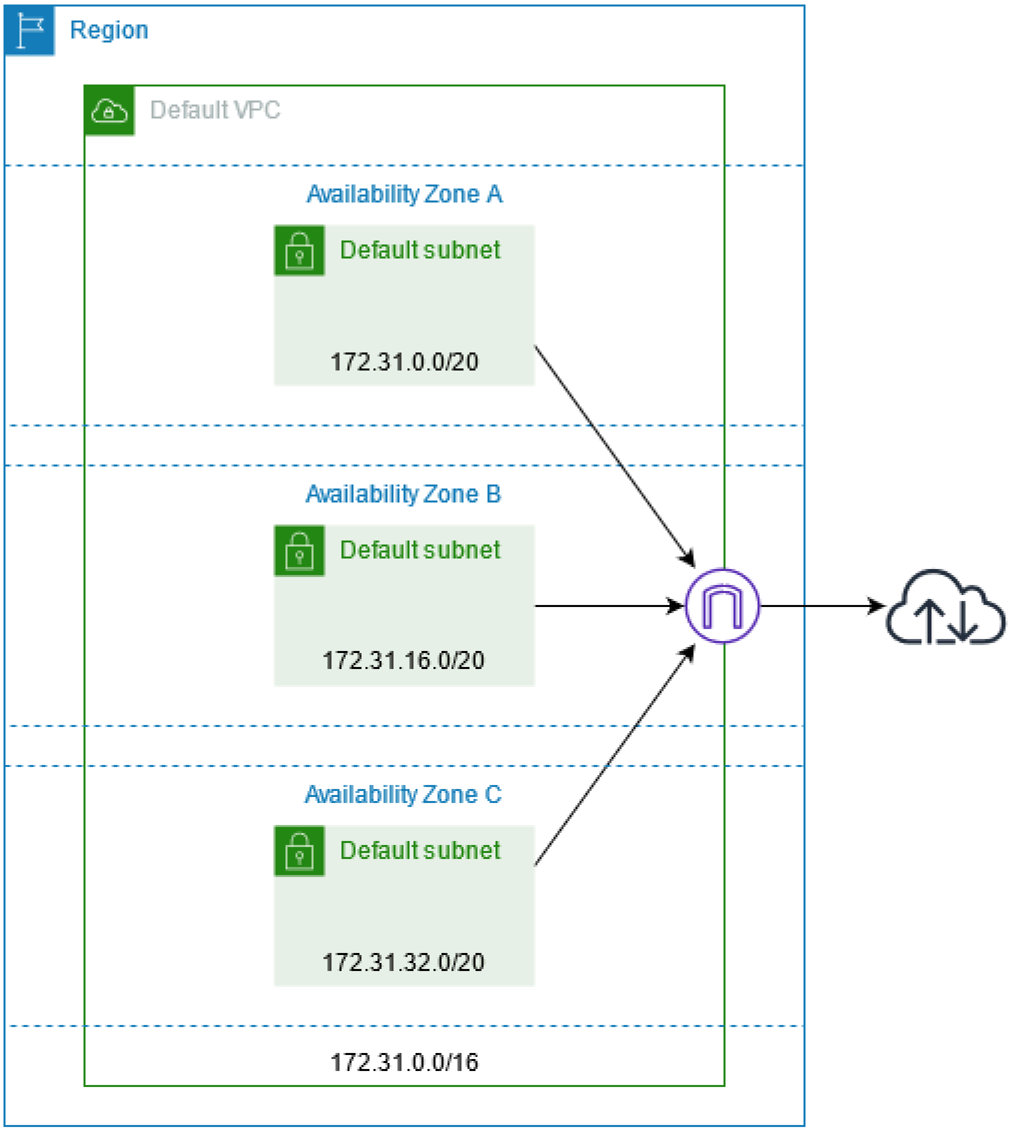
- Create a VPC with a size `/16` IPv4 CIDR block (`172.31.0.0/16`). This provides up to 65,536 private IPv4 addresses.
- Create a size `/20` default subnet in each Availability Zone. This provides up to 4,096 addresses per subnet, a few of which are reserved for our use.
- Create an internet gateway 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. To prevent

Amazon from creating an internet gateway, you would have to deny CreateDefaultVpc and CreateInternetGateway.

- To block all traffic to and from the internet gateways in your account, see Block public access to VPCs and subnets.

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



The following table shows the routes in the main route table for the default VPC.

Destination	Target
172.31.0.0/16	local

Destination	Target
0.0.0.0/0	*internet_gateway_id*

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

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

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

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

Default subnets

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

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

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.

Work with your default VPC and default subnets

This section describes how to work with default VPCs and default subnets.

Contents

- View your default VPC and default subnets
- Create a default VPC
- Create a default subnet
- Delete your default subnets and default VPC

View your default VPC and default subnets

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

To view your default VPC and subnets using the console

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

To describe your default VPC using the command line

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

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

To describe your default subnets using the command line

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

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.

Create a default VPC

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

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

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

To create a default VPC using the console

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

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

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.

If you've associated an IPv6 CIDR block with your default VPC, the new default subnet does not automatically receive an IPv6 CIDR block. Instead, you can associate an IPv6 CIDR block with the default subnet after you create it. For more information, see Add or remove an IPv6 CIDR block from your subnet.

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.

Delete your default subnets and default VPC

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

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

If you delete a default subnet, you can create a new one. For more information, see Create a default subnet. 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 IP addressing attributes of your subnet. You can only have one default subnet per Availability Zone. You cannot create a default subnet in a nondefault VPC.

Create a VPC

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

Contents

- Create a VPC plus other VPC resources
- Create a VPC only
- Create a VPC using the AWS CLI

For information about modifying a VPC, see the section called “Add or remove CIDR block”.

Create a VPC plus other VPC resources

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

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

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the VPC dashboard, choose **Create VPC**.
3. For **Resources to create**, choose **VPC and more**.
4. Keep **Name tag auto-generation** selected to create Name tags for the VPC resources or clear it to provide your own Name tags for the VPC resources.
5. For **IPv4 CIDR block**, enter an IPv4 address range for the VPC. A VPC must have an IPv4 address range.

6. (Optional) To support IPv6 traffic, choose **IPv6 CIDR block**, **Amazon-provided IPv6 CIDR block**.
7. Choose a **Tenancy** option. This option defines if EC2 instances that you launch into the VPC will run on hardware that's shared with other AWS accounts or on hardware that's dedicated for your use only. If you choose the tenancy of the VPC to be `Default`, EC2 instances launched into this VPC will use the tenancy attribute specified when you launch the instance. For more information, see Launch an instance using defined parameters in the *Amazon EC2 User Guide*. If you choose the tenancy of the VPC to be `Dedicated`, the instances will always run as Dedicated Instances on hardware that's dedicated for your use. If you're using AWS Outposts, your Outpost requires private connectivity; you must use `Default` tenancy.
8. For **Number of Availability Zones (AZs)**, we recommend that you provision subnets in at least two Availability Zones for a production environment. To choose the AZs for your subnets, expand **Customize AZs**. Otherwise, let AWS choose them for you.
9. To configure your subnets, choose values for **Number of public subnets** and **Number of private subnets**. To choose the IP address ranges for your subnets, expand **Customize subnets CIDR blocks**. Otherwise, let AWS choose them for you.
10. (Optional) If resources in a private subnet need access to the public internet over IPv4, for **NAT gateways**, choose the number of AZs in which to create NAT gateways. In production, we recommend that you deploy a NAT gateway in each AZ with resources that need access to the public internet. Note that there is a cost associated with NAT gateways. For more information, see Pricing for NAT gateways.
11. (Optional) If resources in a private subnet need access to the public internet over IPv6, for **Egress only internet gateway**, choose **Yes**.
12. (Optional) If you need to access Amazon S3 directly from your VPC, choose **VPC endpoints**, **S3 Gateway**. This creates a gateway VPC endpoint for Amazon S3. For more information, see Gateway endpoints in the *AWS PrivateLink Guide*.
13. (Optional) For **DNS options**, both options for domain name resolution are enabled by default. If the default doesn't meet your needs, you can disable these options.
14. (Optional) To add a tag to your VPC, expand **Additional tags**, choose **Add new tag**, and enter a tag key and a tag value.
15. In the **Preview** pane, you can visualize the relationships between the VPC resources that you've configured. Solid lines represent relationships between resources. Dotted lines represent network traffic to NAT gateways, internet gateways, and gateway endpoints. After you create the VPC, you can visualize the resources in your VPC in this format at any time using the **Resource map** tab. For more information, see Visualize the resources in your VPC.

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

Create a VPC only

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

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

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the VPC dashboard, choose **Create VPC**.
3. For **Resources to create**, choose **VPC only**.
4. (Optional) For **Name tag**, enter a name for your VPC. Doing so creates a tag with a key of `Name` and the value that you specify.
5. For **IPv4 CIDR block**, do one of the following:
   - Choose **IPv4 CIDR manual input** and enter an IPv4 address range for your VPC.
   - Choose **IPAM-allocated IPv4 CIDR block**, select your Amazon VPC IP Address Manager (IPAM) IPv4 address pool and a netmask. The size of the CIDR block is limited by the allocation rules on the IPAM pool. IPAM is a VPC feature that makes it easier for you to plan, track, and monitor IP addresses for your AWS workloads. For more information, see the Amazon VPC IPAM User Guide.

     If you are using IPAM to manage your IP addresses, we recommend that you choose this option. Otherwise, the CIDR block that you specify for your VPC might overlap with an IPAM CIDR allocation.
6. (Optional) To create a dual stack VPC, specify an IPv6 address range for your VPC. For **IPv6 CIDR block**, do one of the following:

   - Choose **IPAM-allocated IPv6 CIDR block** if you are using Amazon VPC IP Address Manager and you want to provision a IPv6 CIDR from an IPAM pool. If you use the IPAM-allocated IPv6 CIDR block to provision IPv6 CIDRs to VPCs, you get the benefit of contiguous IPv6 CIDRs for VPC creation. Contiguously-allocated CIDRs are CIDRs that are allocated sequentially. They enable you to simplify your security and networking rules; the IPv6 CIDRs can be aggregated in a single entry across networking and security constructs like access control lists, route tables, security groups, and firewalls.

     You have two options for provisioning an IP address range to the VPC under **CIDR block**:

  - **Netmask length**: Choose this option to select a netmask length for the CIDR. Do one of the following:
    - If there is a default netmask length selected for the IPAM pool, you can choose **Default to IPAM netmask length** to use the default netmask length set for the IPAM pool by the IPAM administrator. For more information about the optional default netmask length allocation rule, see Create a Regional IPv6 pool in the *Amazon VPC IPAM User Guide*.
    - If there is no default netmask length selected for the IPAM pool, choose a netmask length that's more specific than the netmask length of the IPAM pool CIDR. For example, if the IPAM pool CIDR is /50, you can choose a netmask length between **/52** to **/60** for the VPC. Possible netmask lengths are between **/44** and **/60** in increments of /4.
  - **Select a CIDR**: Choose this option to manually enter an IPv6 address. You can only choose a netmask length that's more specific than the netmask length of the IPAM pool CIDR. For example, if the IPAM pool CIDR is /50, you can choose a netmask length between **/52** to **/60** for the VPC. Possible IPv6 netmask lengths are between **/44** and **/60** in increments of /4.
- Choose **Amazon-provided IPv6 CIDR block** to request an IPv6 CIDR block from an Amazon pool of IPv6 addresses. For **Network Border Group**, select the group from which AWS advertises IP addresses. Amazon provides a fixed IPv6 CIDR block size of **/56**.
- Choose **IPv6 CIDR owned by me** to provision an IPv6 CIDR that you have already brought to AWS. For more information about bringing your own IP address ranges to AWS, see Bring your own IP addresses (BYOIP) in the *Amazon EC2 User Guide*. You can provision an IP address range for the VPC using the following options for **CIDR block**:
  - **No preference**: Choose this option to use netmask length of **/56**.
  - **Select a CIDR**: Choose this option to manually enter an IPv6 address and choose a netmask length that's more specific than the size of BYOIP CIDR. For example, if the BYOIP pool CIDR is /50, you can choose a netmask length between **/52** to **/60** for the VPC. Possible IPv6 netmask lengths are between **/44** and **/60** in increments of /4.

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

you're using AWS Outposts, your Outpost requires private connectivity; you must use `Default` tenancy.

8. (Optional) To add a tag to your VPC, choose **Add new tag** and enter a tag key and a tag value.
9. Choose **Create VPC**.
10. After you create a VPC, you can add subnets. For more information, see Create a subnet.

Create a VPC using the AWS CLI

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

Prerequisites

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

Tagging

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

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

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

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

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

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

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

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

```
vpc-1a2b3c4d5e6f1a2b3
```

2. [Dual stack VPC] Get the IPv6 CIDR block that's associated with your VPC by using the following describe-vpcs command.

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

The following is example output.

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

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

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

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

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

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

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

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

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

```
subnet-1a2b3c4d5e6f1a2b3
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
aws ec2 create-nat-gateway --subnet-id subnet-id-public-subnet --allocation-
id eipalloc-id
```

c. (Optional) If you already created a route table for the private subnet in step 5, skip this step. Otherwise, use the following create-route-table command to create a route table for your private subnet. The command returns the ID of the new route table.

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

d. Create a route in the route table for the private subnet that sends all IPv4 traffic to the NAT gateway by using the following create-route command. Use the ID of the route table for the private subnet, which you created either in this step or in step 5.

```
aws ec2 create-route --route-table-id rtb-id-private --destination-cidr-
block 0.0.0.0/0 --gateway-id nat-id
```

e. (Optional) If you already associated a route table with the private subnet in step 5, skip this step. Otherwise, use the following associate-route-table command to associate the route table with the private subnet. Use the ID of the route table for the private subnet, which you created either in this step or in step 5.

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

Visualize the resources in your VPC

This section describes how you can see a visual representation of the resources in your VPC using the **Resource map** tab. The following resources are visible in the resource map:

- VPC
- Subnets
  - The Availability Zone is represented with a letter.
  - Public subnets are green.

  - Private subnets are blue.

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

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

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

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

To visualize the resources in your VPC

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

   - **VPC**: The IPv4 and IPv6 CIDR ranges assigned to the VPC.
   - **Subnets**: The IPv4 and IPv6 CIDR ranges assigned to each subnet.
   - **Route tables**: The subnet associations, and the number of routes in the route table.
   - **Network connections**: The details related to each type of connection:

- If there are public subnets in the VPC, there is an internet gateway resource with the number of routes and the source and destination subnets for traffic using the internet gateway.
- If there is an egress-only internet gateway, there is an egress-only internet gateway resource with the number of routes and the source and destination subnets for traffic using the egress-only internet gateway.
- If there is a NAT gateway, there is a NAT gateway resource with the number of network interfaces and Elastic IP addresses for the NAT gateway.
- If there is a gateway endpoint, there is a gateway endpoint resource with the name of the AWS service (Amazon S3 or Amazon DynamoDB) that you can connect to using the endpoint.

6. Hover over a resource to see the relationship between the resources. Solid lines represent relationships between resources. Dotted lines represent network traffic to network connections.

Add or remove a CIDR block from your VPC

This section describes how to add or remove IPv4 and IPv6 CIDR blocks from a VPC.

> **Important**
>
> - Your VPC can have up to five IPv4 and five IPv6 CIDR blocks by default, but this limit is adjustable. For more information, see [Amazon VPC quotas](). For information about restrictions on CIDR blocks for a VPC, see [VPC CIDR blocks]().
> - If your VPC has more than one IPv4 CIDR block associated with it, you can remove an IPv4 CIDR block from the VPC. You cannot remove the primary IPv4 CIDR block. You must remove an entire CIDR block; you cannot remove a subset of a CIDR block or a merged range of CIDR blocks. You must first delete all subnets in the CIDR block.
> - If you no longer want IPv6 support in your VPC, but you want to continue using your VPC to create and communicate with IPv4 resources, you can remove the IPv6 CIDR block.
> - To remove an IPv6 CIDR block, you must first unassign any IPv6 addresses that are assigned to any instances in your subnet.

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

To add or 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 then choose **Actions**, **Edit CIDRs**.
4. To remove a CIDR, choose **Remove** next to the CIDR.
5. To add a CIDR, choose **Add new IPv4 CIDR** or **Add new IPv6 CIDR**.
6. To add a CIDR for **IPv4 CIDR block**, do one of the following:
   - Choose **IPv4 CIDR manual input** and enter an IPv4 CIDR block.
   - Choose **IPAM-allocated IPv4 CIDR** and select a CIDR from an IPv4 IPAM pool.
   - Choose **Save**.
7. To add a CIDR for **IPv6 CIDR block**, do the following:
   - Choose **IPAM-allocated IPv6 CIDR block** if you are using Amazon VPC IP Address Manager and you want to provision a IPv6 CIDR from an IPAM pool. You have two options for provisioning an IP address range to the VPC under **CIDR block**:
     - **Netmask length**: Choose this option to select a netmask length for the CIDR. Do one of the following:
       - If there is a default netmask length selected for the IPAM pool, you can choose **Default to IPAM netmask length** to use the default netmask length set for the IPAM pool by the IPAM administrator. For more information about the optional default netmask length allocation rule, see Create a Regional IPv6 pool in the *Amazon VPC IPAM User Guide*.
       - If there is no default netmask length selected for the IPAM pool, choose a netmask length that's more specific than the netmask length of the IPAM pool CIDR. For example, if the IPAM pool CIDR is /50, you can choose a netmask length between **/52** to **/60** for the VPC. Possible netmask lengths are between **/44** and **/60** in increments of /4.
     - **Select a CIDR**: Choose this option to manually enter an IPv6 address. You can only choose a netmask length that's more specific than the netmask length of the IPAM pool CIDR. For

example, if the IPAM pool CIDR is /50, you can choose a netmask length between **/52** to **/60** for the VPC. Possible IPv6 netmask lengths are between **/44** and **/60** in increments of /4.

- Choose **Amazon-provided IPv6 CIDR block** to request an IPv6 CIDR block from an Amazon pool of IPv6 addresses. For **Network Border Group**, select the group from which AWS advertises IP addresses. Amazon provides a fixed IPv6 CIDR block size of **/56**.
- Choose **IPv6 CIDR owned by me** to provision an IPv6 CIDR that you have already brought to AWS. For more information, see Bring your own IP addresses (BYOIP) to Amazon EC2 in the *Amazon EC2 User Guide*. You have two options for provisioning an IP address range to the VPC under **CIDR block**:
  - **No preference**: Choose this option to use netmask length of **/56**.
  - **Select a CIDR**: Choose this option to manually enter an IPv6 address and choose a netmask length that's more specific than the size of BYOIP CIDR. For example, if the BYOIP pool CIDR is /50, you can choose a netmask length between **/52** to **/60** for the VPC. Possible IPv6 netmask lengths are between **/44** and **/60** in increments of /4.
  - Choose **Select CIDR** when you're done.

8. Choose **Close**.
9. If you've added a CIDR block to your VPC, you can create subnets that use the new CIDR block. For more information, see Create a subnet.

To associate or disassociate a CIDR block from a VPC using the AWS CLI

Use the associate-vpc-cidr-block and disassociate-vpc-cidr-block commands.

DHCP option sets in Amazon VPC

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

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

Contents

- What is DHCP?

- DHCP option set concepts
- Work with DHCP option sets

What is DHCP?

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

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

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

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

- An EC2 instance in a dual-stack subnet can only retrieve its IPv6 address from the IPv6 DHCP server. *It cannot retrieve any additional network configurations from the IPv6 DHCP server, such as DNS server names or domain names.*
- An EC2 instance in a IPv6-only subnet can retrieve its IPv6 address from the IPv6 DHCP server *and can retrieve additional networking configuration information, such as DNS server names and domain names.*
- For an EC2 instance in an IPv6-only subnet, the IPv4 DHCP Server will return 169.254.169.253 as the name server if "AmazonProvidedDNS" is explicitly mentioned in the DHCP option set. If "AmazonProvidedDNS" is missing from the option set, the IPv4 DHCP Server won't return an address whether other IPv4 name servers are mentioned in the option set or not.

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

DHCP option set concepts

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

Each Region has a default DHCP option set. Each VPC uses the default DHCP option set for its Region unless you either create and associate a custom DHCP option set with the VPC or configure the VPC with no DHCP option set.

If your VPC has no DHCP option set configured:

- For EC2 instances built on the Nitro System, AWS configures `169.254.169.253` as the default domain name server.
- For EC2 instances built on Xen, no domain name servers are configured and, because instances in the VPC have no access to a DNS server, they can't access the internet.

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

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

Contents

- Default DHCP option set
- Custom DHCP option set

Default DHCP option set

The default DHCP option set contains the following settings:

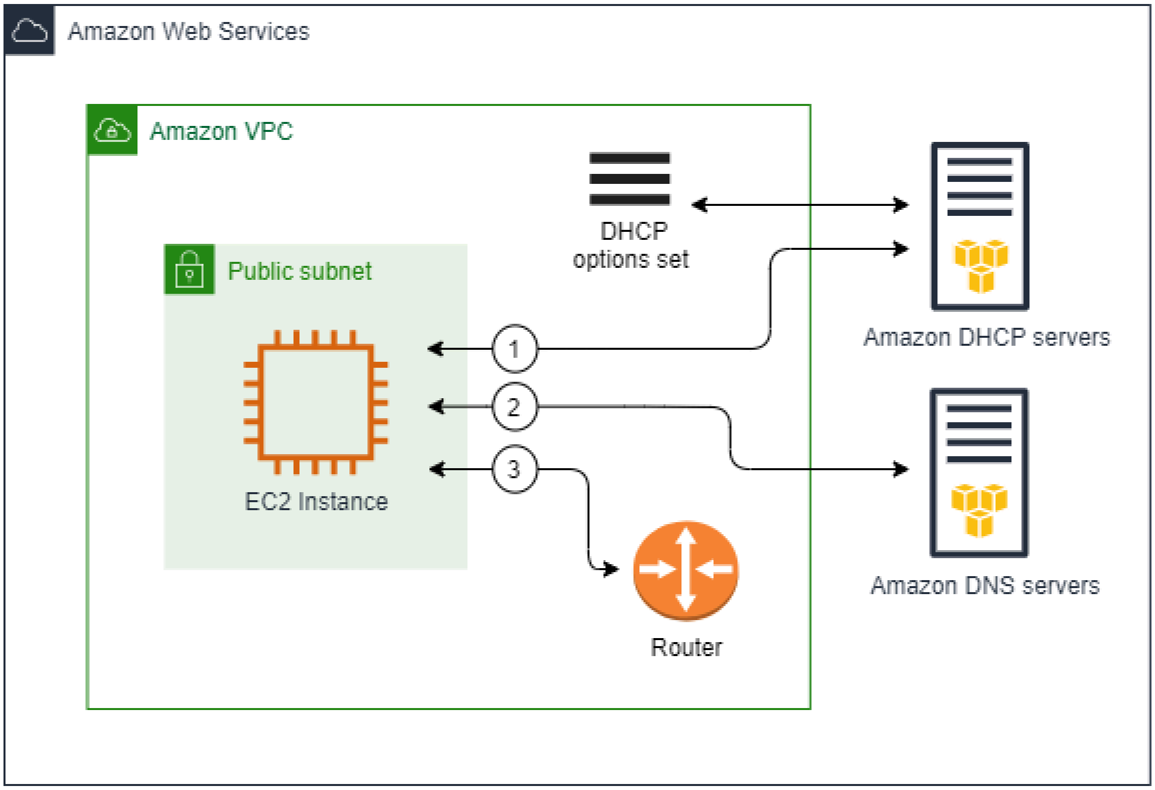
- **Domain name servers**: The DNS servers that your network interfaces use for domain name resolution. For a default DHCP option set, this is always `AmazonProvidedDNS`. For more information, see Amazon DNS server.
- **Domain name**: The domain name that a client should use when resolving hostnames using the Domain Name System (DNS). For more information about the domain names used for EC2 instances, see Amazon EC2 instance hostnames.

- **IPv6 Preferred Lease Time**: How frequently a running instance with an IPv6 assigned to it goes through DHCPv6 lease renewal. The default lease time is 140 seconds. Lease renewal typically occurs when half of the lease time has elapsed.

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

- **NTP servers**: By default, EC2 instances use the Amazon Time Sync Service to retrieve the time.
- **NetBIOS name servers**: For EC2 instances running Windows, the NetBIOS computer name is a friendly name assigned to the instance to identify it on the network. The NetBIOS name server maintains a list of mappings between NetBIOS computer names and network addresses for networks that use NetBIOS as their naming service.
- **NetBIOS node type**: For EC2 instances running Windows, this is the method that the instances use to resolve NetBIOS names to IP addresses.

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



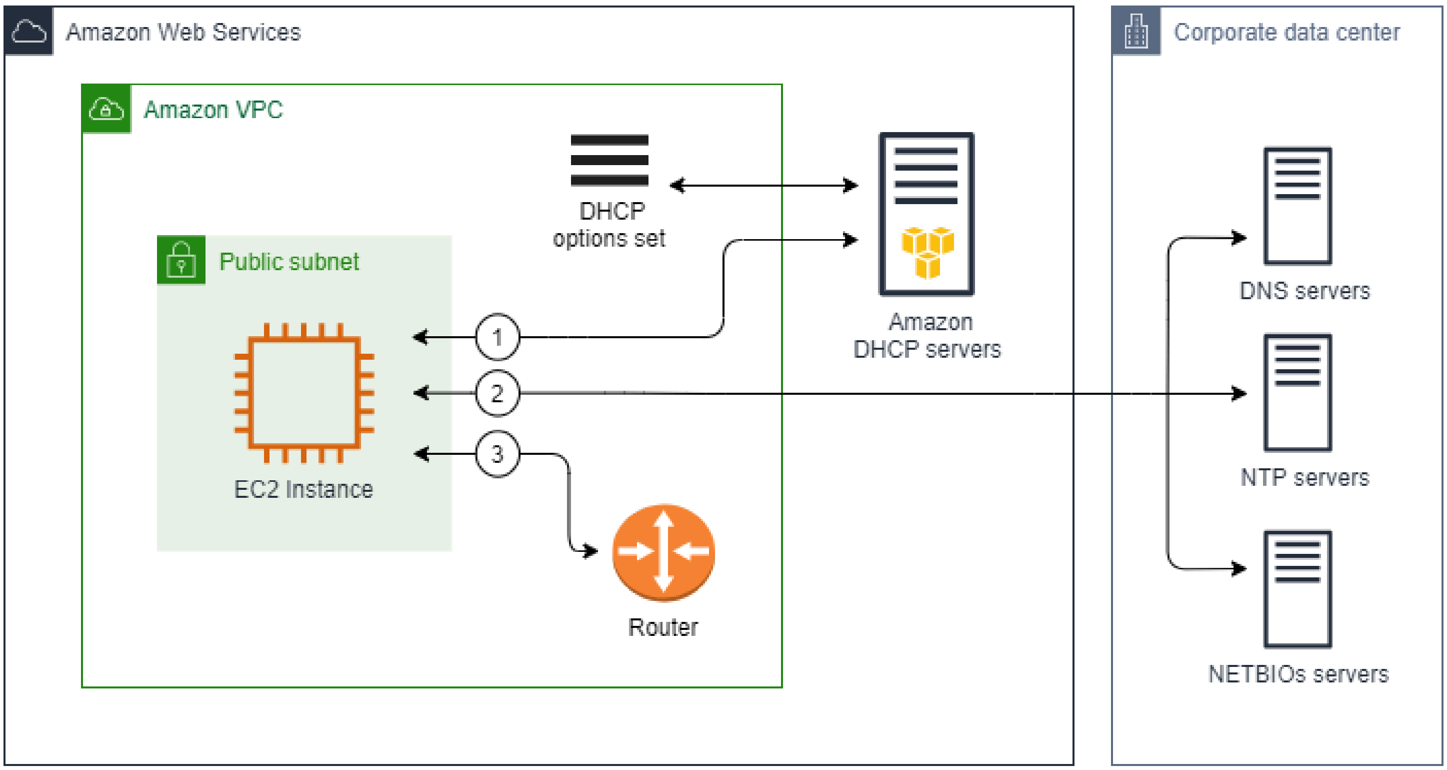
Custom DHCP option set

You can create a custom DHCP option set with the following settings, and then associate it with a VPC:

- **Domain name servers**: The DNS servers that your network interfaces use for domain name resolution.
- **Domain name**: The domain name that a client uses when resolving hostnames using the Domain Name System (DNS).
- **NTP servers**: The NTP servers that provide the time to the instances.
- **NetBIOS name servers**: For EC2 instances running Windows, the NetBIOS computer name is a friendly name assigned to the instance to identify it on the network. A NetBIOS name server maintains a list of mappings between NetBIOS computer names and network addresses for networks that use NetBIOS as their naming service.
- **NetBIOS node type**: For EC2 instances running Windows, the method that the instances use to resolve NetBIOS names to IP addresses.
- **IPv6 Preferred Lease Time** (optional): A value (in seconds, minutes, hours, or years) for how frequently a running instance with an IPv6 assigned to it goes through DHCPv6 lease renewal. Acceptable values are between 140 and 4294967295 seconds (approximately 138 years). If no

value is entered, the default lease time is 140 seconds. If you use long-term addressing for EC2 instances, you can increase the lease time and avoid frequent lease renewal requests. Lease renewal typically occurs when half of the lease time has elapsed.

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



Related tasks

- Create a DHCP option set
- Change the option set associated with a VPC

Work with DHCP option sets

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

Tasks

- Create a DHCP option set
- Change the option set associated with a VPC
- Delete a DHCP option set

Create a DHCP option set

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

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

To create a DHCP options set using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **DHCP option sets**.
3. Choose **Create DHCP options set**.
4. For **Tag settings**, optionally enter a name for the DHCP option set. If you enter a value, it automatically creates a Name tag for the DHCP option set.
5. For **DHCP options**, provide the configuration settings that you need.
   - **Domain name** (optional): Enter the domain name that a client should use when resolving hostnames via the Domain Name System. If you are not using AmazonProvidedDNS, your custom domain name servers must resolve the hostname as appropriate. If you use an Amazon Route 53 private hosted zone, you can use AmazonProvidedDNS. For more information, see DNS attributes for your VPC.

     > **Note**
     >
     > Only use domain names that you fully control.

Some Linux operating systems accept multiple domain names separated by spaces. However, Windows and other Linux operating systems treat the value as a single domain, which results in unexpected behavior. If your DHCP option set is associated with a VPC that has instances running operating systems that treat the value as a single domain, specify only one domain name.

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

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

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

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

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

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

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

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

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

  For EC2 instances running a Windows OS, this is the method that the instances use to resolve NetBIOS names to IP addresses. In the default options set, there is no value for NetBIOS node type.
- **IPv6 Preferred Lease Time** (optional): A value (in seconds, minutes, hours, or years) for how frequently a running instance with an IPv6 assigned to it goes through DHCPv6 lease renewal. Acceptable values are between 140 and 2147483647 seconds (approximately 68 years). If no value is entered, the default lease time is 140 seconds. If you use long-term addressing for EC2 instances, you can increase the lease time and avoid frequent lease renewal requests. Lease renewal typically occurs when half of the lease time has elapsed.

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

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

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

Change the option set associated with a VPC

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

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

To change the DHCP option set associated with a VPC using the console

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

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

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

Delete a DHCP option set

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

To delete a DHCP option set using the console

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

To delete a DHCP option set using the command line

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

DNS attributes for your VPC

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

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

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

Contents

- Understanding Amazon DNS
- View DNS hostnames for your EC2 instance
- View and update DNS attributes for your VPC

Understanding Amazon DNS

As an AWS architect or administrator, one of the foundational networking components you'll encounter is the Amazon DNS server, also known as the Route 53 Resolver. This DNS resolver service is natively integrated into each Availability Zone within your AWS Region, providing a reliable and scalable solution for domain name resolution within your Virtual Private Cloud (VPC). In this section you'll learn about the Amazon DNS server's IP addresses, the private DNS hostnames it can resolve, and the rules that govern its usage.

Contents

- Amazon DNS server

- Rules and considerations
- DNS hostnames for EC2 instances
- DNS attributes for your VPC
- DNS quotas
- Private hosted zones

Amazon DNS server

The Route 53 Resolver (also called "Amazon DNS server" or "AmazonProvidedDNS") is a DNS Resolver service which is built into each Availability Zone in an AWS Region. The Route 53 Resolver is located at `169.254.169.253` (IPv4), `fd00:ec2::253` (IPv6), and at the primary private IPV4 CIDR range provisioned to your VPC plus two. For example, if you have a VPC with an IPv4 CIDR of `10.0.0.0/16` and an IPv6 CIDR of `2001:db8::/32`, you can reach the Route 53 Resolver at `169.254.169.253` (IPv4), `fd00:ec2::253` (IPv6), or `10.0.0.2` (IPv4). Resources within a VPC use a link local address for DNS queries. These queries are transported to the Route 53 Resolver privately and are not visible on the network. In an IPv6-only subnet, the IPv4 link-local address (169.254.169.253) is still reachable as long as "AmazonProvidedDNS" is the name server in the DHCP option set.

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

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

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*.
- The Amazon Route 53 Resolver only supports recursive DNS queries.

DNS hostnames for EC2 instances

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

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

Private IPv4 DNS name

The private IPv4 DNS hostname of an instance resolves to its private IPv4 address. You can use the private IPv4 DNS hostname for communication between instances in the same VPC or in connected VPCs. For more information, see Private IPv4 addresses in the *Amazon EC2 User Guide*.

Public IPv4 DNS name

The public IPv4 DNS hostname of an instance resolves to its public IPv4 address (outside the network of the instance) or its private IPv4 address (inside the network of the instance). For more information, see Public IPv4 addresses in the *Amazon EC2 User Guide*.

To resolve public IPv4 DNS names to private IPv4 addresses over a VPC peering connection, you must enable DNS resolution for the peering connection. For more information, see Enable DNS resolution for a VPC peering connection.

Private resource DNS name

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

DNS attributes for your VPC

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

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

Attribute	Description
`enableDnsHostnames`	Determines whether the VPC supports assigning public DNS hostnames to instances with public IP addresses. The default for this attribute is `false` unless the VPC is a default VPC. Note the **Rules and considerations** for this attribute below.
`enableDnsSupport`	Determines whether the VPC supports DNS resolution through the Amazon provided DNS server. If this attribute is `true`, queries to the Amazon provided DNS server succeed. For more information, see Amazon DNS server. The default for this attribute is `true`. Note the **Rules and considerations** for this attribute below.

Rules and considerations

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

  - Instances receive custom private DNS hostnames if there is a custom domain name in the DHCP options set. 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 Support.

DNS quotas

There is a 1024 packet per second (PPS) limit to services that use link-local addresses. This limit includes the aggregate of Route 53 Resolver DNS queries, Instance Metadata Service (IMDS) requests, Amazon Time Service Network Time Protocol (NTP) requests, and Windows Licensing Service (for Microsoft Windows based instances) requests. This quota cannot be increased.

The number of DNS queries per second supported by Route 53 Resolver varies by the type of query, the size of the response, and the protocol in use. For more information and recommendations for a scalable DNS architecture, see the AWS Hybrid DNS with Active Directory Technical Guide.

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

Private hosted zones

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

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

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

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

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. Knowing these hostnames is important for connecting to your resources.

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 for your VPC"](#).

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

- 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

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

View and update DNS attributes for your VPC

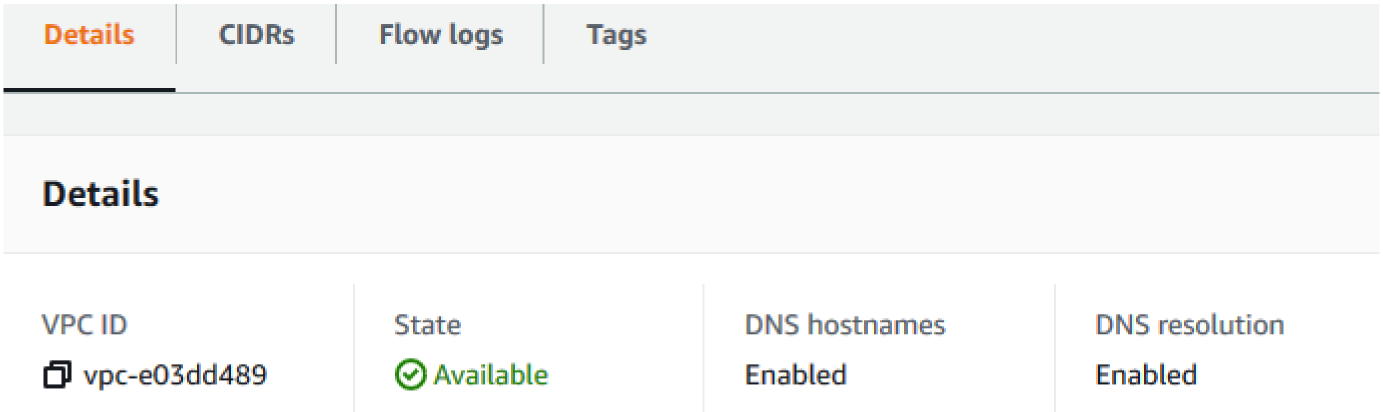
You can view and update the DNS support attributes for your VPC using the Amazon VPC console. These settings control whether your instances get public DNS hostnames and whether the Amazon DNS server can resolve your private DNS names. Configuring these attributes correctly is vital for ensuring seamless communication within your VPC.

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 **Edit VPC settings**. Select or clear **Enable** on the appropriate DNS attribute and choose **Save changes**.

To describe DNS support for a VPC using the command line

- describe-vpc-attribute (AWS CLI)
- Get-EC2VpcAttribute (AWS Tools for Windows PowerShell)

To update DNS support for a VPC using the command line

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

Network Address Usage for your VPC

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

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

- Network Address Usage – The maximum number of NAU units that a single VPC can have. Each VPC can have up to 64,000 NAU units by default. You can request a quota increase up to 256,000.
- Peered Network Address Usage – The maximum number of NAU units for a VPC and all of its peered VPCs. If a VPC is peered with other VPCs in the same Region, the VPCs combined can have up to 128,000 NAU units by default. You can request a quota increase up to 512,000. VPCs that are peered across different Regions do not contribute to this limit.

You can use the NAU in the following ways:

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

How NAU is calculated

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

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

Resource	NAU units
Each private or public IPv4 and each IPv6 address assigned to a network interface for an EC2 instance in the VPC	1
Additional network interfaces attached to an EC2 instance	1
Prefix assigned to a network interface	1
Network Load Balancer per AZ	6
Gateway Load Balancer per AZ	6

Resource	NAU units
VPC endpoint per AZ	6
Transit gateway attachment	6
Lambda function	6
NAT gateway	6
EFS mount target	6
EFA interface (EFA with an ENA device) or an EFA-only interface	1
Amazon EKS pod	1

NAU examples

The following examples show how to calculate NAU.

Example 1 - Two VPCs connected using VPC peering

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

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

Example 2 - Two VPCs connected using a transit gateway

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

- VPC 1
  - 50 Network Load Balancers in 2 subnets in separate Availability Zones - 600 NAU units
  - 5,000 instances (each with an IPv4 address and IPv6 address) in one subnet and 5,000 instances (each with an IPv4 address and IPv6 address) in another subnet - 20,000 units
  - 100 Lambda functions - 600 NAU units
- VPC 2
  - 50 Network Load Balancers in 2 subnets in separate Availability Zones - 600 NAU units
  - 5,000 instances (each with an IPv4 address and IPv6 address) in one subnet and 5,000 instances (each with an IPv4 address and IPv6 address) in another subnet - 20,000 units
  - 100 Lambda functions - 600 NAU units
- Total NAU count per VPC: 21,200 units
- Default NAU quota per VPC: 64,000 units

Share your VPC subnets with other accounts

VPC subnet 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 VPC subnets 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 VPC subnets using connectivity features, such as AWS PrivateLink, transit gateways, and VPC peering. For more information about the benefits of VPC subnet sharing, see VPC sharing: A new approach to multiple accounts and VPC management.

There are quotas related to VPC subnet sharing. For more information, see VPC subnet sharing.

Contents

- Shared subnet prerequisites
- Working with shared subnets
- Billing and metering for owner and participants
- Responsibilities and permissions for owners and participants
- AWS resources and shared VPC subnets

Shared subnet prerequisites

This section contains prerequisites for working with shared subnets:

- The accounts for the VPC owner and participant must be managed by AWS Organizations.
- You must enable resource sharing in the AWS RAM console from the management account for your organization. For more information, see Enable resource sharing within AWS Organizations in the *AWS RAM User Guide*.
- You must create a resource share. You can specify the subnets to share when you create the resource share, or add the subnets to the resource share later on using the procedure in the next section. For more information, see Create a resource share in the *AWS RAM User Guide*.

Working with shared subnets

This section describes how to work with shared subnets in the AWS console and AWS CLI.

Contents

- Share a subnet
- Unshare a shared subnet
- Identify the owner of a shared subnet

Share a subnet

You can share non-default subnets with other accounts within your organization as follows. In addition, you can share security groups across AWS Organizations. For more information, see Share security groups with AWS Organizations.

To share a subnet using the console

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

To share a subnet using the AWS CLI

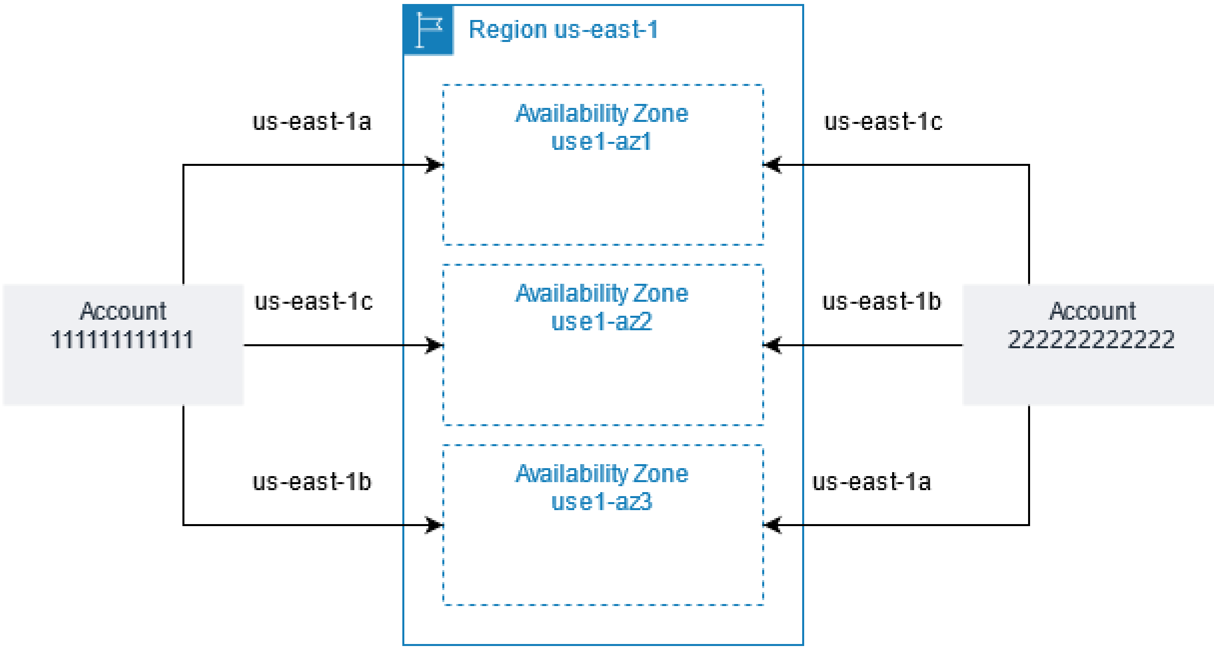
Use the create-resource-share and associate-resource-share commands.

Map subnets across Availability Zones

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

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

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



Unshare a shared subnet

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

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

To unshare a subnet using the console

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

To unshare a subnet using the AWS CLI

Use the disassociate-resource-share command.

Identify the owner of a shared subnet

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

To identify a subnet owner using the console

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

To identify a subnet owner using the AWS CLI

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

Billing and metering for owner and participants

This section contains billing and metering details for those who own the shared subnet and for those working with the shared subnet:

- 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 as well as data transfer over VPC peering connections, across internet gateways, and across AWS Direct Connect gateways.
- 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. In addition, public IPv4 addresses used in shared VPCs are billed to VPC owners. For more information about public IPv4 address pricing, see the **Public IPv4 Address** tab on the Amazon VPC pricing page.
- Data transfer within the same Availability Zone (uniquely identified using the AZ-ID) is free irrespective of account ownership of the communicating resources.

Responsibilities and permissions for owners and participants

This section includes details about the responsibilities and permissions for those who own the shared subnet (owner) and for those who are using the shared subnet (participant).

Owner resources

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

Participant resources

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

VPC resources

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

Flow logs

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

Internet gateways and egress-only internet gateways

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

NAT gateways

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

Network access control lists (NACLs)

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

Network interfaces

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

Route tables

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

Security groups

- Participants can work with (create, delete, describe, modify, or create ingress and egress rules for) security groups that they own in a shared VPC subnet. Participants can work with security groups created by VPC owners if the VPC owner shares the security group with the participant.

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

Subnets

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

Transit gateways

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

VPCs

- Participants cannot modify VPCs or their related attributes. Only the VPC owner can. Participants can describe VPCs, their attibutes, and the DHCP option sets.
- VPC tags and tags for the resources within the shared VPC are not shared with the participants.
- Participants can associate their own security groups with a shared VPC. This allows the participant to use the security group with Elastic network interfaces they own in the shared VPC.

AWS resources and shared VPC subnets

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

- Amazon Aurora
- AWS Database Migration Service
- Amazon EC2
- Amazon ECS
- Amazon ElastiCache (Redis OSS)
- Amazon EFS
- Amazon Elastic Kubernetes Service
- Elastic Load Balancing
  - Application Load Balancers
  - Gateway Load Balancers
  - Network Load Balancers
- Amazon EMR
- AWS Glue
- AWS Lambda
- Amazon MQ running Apache MQ (not Rabbit MQ)
- Amazon MSK
- AWS Network Manager
  - AWS Cloud WAN
  - Network Access Analyzer
  - Reachability Analyzer
- Amazon OpenSearch Service
- AWS PrivateLink[†]
- Amazon Relational Database Service (RDS)
- Amazon Redshift
- Amazon Route 53
- AWS Transit Gateway
- AWS Verified Access
- Amazon VPC
  - Peering
  - Traffic Mirroring

- Amazon VPC Lattice

† You can connect to all AWS services that support PrivateLink using a VPC endpoint in a shared VPC. For a list of services that support PrivateLink, see AWS services that integrate with AWS PrivateLink in the *AWS PrivateLink Guide*.

This list is intended to include all services that support launching resources in shared VPC subnets. Despite our best efforts, there might be services that support launching resources in shared VPC subnets but are not listed here. We encourage you to submit documentation feedback if you have questions.

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

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

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

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

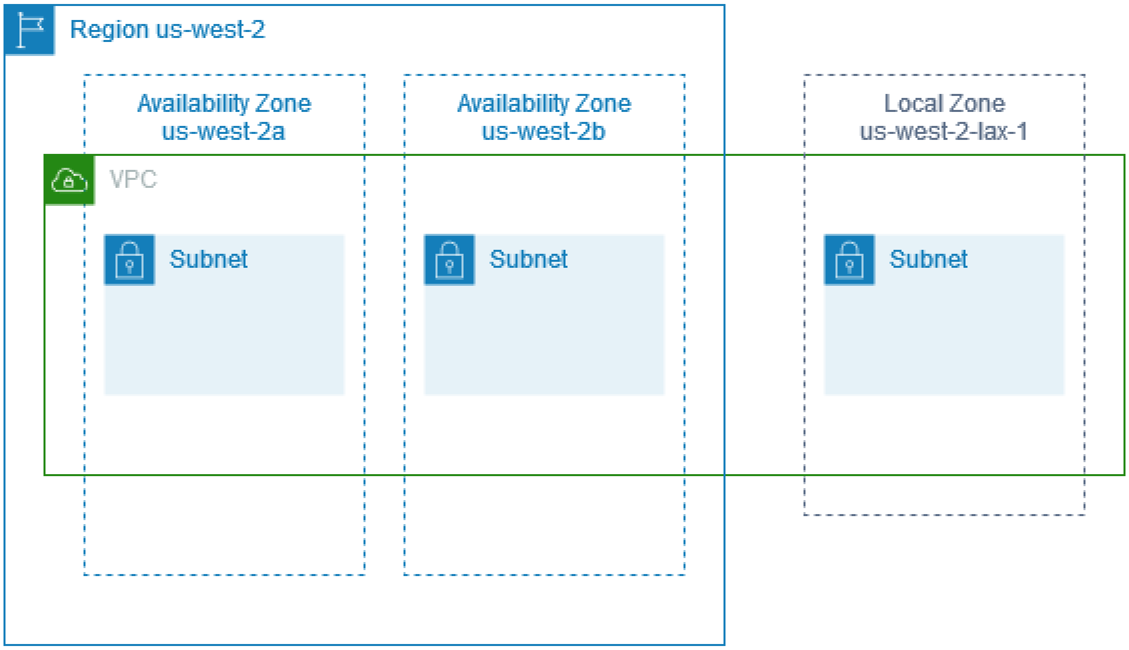
Subnets in AWS Local Zones

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

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

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

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



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

The following rules apply to Local Zones:

- The Local Zone subnets follow the same routing rules as Availability Zone subnets, including route tables, security groups, and network ACLs.
- Outbound internet traffic leaves a Local Zone from the Local Zone.
- You must provision public IP addresses for use in a Local Zone. When you allocate addresses, you can specify the location from which the IP address is advertised. We refer to this as a network border group, and you can set this parameter to limit the addresses to this location. After you

provision the IP addresses, you cannot move them between the Local Zone and the parent Region (for example, from `us-west-2-lax-1a` to `us-west-2`).

- If the Local Zone supports IPv6, you can request IPv6 Amazon-provided IP addresses and associate them with the network border group for a new or existing VPC. For the list of Local Zones that support IPv6, see Considerations in the *AWS Local Zones User Guide*
- You can't create VPC endpoints in Local Zone subnets.

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

Considerations for internet gateways

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

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

  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”.
- 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”.

Access Local Zones using a Direct Connect gateway

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

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

You configure the following resources for this configuration:

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

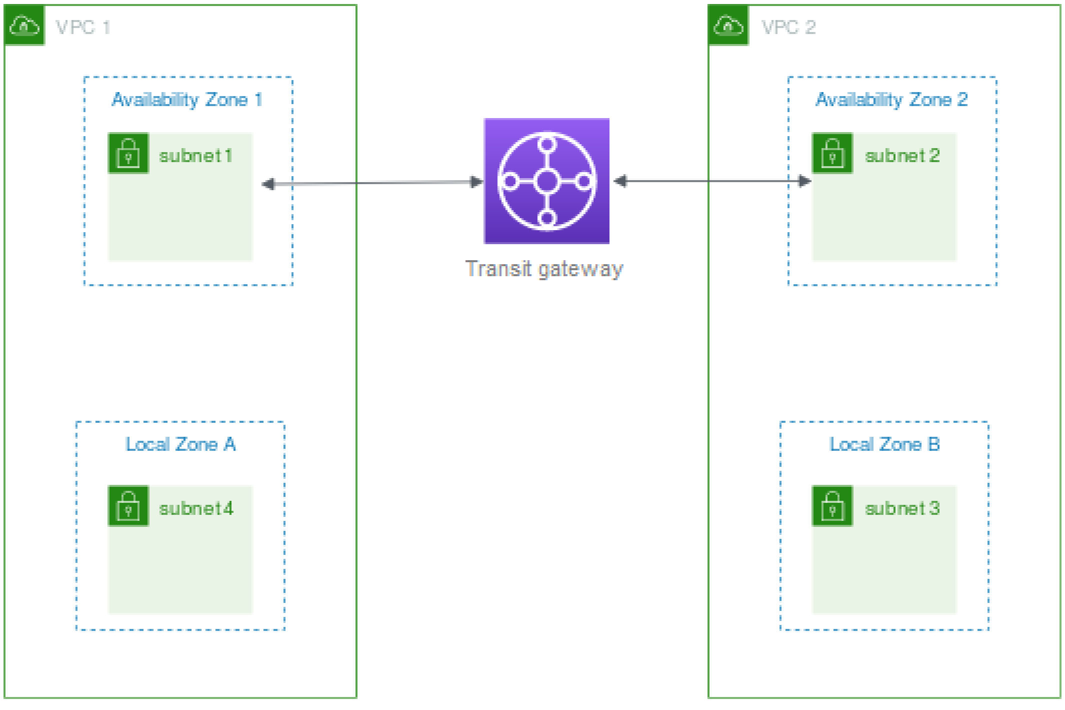
  For information about creating a virtual private gateway, see Create a target gateway in the *AWS Site-to-Site VPN User Guide*.
- A Direct Connect connection. For the best latency performance, AWS recommends that you use the Direct Connect location closest to the Local Zone to which you'll be extending your subnet.

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

Connect Local Zone subnets to a transit gateway

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

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



Create the following resources for this scenario:

- A subnet in each parent Availability Zone. For more information, see the section called "Create a subnet".
- 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 subnet route tables of the Local Zone subnets that have 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".

The following is an example route table for VPC 1.

Destination	Target

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

Subnets in AWS Wavelength

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

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

The following rules apply to Wavelength Zones:

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

Considerations for multiple Wavelength Zones

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

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

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

- For each Wavelength Zone, a subnet in an Availability Zone that is the parent Availability Zone for the Wavelength Zone. In the example, you create subnet 1 and subnet 2. For information about creating subnets, see the section called "Create a subnet". Use the describe-availability-zones command 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".

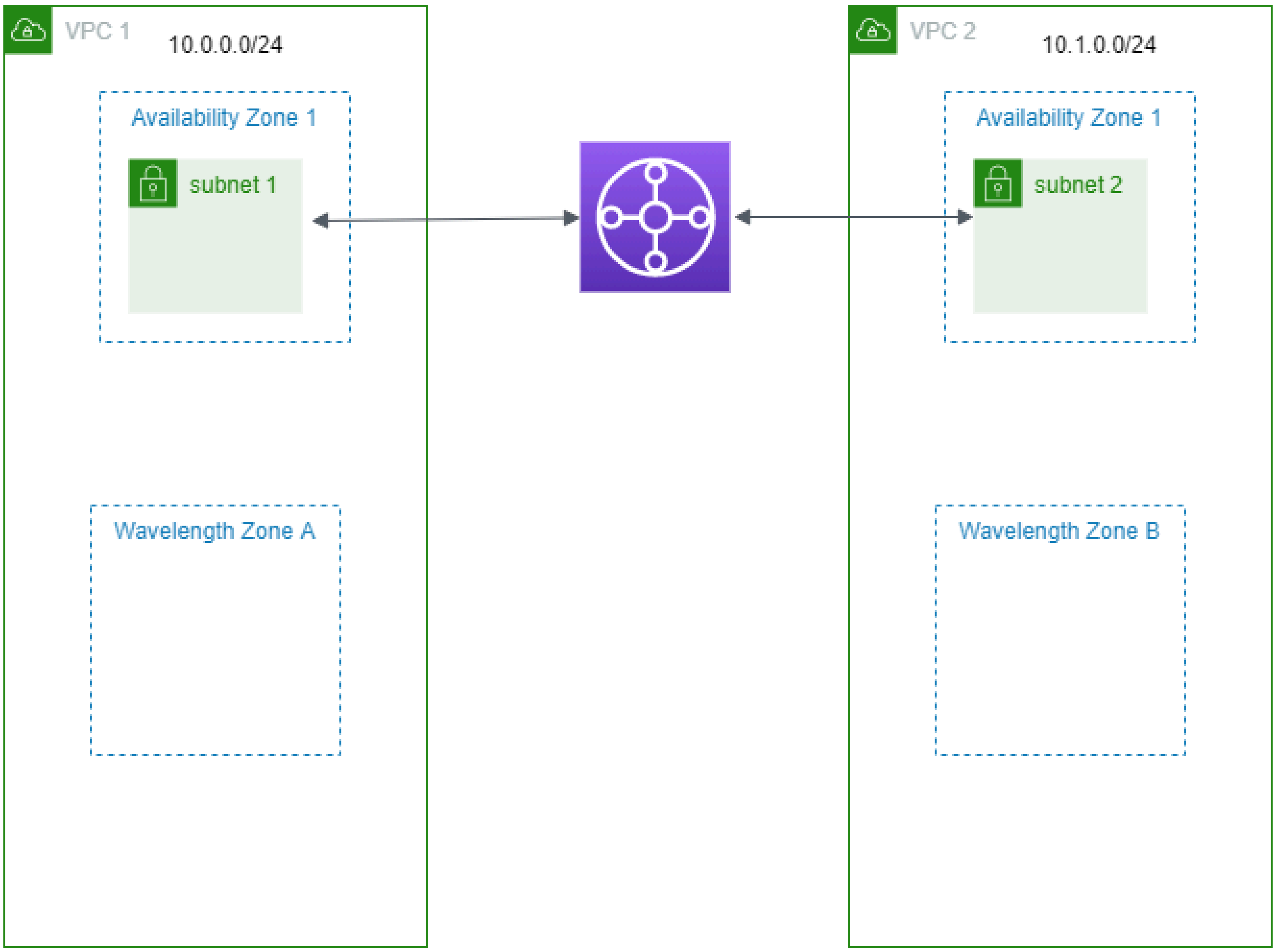
  In the example, the route table for VPC 1 has the following entry:

Destination	Target
10.1.0.0/24	tgw-22222222222222222

  The route table for VPC 2 has the following entry:

Destination	Target
	tgw-22222222222222222

Destination	Target
10.0.0.0/24	



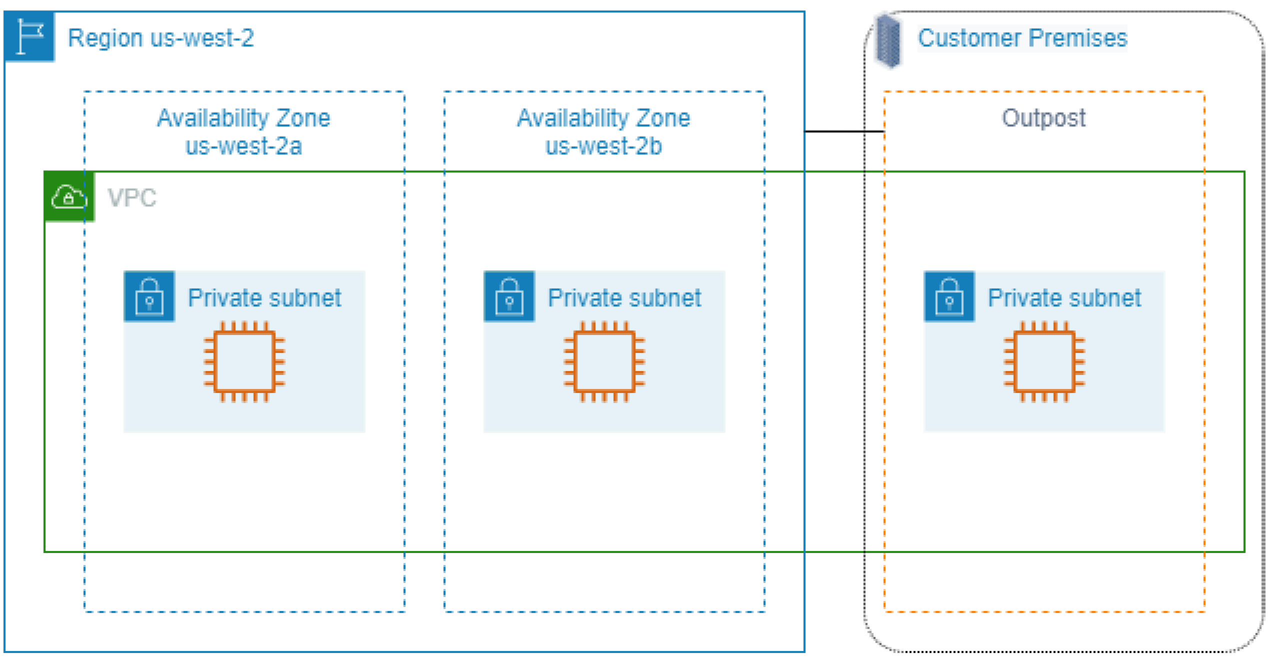
Subnets in AWS Outposts

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

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

The following rules apply to AWS Outposts:

- The subnets must reside in one Outpost location.
- You create a subnet for an Outpost by specifying the Amazon Resource Name (ARN) of the Outpost when you create the subnet.
- Outposts rack - A local gateway handles the network connectivity between your VPC and on-premises networks. For more information, see Local gateways in the *AWS Outposts User Guide for Outposts rack*.
- Outposts servers - A local network interface handles the network connectivity between your VPC and on-premises networks. For more information, see Local network interfaces in the *AWS Outposts User Guide for Outposts servers*.
- By default, every subnet that you create in a VPC, including subnets for your Outposts, is implicitly associated with the main route table for the VPC. Alternatively, you can 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 destined for your on-premises network.



Delete your VPC

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

Requirement

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

> **Note**
>
> If you have created a flow log for the VPC you are deleting, note that flow logs for deleted VPCs are eventually automatically removed.

Contents

- Delete a VPC using the console
- Delete a VPC using the command line

Delete a VPC using the console

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

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

To delete your VPC using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Terminate all instances in the VPC. For more information, see Terminate Your Instance in the *Amazon EC2 User Guide*.

3. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
4. In the navigation pane, choose **Your VPCs**.
5. Select the VPC to delete and choose **Actions**, **Delete VPC**.
6. If there are resources that you must delete or terminate before we can delete the VPC, we display them. Delete or terminate these resources and then try again. Otherwise, we display the resources that we will delete in addition to the VPC. Review the list and then proceed to the next step.
7. (Optional) If you have a Site-to-Site VPN connection, you can select the option to delete it. If you plan to use the customer gateway with another VPC, we recommend that you keep the Site-to-Site VPN connection and the gateways. Otherwise, you must configure your customer gateway device again after you create a new Site-to-Site VPN connection.
8. When prompted for confirmation, enter **`delete`** and then choose **Delete**.

Delete a VPC using the command line

Before you can delete a VPC using the command line, you must terminate or delete any resources that created a requester-managed network interface in the VPC. You must also delete or detach all VPC resources that you created, such as subnets, security groups, network ACLs, route tables, internet gateways, and egress-only internet gateways. You do not need to delete the default security group, default route table, or default network ACL.

The following procedure demonstrates the commands that you use to delete common VPC resources and then to delete your VPC. You must use these commands in this order. If you created additional VPC resources, you'll also need to use their corresponding delete command before you can delete the VPC.

To delete a VPC by using the AWS CLI

1. Delete your security group by using the delete-security-group command.

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

2. Delete each network ACL by using the delete-network-acl command.

```
aws ec2 delete-network-acl --network-acl-id acl-id
```

3. Delete each subnet by using the delete-subnet command.

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

4. Delete each custom route table by using the [delete-route-table](#) command.

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

5. Detach your internet gateway from your VPC by using the [detach-internet-gateway](#) command.

```
aws ec2 detach-internet-gateway --internet-gateway-id igw-id --vpc-id vpc-id
```

6. Delete your internet gateway by using the [delete-internet-gateway](#) command.

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

7. [Dual stack VPC] Delete your egress-only internet gateway by using the [delete-egress-only-internet-gateway](#) command.

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

8. Delete your VPC by using the [delete-vpc](#) command.

```
aws ec2 delete-vpc --vpc-id vpc-id
```

Generate infrastructure-as-code from your VPC console actions with Console-to-Code

The console provides a guided path for creating resources and testing prototypes. If you want to create the same resources at scale, you'll need automation code. Console-to-Code is a feature of Amazon Q Developer that can help you get started with your automation code. Console-to-Code records your console actions, including default values and compatible parameters. It then uses generative AI to suggest code in your preferred infrastructure-as-code (IaC) format for the actions you want. Because the console workflow makes sure the parameter values that you specify are valid together, the code that you generate by using Console-to-Code has compatible parameter values. You can use the code as a starting point, and then customize it to make it production-ready for your specific use case.

For example, with Console-to-Code you can record yourself using the VPC console to create subnets, security groups, NACLs, a custom routing table, and an internet gateway and generate code in AWS CloudFormation JSON format. Then, you can copy that code and customize it for use in your AWS CloudFormation template.

Console-to-Code can currently generate infrastructure-as-code (IaC) in the following languages and formats:

- CDK Java
- CDK Python
- CDK TypeScript
- CloudFormation JSON
- CloudFormation YAML

For more information and instructions on how to use Console-to-Code, see Automating AWS services with Amazon Q Developer Console-to-Code in the *Amazon Q Developer User Guide*.

Subnets for your VPC

A *subnet* is a range of IP addresses in your VPC. You can create AWS resources, such as EC2 instances, in specific subnets.

Contents

- Subnet basics
- Subnet security
- Create a subnet
- Add or remove an IPv6 CIDR block from your subnet
- Modify the IP addressing attributes of your subnet
- Subnet CIDR reservations
- Configure route tables
- Middlebox routing wizard
- Delete a subnet

Subnet basics

Each subnet must reside entirely within one Availability Zone and cannot span zones. By launching AWS resources in separate Availability Zones, you can protect your applications from the failure of a single Availability Zone.

Contents

- Subnet IP address range
- Subnet types
- Subnet diagram
- Subnet routing
- Subnet settings

Subnet IP address range

When you create a subnet, you specify its IP addresses, depending on the configuration of the VPC:

- **IPv4 only** – The subnet has an IPv4 CIDR block but does not have an IPv6 CIDR block. Resources in an IPv4-only subnet must communicate over IPv4.
- **Dual stack** – The subnet has both an IPv4 CIDR block and an IPv6 CIDR block. The VPC must have both an IPv4 CIDR block and an IPv6 CIDR block. Resources in a dual-stack subnet can communicate over IPv4 and IPv6.
- **IPv6 only** – The subnet has an IPv6 CIDR block but does not have an IPv4 CIDR block. The VPC must have an IPv6 CIDR block. Resources in an IPv6-only subnet must communicate over IPv6.

> **Note**
>
> Resources in IPv6-only subnets are assigned IPv4 link-local addresses from CIDR block 169.254.0.0/16. These addresses are used to communicate with services that are available only in the VPC. For examples, see Link-local addresses in the *Amazon EC2 User Guide*.

For more information, see IP addressing for your VPCs and subnets.

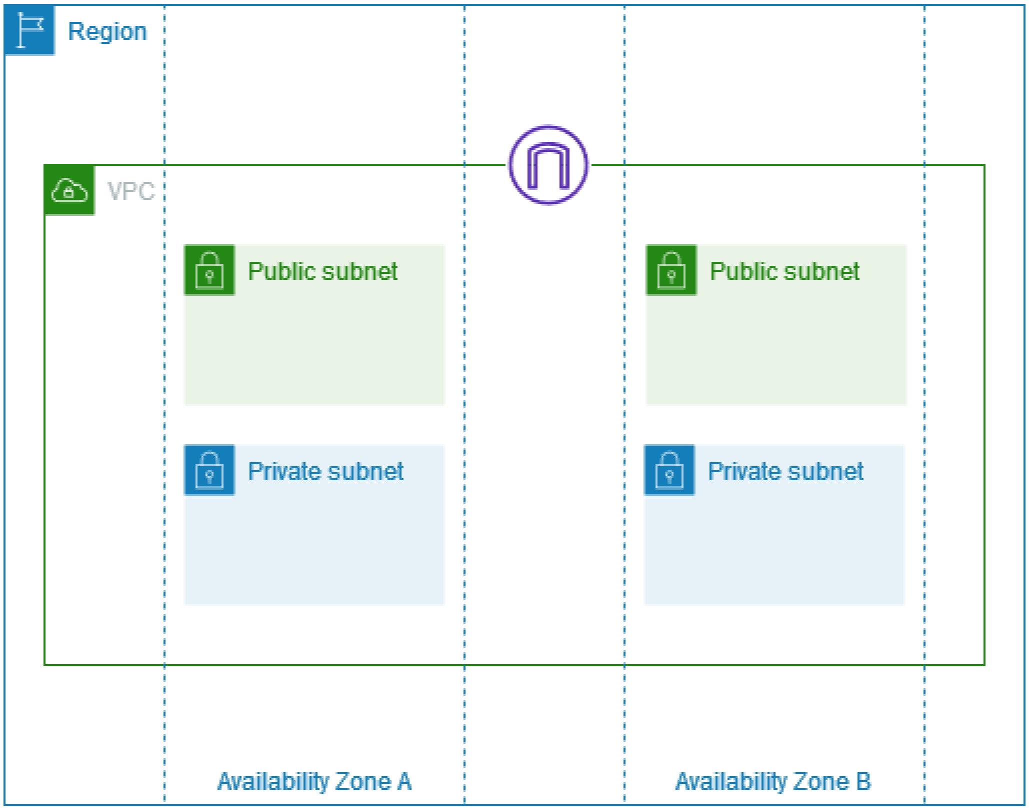
Subnet types

The subnet type is determined by how you configure routing for your subnets. For example:

- **Public subnet** – The subnet has a direct route to an internet gateway. Resources in a public subnet can access the public internet.
- **Private subnet** – The subnet does not have a direct route to an internet gateway. Resources in a private subnet require a NAT device to access the public internet.
- **VPN-only subnet** – The subnet has a route to a Site-to-Site VPN connection through a virtual private gateway. The subnet does not have a route to an internet gateway.
- **Isolated subnet** – The subnet has no routes to destinations outside its VPC. Resources in an isolated subnet can only access or be accessed by other resources in the same VPC.
- **EVS subnet** – This type of subnet is created using Amazon EVS. For more information, see VLAN subnet in the *Amazon EVS User Guide*.

Subnet diagram

The following diagram shows a VPC with subnets in two Availability Zones and an internet gateway. Each Availability Zone has a public subnet and a private subnet.



For diagrams that show subnets in Local Zones and Wavelength Zones, see How AWS Local Zones work and How AWS Wavelength works.

Subnet routing

Each subnet must be associated with a route table, which specifies the allowed routes for outbound traffic leaving the subnet. Every subnet that you create is automatically associated with the main route table for the VPC. You can change the association, and you can change the contents of the main route table. For more information, see Configure route tables.

Subnet settings

All subnets have a modifiable attribute that determines whether a network interface created in that subnet is assigned a public IPv4 address and, if applicable, an IPv6 address. This includes the primary network interface (for example, eth0) that's created for an instance when you launch an instance in that subnet. Regardless of the subnet attribute, you can still override this setting for a specific instance during launch.

After you create a subnet, you can modify the following settings for the subnet:

- **Auto-assign IP settings**: Enables you to configure the auto-assign IP settings to automatically request a public IPv4 or IPv6 address for a new network interface in this subnet.
- **Resource-based Name (RBN) settings**: Enables you to specify the hostname type for EC2 instances in this subnet and configure how DNS A and AAAA record queries are handled. For more information, see Amazon EC2 instance hostname types in the *Amazon EC2 User Guide*.

Subnet security

To protect your AWS resources, we recommend that you use private subnets. Use a bastion host or NAT device to provide internet access to resources, such as EC2 instances, in a private subnet.

AWS provides features that you can use to increase security for the resources in your VPC. *Security groups* allow inbound and outbound traffic for associated resources, such as EC2 instances. *Network ACLs* allow or deny inbound and outbound traffic at the subnet level. In most cases, security groups can meet your needs. However, you can use network ACLs if you want an additional layer of security. For more information, see the section called “Compare security groups and network ACLs”.

By design, each subnet must be associated with a network ACL. Every subnet that you create is automatically associated with the default network ACL for the VPC. The default network ACL allows all inbound and outbound traffic. You can update the default network ACL, or create custom network ACLs and associate them with your subnets. For more information, see Control subnet traffic with network access control lists.

You can create a flow log on your VPC or subnet to capture the traffic that flows to and from the network interfaces in your VPC or subnet. You can also create a flow log on an individual network interface. For more information, see Logging IP traffic using VPC Flow Logs.

Create a subnet

Use the following procedure to create subnets for your virtual private cloud (VPC). Depending on the connectivity that you need, you might also need to add gateways and route tables.

Considerations

- You must specify an IPv4 CIDR block for the subnet from the range of your VPC. You can optionally specify an IPv6 CIDR block for a subnet if there is an IPv6 CIDR block associated with the VPC. For more information, see IP addressing for your VPCs and subnets.
- If you create an IPv6-only subnet, be aware of the following. An EC2 instance launched in an IPv6-only subnet receives an IPv6 address but not an IPv4 address. Any instances that you launch into an IPv6-only subnet must be instances built on the Nitro System.
- To create the subnet in a Local Zone or a Wavelength Zone, you must enable the Zone. For more information, see Regions and Zones in the *Amazon EC2 User Guide*.

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**.
3. Choose **Create subnet**.
4. Under **VPC ID**, choose the VPC for the subnet.
5. (Optional) For **Subnet name**, enter a name for your subnet. Doing so creates a tag with a key of `Name` and the value that you specify.
6. For **Availability Zone**, you can choose a Zone for your subnet, or leave the default **No Preference** to let AWS choose one for you.
7. For **IPv4 CIDR block**, select **Manual input** to enter an IPv4 CIDR block for your subnet (for example, `10.0.1.0/24`) or select **No IPv4 CIDR**. If you are using Amazon VPC IP Address Manager (IPAM) to plan, track, and monitor IP addresses for your AWS workloads, when you create a subnet you have the option to allocate a CIDR block from IPAM (**IPAM-allocated**). For more information on planning VPC IP address space for subnet IP allocations, see Tutorial: Plan VPC IP address space for subnet IP allocations in the *Amazon VPC IPAM User Guide*.
8. For **IPv6 CIDR block**, select **Manual input** to choose the VPC's IPv6 CIDR that you want to create a subnet in. This option is available only if the VPC has an associated IPv6 CIDR block. If you are using Amazon VPC IP Address Manager (IPAM) to plan, track, and monitor IP addresses for your AWS workloads, when you create a subnet you have the option to allocate a CIDR block from IPAM (**IPAM-allocated**). For more information on planning VPC IP address space for subnet IP allocations, see Tutorial: Plan VPC IP address space for subnet IP allocations in the *Amazon VPC IPAM User Guide*.
9. Choose an **IPv6 VPC CIDR block**.

10. For **IPv6 subnet CIDR block**, choose a CIDR for the subnet that's equal to or more specific than the VPC CIDR. For example, if the VPC pool CIDR is /50, you can choose a netmask length between **/50** to **/64** for the subnet. Possible IPv6 netmask lengths are between **/44** and **/64** in increments of /4.
11. Choose **Create subnet**.

To add a subnet to your VPC using the AWS CLI

Use the create-subnet command.

Next steps

After you create a subnet, you can configure it as follows:

- Configure routing. You can then create a custom route table and route that send traffic to a gateway that's associated with the VPC, such as an internet gateway. For more information, see Configure route tables.
- Modify the IP addressing behavior. You can specify whether instances launched in the subnet receive a public IPv4 address, an IPv6 address, or both. For more information, see Modify the IP addressing attributes of your subnet.
- Modify the resource-based name (RBN) settings. For more information, see Amazon EC2 instance hostname types.
- Create or modify your network ACLs. For more information, see Control subnet traffic with network access control lists.
- Share the subnet with other accounts. For more information, see ???.

Add or remove an IPv6 CIDR block from 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.

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

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

To add or remove an IPv6 CIDR block to a subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Subnets**.
3. Select your subnet and choose **Actions**, **Edit IPv6 CIDRs**.
4. To add a CIDR, choose **Add IPv6 CIDR**, choose a **VPC CIDR block**, enter a **Subnet CIDR block**, and choose a netmask length that's equal to or more specific than the netmask length of the VPC CIDR. For example, if the VPC pool CIDR is /50, you can choose a netmask length between **/50** to **/64** for the subnet. Possible IPv6 netmask lengths are between **/44** and **/64** in increments of /4.
5. To remove a CIDR, find the IPv6 CIDR block and choose **Remove**.
6. Choose **Save**.

To associate an IPv6 CIDR block with a subnet using the AWS CLI

Use the associate-subnet-cidr-block command.

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

Use the disassociate-subnet-cidr-block command.

Modify the IP addressing attributes of 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.

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 IP addressing behavior

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Subnets**.
3. Select your subnet and choose **Actions**, **Edit subnet settings**.
4. The **Enable auto-assign public IPv4 address** check box, if selected, requests a public IPv4 address for all instances launched into the selected subnet. Select or clear the check box as required, and then choose **Save**.
5. The **Enable auto-assign IPv6 address** check box, if selected, requests an IPv6 address for all network interfaces created in the selected subnet. Select or clear the check box as required, and then choose **Save**.

To modify a subnet attribute using the AWS CLI

Use the modify-subnet-attribute command.

Subnet CIDR reservations

A *subnet CIDR reservation* is a range of IPv4 or IPv6 addresses that you set aside so that AWS won't assign them to your network interfaces. This enables you to reserve IPv4 or IPv6 CIDR blocks (also called "prefixes") for use with your network interfaces.

When you create a subnet CIDR reservation, you specify how you will use the reserved IP addresses. The following options are available:

- **Prefix** — Allows you to assign a prefix to a single network interface. For more information, see Assign prefixes to Amazon EC2 network interfaces in the *Amazon EC2 User Guide*.
- **Explicit** — Allows you to manually assign an individual IP address to a single network interface.

The following rules apply to subnet CIDR reservations:

- When you create a subnet CIDR reservation, the IP address range can include addresses that are already in use. Creating a subnet reservation does not unassign any IP addresses that are already in use.
- You can reserve multiple CIDR ranges per subnet. When you reserve multiple CIDR ranges within the same VPC, the CIDR ranges cannot overlap.
- When you reserve more than one range in a subnet for Prefix Delegation, and Prefix Delegation is configured for automatic assignment, we choose the IP addresses to assign to network interfaces at random.
- When you delete a subnet reservation, the unused IP addresses are available for AWS to assign to your network interfaces. Deleting a subnet reservation does not unassign any IP addresses that are in use.

For more information about Classless Inter-Domain Routing (CIDR) notation, see *IP addressing*.

Contents

- Work with subnet CIDR reservations using the console
- Work with subnet CIDR reservations using the AWS CLI

Work with subnet CIDR reservations using the console

You can create and manage subnet CIDR reservations as follows.

To edit subnet CIDR reservations

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Subnets**.
3. Select the subnet.
4. Choose the **CIDR reservations** tab to get information about any existing subnet CIDR reservations.
5. To add or remove subnet CIDR reservations, choose **Actions**, **Edit CIDR reservations** and then do the following:
    - To add an IPv4 CIDR reservation, choose **IPv4**, **Add IPv4 CIDR reservation**. Choose the reservation type, enter the CIDR range, and choose **Add**.
    - To add an IPv6 CIDR reservation, choose **IPv6**, **Add IPv6 CIDR reservation**. Choose the reservation type, enter the CIDR range, and choose **Add**.

- To remove a CIDR reservation, choose **Remove** for the subnet CIDR reservation.

Work with subnet CIDR reservations using the AWS CLI

You can use the AWS CLI to create and manage subnet CIDR reservations.

Tasks

- Create a subnet CIDR reservation
- View subnet CIDR reservations
- Delete a subnet CIDR reservation

Create a subnet CIDR reservation

You can use create-subnet-cidr-reservation to create a subnet CIDR reservation.

```
aws ec2 create-subnet-cidr-reservation --subnet-id subnet-03c51e2eEXAMPLE --
reservation-type prefix --cidr 2600:1f13:925:d240:3a1b::/80
```

The following is example output.

```
{
    "SubnetCidrReservation": {
        "SubnetCidrReservationId": "scr-044f977c4eEXAMPLE",
        "SubnetId": "subnet-03c51e2ef5EXAMPLE",
        "Cidr": "2600:1f13:925:d240:3a1b::/80",
        "ReservationType": "prefix",
        "OwnerId": "123456789012"
    }
}
```

View subnet CIDR reservations

You can use get-subnet-cidr-reservations to view the details of a subnet CIDR reservation.

```
aws ec2 get-subnet-cidr-reservations --subnet-id subnet-05eef9fb78EXAMPLE
```

Delete a subnet CIDR reservation

You can use delete-subnet-cidr-reservation to delete a subnet CIDR reservation.

```
aws ec2 delete-subnet-cidr-reservation --subnet-cidr-reservation-id scr-044f977c4eEXAMPLE
```

Configure route tables

A *route table* serves as the traffic controller for your virtual private cloud (VPC). Each route table contains a set of rules, called *routes*, that determine where network traffic from your subnet or gateway is directed. When you create a VPC, we also create the main route table for the VPC. You can create additional route tables for your VPC, so that you have more granular control over the network paths for your VPC.

You can use route tables to specify which networks your VPC can communicate with, such as other VPCs or on-premises networks. Each route specifies a destination (CIDR block or prefix list) and a target (such as an internet gateway, NAT gateway, VPC peering connection, or VPN connection). Traffic is routed to targets based on its destination IP address. Route tables enable you to create complex networking architectures that include public subnets, private subnets, VPN-only subnets, and isolated subnets.

Contents

- Route table concepts
- Subnet route tables
- Gateway route tables
- How route priority works
- Example routing options
- Create a route table for your VPC
- Manage subnet route tables
- Replace the main route table
- Control traffic entering your VPC using a gateway route table
- Replace or restore the target for a local route
- Advanced routing in your VPC
- Troubleshoot reachability issues in your VPC

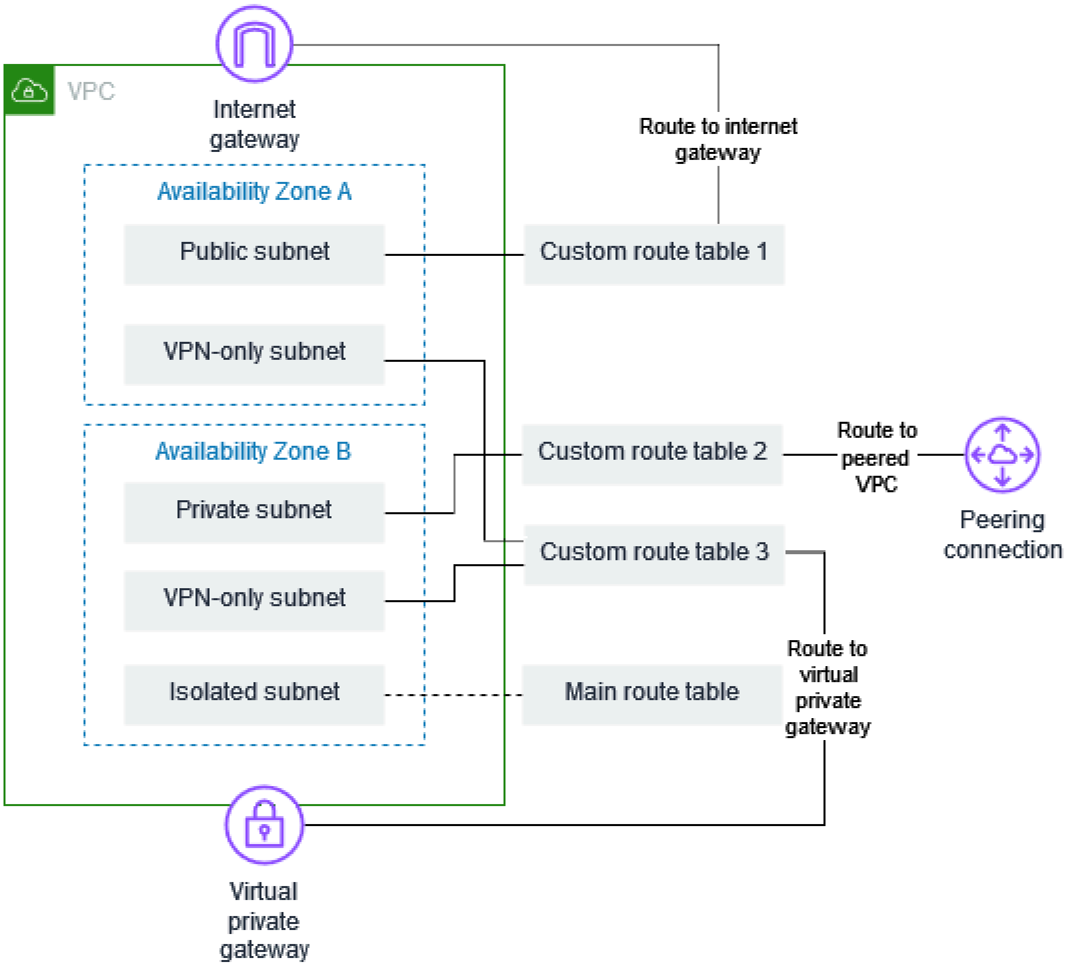
Route table concepts

The following are the key concepts for route tables:

- **Main route table**—The route table that automatically comes with your VPC. It controls the routing for all subnets that are not explicitly associated with any other route table.
- **Custom route table**—A route table that you create for your VPC.
- **Destination**—The range of IP addresses where you want traffic to go (destination CIDR). For example, an external corporate network with the CIDR `172.16.0.0/12`.
- **Target**—The gateway, network interface, or connection through which to send the destination traffic; for example, an internet gateway.
- **Local route**—A default route for communication within the VPC. If the VPC has both IPv4 and IPV6 addresses, there is a local route for IPv4 and a local route for IPv6.
- **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.
- **Propagation**—If you've attached a virtual private gateway to your VPC and enable route propagation, we automatically add routes for your VPN connection to your subnet route tables. This means that you don't need to manually add or remove VPN routes. For more information, see Site-to-Site VPN routing options in the *Site-to-Site VPN User Guide*.
- **Gateway route table**—A route table that's associated with an internet gateway or virtual private gateway.
- **Edge association**—A route table that you use to route inbound VPC traffic to an appliance. You associate a route table with the internet gateway or virtual private gateway, and specify the network interface of your appliance as the target for VPC traffic.
- **Transit gateway route table**—A route table that's associated with a transit gateway. For more information, see Transit gateway route tables in *Amazon VPC Transit Gateways*.
- **Local gateway route table**—A route table that's associated with an Outposts local gateway. For more information, see Local gateways in the *AWS Outposts User Guide*.

Example VPC with route tables

The following diagram shows a VPC with five subnets, a main route table, and three custom route tables. All four route tables have local routes. Custom route table 1 has a route to an internet gateway, and it is associated with the public subnet in Availability Zone A. Custom route table 2 has a route to a peered VPC, and it is associated with the private subnet in Availability Zone B. Custom route table 3 has a route to a virtual private gateway, and it is associated with the VPN-only subnets in both Availability Zones.



Subnet route tables

Your VPC has an implicit router, and you use route tables to control where network traffic is directed. Each subnet in your VPC must be associated with a route table, which controls the routing for the subnet (subnet route table). You can explicitly associate a subnet with a particular route table. Otherwise, the subnet is implicitly associated with the main route table. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same subnet route table.

Contents

- Routes
- Main route table

- Custom route tables
- Subnet route table association

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 to group them together. You can then specify the prefix list as the destination in your route table entry.

Every route table contains a local route for communication within the VPC. This route is added by default to all route tables. If your VPC has more than one IPv4 CIDR block, your route tables contain a local route for each IPv4 CIDR block. If you've associated an IPv6 CIDR block with your VPC, your route tables contain a local route for the IPv6 CIDR block. You can replace or restore the target of each local route as needed.

Rules and considerations

- You can add a route to your route tables that is more specific than the local route. The destination must match the entire IPv4 or IPv6 CIDR block of a subnet in your VPC. The target must be a NAT gateway, network interface, or Gateway Load Balancer endpoint.
- If your route table has multiple routes, we use the most specific route that matches the traffic (longest prefix match) to determine how to route the traffic.
- You can't add routes to IPv4 addresses that are an exact match or a subset of the following range: 169.254.168.0/22. This range is within the link-local address space and is reserved for

use by AWS services. For example, Amazon EC2 uses addresses in this range for services that are accessible only from EC2 instances, such as the Instance Metadata Service (IMDS) and the Amazon DNS server. You can use a CIDR block that is larger than but overlaps 169.254.168.0/22, but packets destined for addresses in 169.254.168.0/22 will not be forwarded.

- You can't add routes to IPv6 addresses that are an exact match or a subset of the following range: fd00:ec2::/32. This range is within the unique local address (ULA) space and is reserved for use by AWS services. For example, Amazon EC2 uses addresses in this range for services that are accessible only from EC2 instances, such as the Instance Metadata Service (IMDS) and the Amazon DNS server. You can use a CIDR block that is larger than but overlaps fd00:ec2::/32, but packets destined for addresses in fd00:ec2::/32 will not be forwarded.
- You can add middlebox appliances to the routing paths for your VPC. For more information, see the section called "Routing for a middlebox appliance".

Example

In the following example, suppose that the VPC has both an IPv4 CIDR block and an IPv6 CIDR block. IPv4 and IPv6 traffic are treated separately, as shown in the following route table.

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

- IPv4 traffic to be routed within the VPC (10.0.0.0/16) is covered by the Local route.
- IPv6 traffic to be routed within the VPC (2001:db8:1234:1a00::/56) is covered by the Local route.
- The route for 172.31.0.0/16 sends traffic to a peering connection.
- The route for all IPv4 traffic (0.0.0.0/0) sends traffic to an internet gateway. Therefore, all IPv4 traffic, except for traffic within the VPC and to the peering connection, is routed to the internet gateway.

- The route for all IPv6 traffic (::/0) sends traffic to an egress-only internet gateway. Therefore, all IPv6 traffic, except for traffic within the VPC, is routed to the egress-only internet gateway.

Main route table

When you create a VPC, it automatically has a main route table. When a subnet does not have an explicit routing table associated with it, the main routing table is used by default. On the **Route tables** page in the Amazon VPC console, you can view the main route table for a VPC by looking for **Yes** in the **Main** column.

By default, when you create a nondefault VPC, the main route table contains only a local route. If you Create a VPC and choose a NAT gateway, Amazon VPC automatically adds routes to the main route table for the gateways.

The following rules apply to the main route table:

- You can add, remove, and modify routes in the main route table.
- You can't delete the main route table.
- You can't set a gateway route table as the main route table.
- You can replace the main route table by associating a custom route table with a subnet.
- You can explicitly associate a subnet with the main route table, even if it's already implicitly associated.

  You might want to do that if you change which table is the main route table. When you change which table is the main route table, it also changes the default for additional new subnets, or for any subnets that are not explicitly associated with any other route table. For more information, see Replace the main route table.

Custom route tables

By default, a route table contains a local route for communication within the VPC. If you Create a VPC and choose a public subnet, Amazon VPC creates a custom route table and adds a route that points to the internet gateway. One way to protect your VPC is to leave the main route table in its original default state. Then, explicitly associate each new subnet that you create with one of the custom route tables you've created. This ensures that you explicitly control how each subnet routes traffic.

You can add, remove, and modify routes in a custom route table. You can delete a custom route table only if it has no associations.

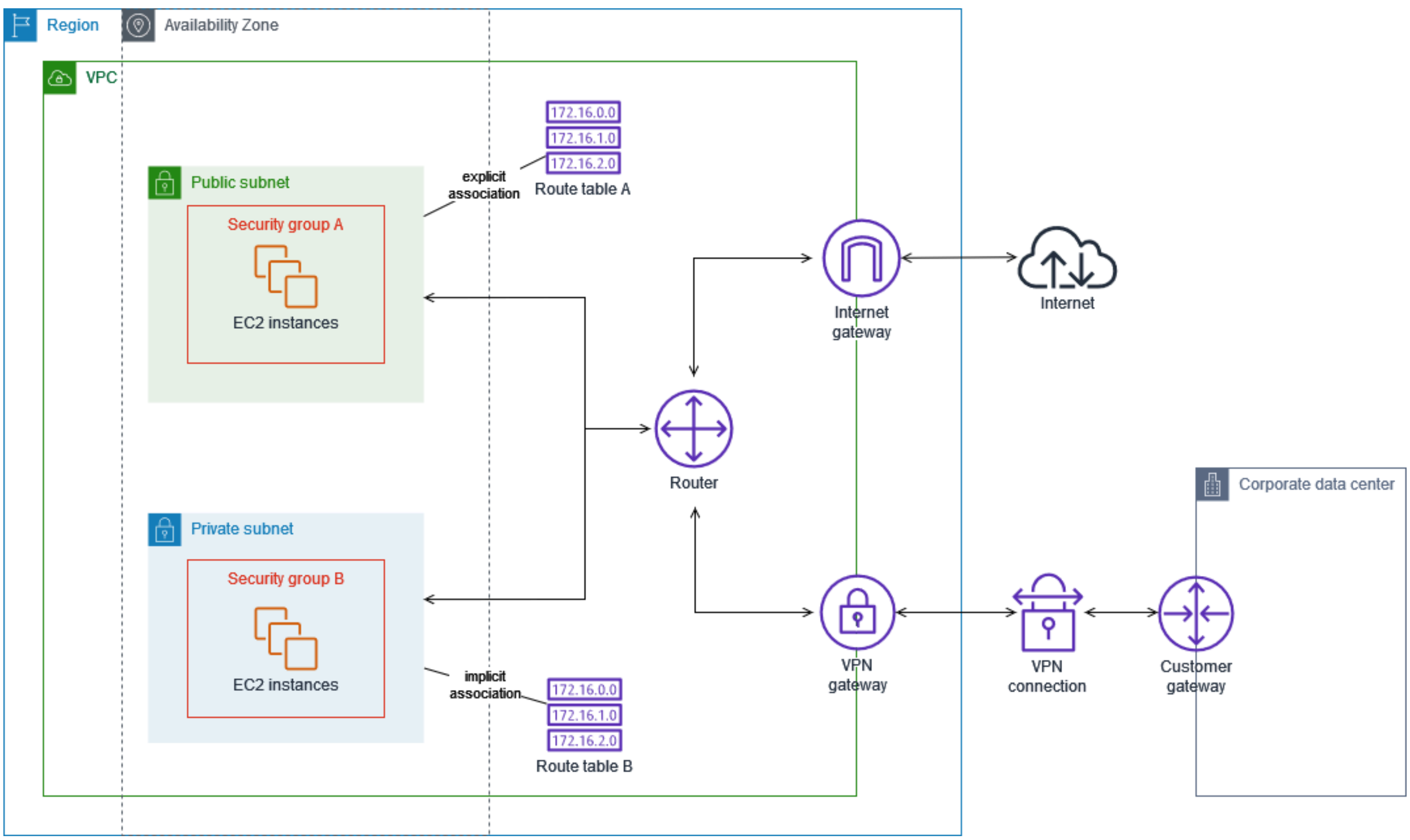
Subnet route table association

Each subnet in your VPC must be associated with a route table. A subnet can be explicitly associated with custom route table, or implicitly or explicitly associated with the main route table. For more information about viewing your subnet and route table associations, see [Determine the explicit associations]().

Subnets that are in VPCs associated with Outposts can have an additional target type of a local gateway. This is the only routing difference from non-Outposts subnets.

Example 1: Implicit and explicit subnet association

The following diagram shows the routing for a VPC with an internet gateway, a virtual private gateway, a public subnet, and a VPN-only subnet.



Route table A is a custom route table that is explicitly associated with the public subnet. It has a route that sends all traffic to the internet gateway, which is what makes the subnet a public subnet.

Destination	Target
VPC CIDR	Local
0.0.0.0/0	*igw-id*

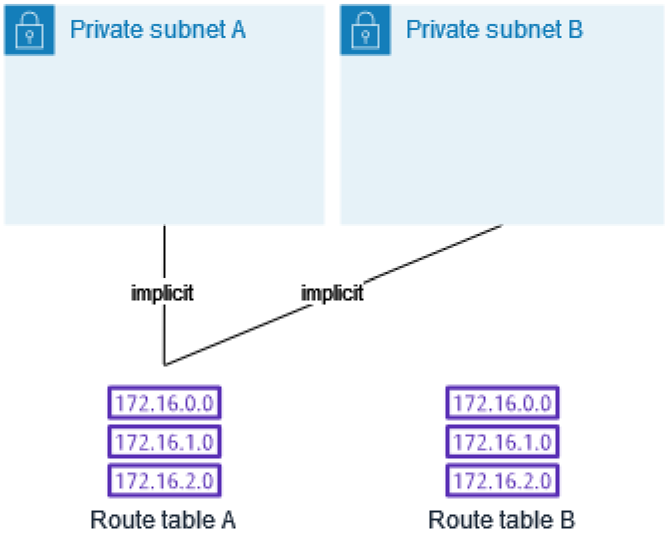
Route table B is the main route table. It is implicitly associated with the private subnet. It has a route that sends all traffic to the virtual private gateway, but no route to the internet gateway, which is what makes the subnet a VPN-only subnet. If you create another subnet in this VPC and don't associate a custom route table, the subnet will also be implicitly associated with this route table because it is the main route table.

Destination	Target
VPC CIDR	Local
0.0.0.0/0	*vgw-id*

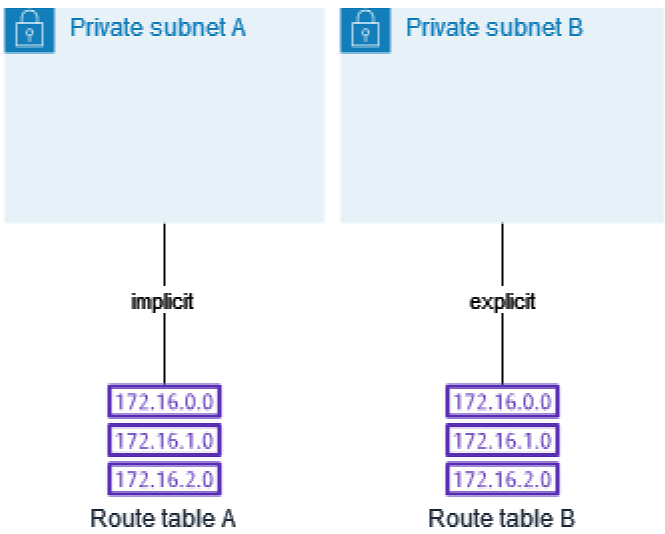
Example 2: Replacing the main route table

You might want to make changes to the main route table. To avoid any disruption to your traffic, we recommend that you first test the route changes using a custom route table. After you're satisfied with the testing, you can replace the main route table with the new custom table.

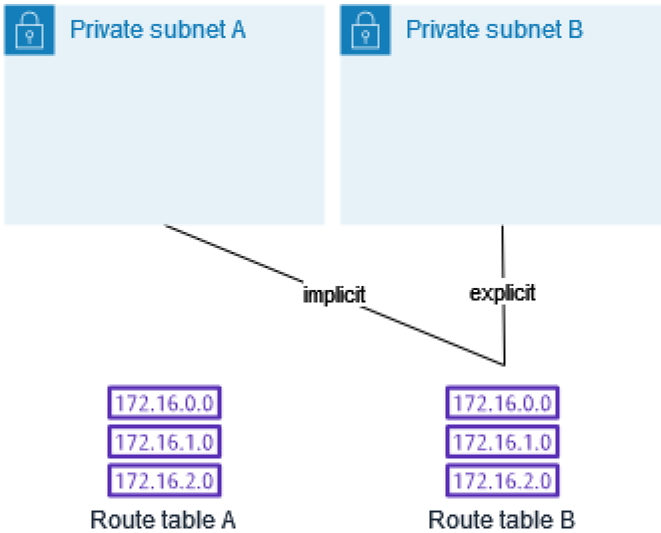
The following diagram shows two subnets and two route tables. Subnet A is implicitly associated with route table A, the main route table. Subnet B is implicitly associated with route table A. Route table B, a custom route table, isn't associated with either subnet.



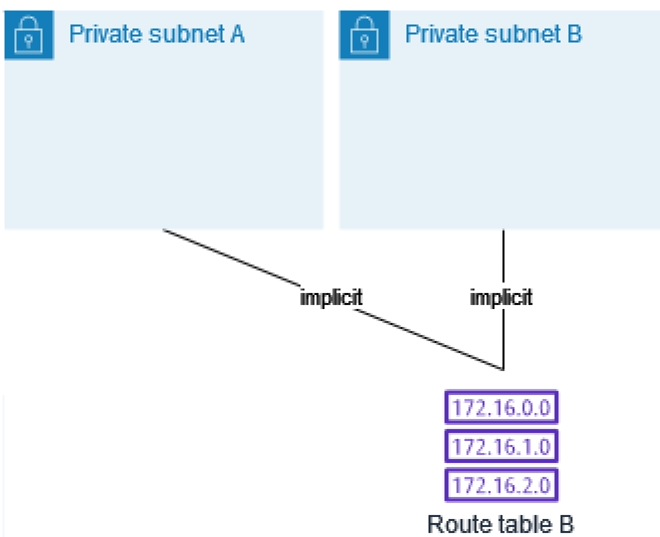
To replace the main route table, start by creating an explicit association between subnet B and route table B. Test route table B.



After you've tested route table B, make it the main route table. Subnet B still has an explicit association with route table B. However, subnet A now has an implicit association with route table B, because route table B is the new main route table. Route table A is no longer associated with either subnet.



(Optional) If you disassociate subnet B from route table B, there is still an implicit association between subnet B 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.

Contents

- Gateway route table routes
- Rules and considerations

Gateway route table routes

A gateway route table associated with an internet gateway supports routes with the following targets:

- The default local route
- A Gateway Load Balancer endpoint
- A network interface for a middlebox appliance

A gateway route table associated with a virtual private gateway supports routes with the following targets:

- The default local route
- A Gateway Load Balancer endpoint
- A network interface for a middlebox appliance

When the target is a Gateway Load Balancer endpoint or a network interface, the following destinations are allowed:

- The entire IPv4 or IPv6 CIDR block of your VPC. In this case, you replace the target of the default local route.
- The entire IPv4 or IPv6 CIDR block of a subnet in your VPC. This is a more specific route than the default local route.

If you change the target of the local route in a gateway route table to a network interface in your VPC, you can later restore it to the default `local` target. For more information, see Replace or restore the target for a local route.

Example

In the following gateway route table, traffic destined for a subnet with the `172.31.0.0/20` CIDR block is routed to a specific network interface. Traffic destined for all other subnets in the VPC uses the local route.

Destination	Target
172.31.0.0/16	Local
172.31.0.0/20	*eni-id*

Example

In the following gateway route table, the target for the local route is replaced with a network interface ID. Traffic destined for all subnets within the VPC is routed to the network interface.

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".
- 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.
- 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.
- Not all Local Zones support edge association with virtual private gateways. For more information on available zones, see Considerations in the *AWS Local Zones User Guide*.

How route priority works

In general, we direct traffic using the most specific route that matches the traffic. This is known as the longest prefix match. If your route table has overlapping or matching routes, additional rules apply.

The following list shows a route priority summary with links to sections below with more detailed information and examples:

1. Longest prefix (for example, 10.10.2.15/32 has priority over 10.10.2.0/24)
2. Static routes (like VPC peering and internet gateway connections)
3. Prefix list routes
4. Propagated routes
   a. Direct Connect BGP routes (dynamic routes)
   b. VPN static routes
   c. VPN BGP routes (dynamic routes) (like virtual private gateways)

Longest prefix match

Routes to IPv4 and IPv6 addresses or CIDR blocks are independent of each other. We use the most specific route that matches either IPv4 traffic or IPv6 traffic to determine how to route the traffic.

The following example subnet route table has a route for IPv4 internet traffic (`0.0.0.0/0`) that points to an internet gateway, and a route for `172.31.0.0/16` IPv4 traffic that points to a peering connection (`pcx-11223344556677889`). Any traffic from the subnet that's destined for the `172.31.0.0/16` IP address range uses the peering connection, because this route is more specific than the route for internet gateway. Any traffic destined for a target within the VPC (`10.0.0.0/16`) is covered by the `local` route, and therefore is routed within the VPC. All other traffic from the subnet uses the internet gateway.

Destination	Target
10.0.0.0/16	local
172.31.0.0/16	pcx-11223344556677889
0.0.0.0/0	igw-12345678901234567

Route priority for static and dynamically propagated routes

If you've attached a virtual private gateway to your VPC and enabled route propagation on your subnet route table, routes representing your Site-to-Site VPN connection automatically appear as propagated routes in your route table.

If the destination of a propagated route is identical to the destination of a static route, the static route takes priority. The following resources use static routes:

- internet gateway
- NAT gateway
- Network interface
- Instance ID
- Gateway VPC endpoint
- Transit gateway
- VPC peering connection
- Gateway Load Balancer endpoint

For more information, see Route tables and VPN route priority in the *AWS Site-to-Site VPN User Guide*.

The following example route table has a static route to an internet gateway and a propagated route to a virtual private gateway. Both routes have a destination of `172.31.0.0/24`. Because a static route to an internet gateway takes priority, all traffic destined for `172.31.0.0/24` is routed to the internet gateway.

Destination	Target	Propagated
10.0.0.0/16	local	No
172.31.0.0/24	vgw-11223344556677889	Yes
172.31.0.0/24	igw-12345678901234567	No

Route priority for prefix lists

If your route table references a prefix list, the following rules apply:

- If your route table contains a propagated route that matches a route that references a prefix list, the route that references the prefix list takes priority. Please note that for routes that overlap, more specific routes always take priority irrespective of whether they are propagated routes, static routes, or routes that reference prefix lists.
- If your route table references multiple prefix lists that have overlapping CIDR blocks to different targets, we randomly choose which route takes priority. Thereafter, the same route always takes priority.

Example routing options

The following topics describe routing for specific gateways or connections in your VPC.

Contents

- Routing to an internet gateway
- Routing to a NAT device
- Routing to a virtual private gateway
- Routing to an AWS Outposts local gateway
- Routing to a VPC peering connection

- Routing to a gateway VPC endpoint
- Routing to an egress-only internet gateway
- Routing for a transit gateway
- Routing for a middlebox appliance
- Routing using a prefix list
- Routing to a Gateway Load Balancer endpoint

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 Enable internet access for a VPC using an internet gateway.

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 prefix list that contains the IP address ranges for Amazon S3 in a specific Region) is routed to a gateway VPC endpoint, and 10.25.0.0/16 traffic is routed to a VPC peering connection. These 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 devices.

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

This section describes routing table configurations for routing to an AWS Outposts local gateway.

Contents

- Enable traffic between Outpost subnets and your on-premises network
- Enable traffic between subnets in the same VPC across Outposts

Enable traffic between Outpost subnets and your on-premises network

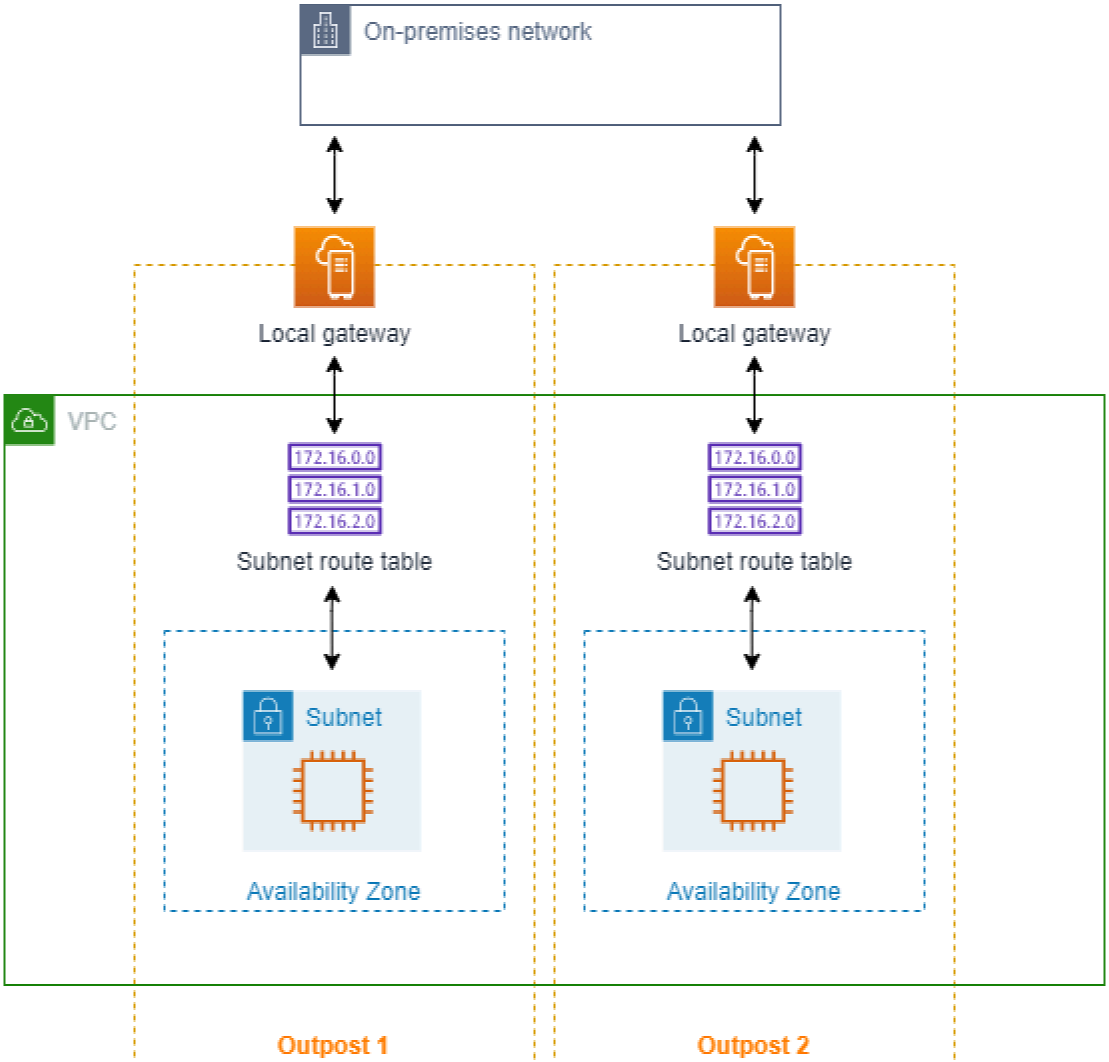
Subnets that are in VPCs associated with AWS Outposts can have an additional target type of a local gateway. Consider the case where you want to have the local gateway route traffic with a destination address of 192.168.10.0/24 to the customer network. To do this, add the following route with the destination network and a target of the local gateway (`lgw-xxxx`).

Destination	Target
192.168.10.0/24	*lgw-id*

Enable traffic between subnets in the same VPC across Outposts

You can establish communication between subnets that are in the same VPC across different Outposts using Outpost local gateways and your on-premise network.

You can use this feature to build architectures similar to multi-Availability Zone (AZ) architectures for your on-premise applications running on Outposts racks by establishing connectivity between Outposts racks that are anchored to different AZs.



To enable this feature, add a route to your Outpost rack subnet route table that is more specific than the local route in that route table and has a target type of local gateway. The destination of the route must match the entire IPv4 block of the subnet in your VPC that is in another Outpost. Repeat this configuration for all the Outpost subnets that need to communicate.

Important

- To use this feature, you must use direct VPC routing. You cannot use your own customer-owned IP addresses.
- Your on-premise network that the Outposts local gateways are connected to must have the required routing so that subnets can access to each other.

- If you want to use security groups for resources in the subnets, you must use rules that include IP address ranges as source or destination in the Outpost subnets. You cannot use security group IDs.
- Existing Outposts racks may require an update to enable support for intra-VPC communication across multiple Outposts. If this feature doesn't work for you, contact AWS Support.

Example Example

For a VPC with a CIDR of 10.0.0.0/16, an Outpost 1 subnet with a CIDR of 10.0.1.0/24, and an Outpost 2 subnet with a CIDR of 10.0.2.0/24, the entry for Outpost 1 subnet’s route table would be as follows:

Destination	Target
10.0.0.0/16	Local
10.0.2.0/24	*lgw-1-id*

The entry for Outpost 2 subnet’s route table would be as follows:

Destination	Target
10.0.0.0/16	Local
10.0.1.0/24	*lgw-2-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. 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 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 Enable outbound IPv6 traffic using an egress-only internet gateway.

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 “Middlebox routing wizard”.
- 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. You can also route outbound traffic from your subnet to a middlebox appliance in another subnet.

For middlebox routing examples, see Middlebox scenarios.

Contents

- Appliance considerations
- Routing traffic between a gateway and an appliance
- Routing inter-subnet traffic to an appliance

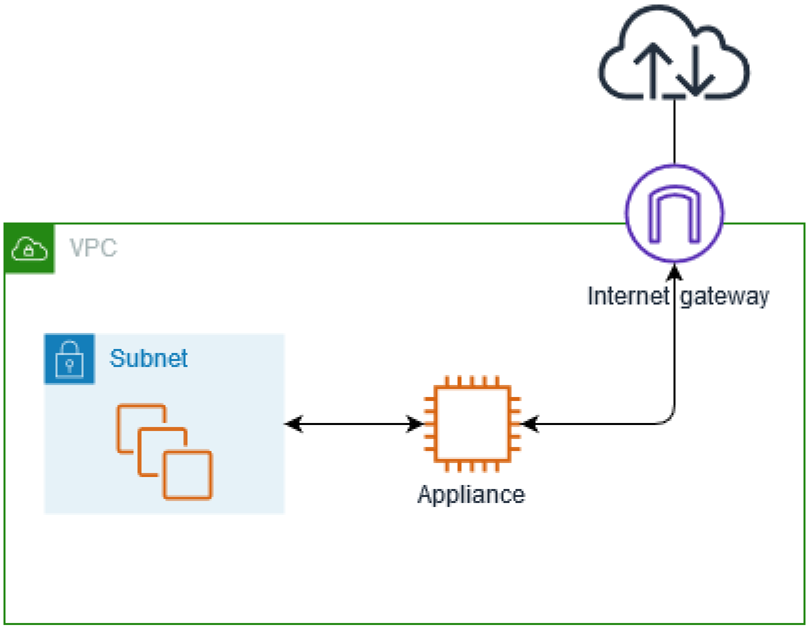
Appliance considerations

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*.
- You cannot route traffic between hosts in the same subnet through an appliance.
- The appliance does not have to perform network address translation (NAT).
- You can add a route to your route tables that is more specific than the local route. You can use more specific routes to redirect traffic between subnets within a VPC (East-West traffic) to a middlebox appliance. The destination of the route must match the entire IPv4 or IPv6 CIDR block of a subnet in your VPC.
- To intercept IPv6 traffic, ensure that your VPC, subnet, and appliance support IPv6.

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.

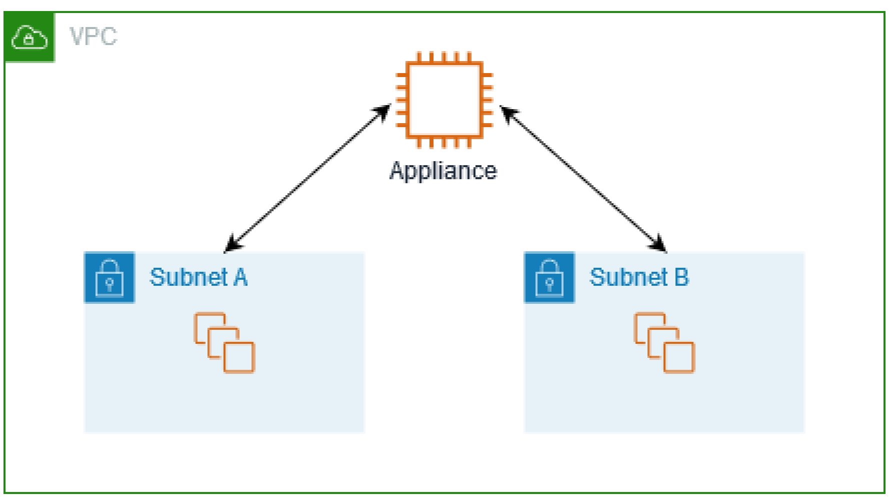
Destination	Target
VPC CIDR	Local
0.0.0.0/0	*Appliance network interface ID*

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 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 create a Gateway Load Balancer, configure a Gateway Load Balancer endpoint service, and then create a Gateway Load Balancer endpoint in your VPC to connect it to the service.

To route your traffic to the Gateway Load Balancer (for example, for security inspection), specify the Gateway Load Balancer endpoint as a target in your route tables.

For an example of a security appliances behind a Gateway Load Balancer, see the section called "Inspect traffic using security appliances".

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*

When using a Gateway Load Balancer endpoint as a target, you cannot specify a prefix list as a destination. If you attempt to create or replace a prefix list route targeting a VPC Endpoint, you'll receive the error: "Cannot create or replace a prefix list route targeting a VPC Endpoint."

For more information, see Gateway Load Balancers.

Create a route table for your VPC

Complete the following tasks to create and configure a custom route table for your VPC. By default your new route table contains local routes that allow communication within the VPC. You can add routes to direct network traffic to specific targets based on the destination IP address range.

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.

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.

> **Note**
>
> 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.

Tasks

- Create the route table
- Add routes to the route table
- Associate a subnet with the route table

Create the route table

To create 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. Choose **Create route table**.
4. (Optional) For **Name**, enter a name for your route table.
5. For **VPC**, choose your VPC.
6. (Optional) To add a tag, choose **Add new tag** and enter the tag key and tag value.
7. Choose **Create route table**.

To create a route table using the AWS CLI

Use the create-route-table command.

Add routes to the route table

To add routes 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. Choose **Add route**.
5. For **Destination** enter one of the following:
   - An IP address range - For example, 192.168.0.0/16
   - A single IP address - For example, 192.168.10.1/32
   - The ID of a prefix list - For example, pl-0abcdef1234567890
6. For **Target**, select a resource type (for example, a network interface) and then enter the ID of the resource (for example, eni-11223344556677889).
7. Choose **Save changes**.

To add routes to a route table using the AWS CLI

Use the create-route command.

Associate a subnet with the route table

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.
5. Choose **Save associations**.

To associate or disassociate a subnet with a route table using the AWS CLI

- associate-route-table
- disassociate-route-table

Manage subnet route tables

Use the following procedures to manage VPC routing using route tables.

Tasks

- Determine the route table for a subnet
- Determine the explicit associations
- Add, modify, and remove routes
- Enable or disable route propagation
- Change the route table for a subnet

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 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.
4. Choose the **Route table** tab to view information about the route table and its routes. To determine whether the association is to the main route table, and if that association is explicit, see Determine the explicit associations.

Determine the explicit associations

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.

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. Check the **Explicit subnet association** column to determine the explicitly associated subnets and the **Main** column to determine whether this is the main route table.
4. Select the route table and choose the **Subnet associations** tab.
5. The subnets under **Explicit subnet associations** are explicitly associated with the route table. The subnets under **Subnets without explicit associations** belong to the same VPC as the route table, but are not associated with any route table, so they are implicitly associated with the main route table for the VPC.

To determine which gateways are explicitly associated using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route tables**.
3. Select the route table and choose the **Edge associations** tab.

To describe one or more route tables and view its associations using the AWS CLI

Use the describe-route-tables command.

Add, modify, and remove routes

You can add, modify, and delete routes for your route tables.

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*.

Considerations

- You can only modify routes that you've added.
- When you modify or delete a route, existing connections that use these routes are affected. Connections that use the other routes are not affected.
- There is a quota on the number of routes that you can add per route table. For more information, see Amazon VPC quotas.

To update the routes for 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 an IP address range, a single IP address, or the ID of a prefix list. For **Target**, select the resource type and then enter the ID of the resource.
5. To modify a route, enter the new destination CIDR block or prefix list ID and choose a target.
6. To delete a route, choose **Remove**.
7. Choose **Save changes**.

To update the routes for a route table using the AWS CLI

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`.

- create-route
- replace-route

- delete-route

Enable or disable route propagation

Route propagation allows a virtual private gateway to automatically propagate routes to your route tables. This means that you don't need to manually add or remove VPN routes.

To complete this process, you must have a virtual private gateway.

For more information, see Site-to-Site VPN routing options in the *Site-to-Site VPN User Guide*.

To enable or 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. Select or clear the **Enable** check box next to the virtual private gateway.
5. Choose **Save**.

To enable or disable route propagation using the AWS CLI

- enable-vgw-route-propagation
- disable-vgw-route-propagation

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. From the **Route table** tab, choose **Edit route table association**.
4. For **Route table ID**, select the new route table.

5. Choose **Save**.

To change the route table associated with a subnet using the AWS CLI

Use the replace-route-table-association command.

Replace the main route table

This section describes how to change which route table is the main route table in your VPC.

To replace the main route table using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route tables**, and then select the new main route table.
3. Choose **Actions**, **Set main route table**.
4. When prompted for confirmation, enter **`set`**, and then choose **OK**.

To replace the main route table using AWS CLI

- Use the replace-route-table-association command.

The following procedure describes how to remove an explicit association between a subnet and the main route table. The result is an implicit association between the subnet and the main route table. The process is the same as disassociating any subnet from any route table.

To remove an explicit association with the main route table

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route tables**, and then select the route table.
3. From the **Subnet associations** tab, choose **Edit subnet associations**.
4. Clear the checkbox for the subnet.
5. Choose **Save associations**.

Control traffic entering your VPC using a gateway route table

To control traffic entering your VPC with a gateway route table, you can associate or disassociate an internet gateway or a virtual private gateway with a route table. For more information, see Gateway route tables.

To associate or disassociate a gateway with a route table using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route tables**, and then select the route table.
3. From the **Edge associations** tab, choose **Edit edge associations**.
4. Select or deselect the checkbox for the gateway.
5. Choose **Save changes**.

To associate a gateway with a route table using the AWS CLI

Use the associate-route-table command. The following example associates the specified route table with the specified internet gateway.

```
aws ec2 associate-route-table
    --route-table-id rtb-01234567890123456 \
    --gateway-id igw-11aa22bb33cc44dd1
```

To disassociate a gateway from a route table using the AWS CLI

Use the disassociate-route-tablecommand. Specify the ID of the association between the route table and the gateway.

```
aws ec2 disassociate-route-table \
    --association-id rtbassoc-0abcdef1234567890
```

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, 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 local route using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route tables**, and then select the route table.
3. From the **Routes** tab, choose **Edit routes**.
4. For the local route, clear **Target** and then choose a new target.
5. Choose **Save changes**.

To restore the target for a local route using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route tables**, and then select the route table.
3. Choose **Actions**, **Edit routes**.
4. For the route, clear **Target**, and then choose **local**.
5. Choose **Save changes**.

To replace the target for a local route using the AWS CLI

Use the replace-route command. The following example replaces the target of the local route with the specified network interface.

```
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 the specified route table.

```
aws ec2 replace-route \
    --route-table-id rtb-01234567890123456 \
    --destination-cidr-block 10.0.0.0/16 \
    --local-target
```

Advanced routing in your VPC

Configure advanced routing scenarios for your VPC. This section covers both static and dynamic routing approaches for managing traffic flow:

- **Static ingress routing**: Configure static routes to direct inbound internet traffic destined for your BYOIP (Bring Your Own IP) address pools to specific network interfaces within your VPC.
- **Dynamic routing with VPC Route Server:** Use BGP-based dynamic routing to automatically update VPC and internet gateway route tables, providing fault tolerance and automatic failover for your workloads.

Contents

- Route internet traffic to a single network interface
- Dynamic routing in your VPC using VPC Route Server

Route internet traffic to a single network interface

You can route inbound internet traffic destined for large pools of public IP addresses to a single elastic network interface (ENI) in your VPC.

Previously, internet gateways only accepted traffic destined for public IP addresses that were directly associated with network interfaces in the VPC. Instance types have limits on the number of IP addresses that can be associated with network interfaces, creating challenges for industries like Telecommunications and Internet of Things (IoT) that need to handle traffic for IP pools larger than these limits.

This routing eliminates complex address translation on inbound internet connections. You can bring your own public IP pools (BYOIP) and configure your VPC internet gateway to accept and route traffic for the entire pool to a single network interface. This feature is particularly valuable for:

- **Telecommunications**: Managing large subscriber IP pools without address translation overhead
- **IoT applications**: Consolidating traffic from thousands of device IP addresses
- **Any scenario**: Requiring traffic routing beyond ENI association limits

You can integrate this routing with VPC Route Server for dynamic route updates during failover scenarios.

Key benefits

This routing approach provides the following benefits:

- **No address translation required** - Direct routing eliminates NAT complexity
- **Bypass ENI limits** - Handle IP pools larger than instance association limits
- **Industry-optimized** - Purpose-built for Telco and IoT requirements
- **Dynamic failover** - Integrates with Route Server for automatic updates

Availability

You can use this feature in all AWS commercial regions, AWS China regions, and AWS GovCloud regions.

Contents

- Before you begin
- How this feature works
- Step 1: Create a VPC
- Step 2: Create and attach an internet gateway
- Step 3: Create a subnet for your target instance
- Step 4: Create a route table for the subnet
- Step 5: Create a security group for the target instance
- Step 6: Launch target EC2 instance
- Step 7: Create the internet gateway route table
- Step 8: Associate the route table with the internet gateway
- Step 9: Associate your BYOIP pool with the internet gateway
- Step 10: Add static route to target your instance
- Step 11: Configure the target instance
- Step 12: Configure instance for traffic handling
- Step 13: Test connectivity
- Troubleshooting
- Advanced option: Route server integration for dynamic routing

- [Clean up]

Before you begin

Before starting this tutorial, ensure you have:

1. **A BYOIP pool**: You must have already brought your own IP address range to AWS. Complete the steps in [Bring your own IP addresses (BYOIP) in Amazon EC2].
2. **Verify your BYOIP pool**: Confirm your pool is ready by running:

   ```
   aws ec2 describe-public-ipv4-pools --region us-east-1
   ```

   Look for your pool in the output and ensure the `PoolAddressRanges` shows `Available` addresses.
3. **Appropriate permissions**: Ensure your AWS account has permissions to create VPC resources, EC2 instances, and manage BYOIP pools.

How this feature works

This section explains the technical concepts behind internet gateway ingress routing and how traffic flows from the internet to your target instance.

Why use internet gateway ingress routing

Previously, you needed to perform address translation to consolidate traffic for large numbers of IP addresses due to ENI association limits. This enhancement removes that complexity by allowing direct routing of BYOIP pools to target instances.

How the routing works

This feature only works with the public IP CIDRs that you bring to AWS following the BYOIP process. The BYOIP process ensures that your account owns the public IP CIDR. Once you have the BYOIP public CIDR:

1. You associate this public IP address pool with an internet gateway route table. The internet gateway must already be associated with a VPC. This association allows the VPC to accept traffic destined for the IP CIDR. Ensure that the internet gateway has a dedicated route table that is not shared with any subnets.

2. Now that you have associated the BYOIP pool with the internet gateway route table, you can enter a route with a destination equal to the IP CIDR or a subset of it in the internet gateway route table. The target of this route would be the ENI where you want to route your traffic.
3. When your traffic destined for your BYOIP CIDR enters AWS, AWS looks at the internet gateway route table and accordingly routes traffic to the relevant VPC.
4. Inside the VPC, the internet gateway routes the traffic to the target ENI.
5. The target (an elastic network interface associated with your workload) processes the traffic.

Best practices

- **Keep route tables separate**: The internet gateway route table must be dedicated only to the internet gateway. Do not associate this route table with any VPC subnets. Use separate route tables for subnet routing.
- **Don't directly assign BYOIP IPs**: Do not associate public IP addresses from your BYOIP pool directly to EC2 instances or network interfaces. The internet gateway ingress routing feature routes traffic to instances without requiring direct IP association.

> ⚠ **Important**
>
> If you are using VPC Block Public Access (BPA), when BPA is enabled, it will block traffic to subnets using ingress routing, even if you've set a subnet-level BPA exclusion. Subnet-level exclusions do not work for ingress routing. To allow ingress routing traffic with BPA enabled:
>
> - Disable BPA completely, or
> - Use a VPC-level exclusion

Step 1: Create a VPC

Complete this step to create a VPC that will host your target instance and internet gateway.

> ⓘ **Note**
>
> Ensure you have not reached your VPC quota limit. For more information, see Amazon VPC quotas.

AWS console

1. Open the [Amazon VPC console](https://console.aws.amazon.com/vpc/).
2. On the VPC dashboard, choose **Create VPC**.
3. For **Resources to create**, choose **VPC only**.
4. For **Name tag**, enter a name for your VPC (for example, **`IGW-Ingress-VPC`**).
5. For **IPv4 CIDR block**, enter a CIDR block (for example, **`10.0.0.0/16`**).
6. Choose **Create VPC**.

AWS CLI

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 'ResourceType=vpc,Tags=[{Key=Name,Value=IGW-Ingress-VPC}]' --region us-east-1
```

Step 2: Create and attach an internet gateway

Complete this step to create an internet gateway and attach it to your VPC to enable internet connectivity.

AWS console

1. Open the [Amazon VPC console](https://console.aws.amazon.com/vpc/).
2. In the VPC console, choose **Internet gateways**.
3. Choose **Create internet gateway**.
4. For **Name tag**, enter a name for your internet gateway (for example, **`IGW-Ingress-Gateway`**).
5. Choose **Create internet gateway**.
6. Select your internet gateway and choose **Actions**, **Attach to VPC**.
7. Select your VPC and choose **Attach internet gateway**.

AWS CLI

```
aws ec2 create-internet-gateway --tag-specifications 'ResourceType=internet-
gateway,Tags=[{Key=Name,Value=IGW-Ingress-Gateway}]' --region us-east-1
```

```
aws ec2 attach-internet-gateway --internet-gateway-id igw-0123456789abcdef0 --vpc-id
 vpc-0123456789abcdef0 --region us-east-1
```

Note: Replace the resource IDs with your actual IDs from the previous step.

Step 3: Create a subnet for your target instance

Complete this step to create a subnet where your target instance will be deployed.

AWS console

1. In the VPC console navigation pane, choose **Subnets**.
2. Choose **Create subnet**.
3. Under **VPC ID**, choose your VPC.
4. For **Subnet name**, enter a name (for example, **`Target-Subnet`**).
5. For **Availability Zone**, you can choose a Zone for your subnet, or leave the default **No Preference** to let AWS choose one for you.
6. For **IPv4 CIDR block**, select **Manual input** and enter a CIDR block (for example, **`10.0.1.0/24`**).
7. Choose **Create subnet**.

AWS CLI

```
aws ec2 create-subnet \
    --vpc-id vpc-0123456789abcdef0 \
    --cidr-block 10.0.1.0/24 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Target-Subnet}]' \
    --region us-east-1
```

Step 4: Create a route table for the subnet

Complete this step to create a route table for your subnet and associate it with the subnet.

AWS console

1. In the VPC console navigation pane, choose **Route tables**.

2. Choose **Create route table**.
3. For **Name**, enter a name for your route table (for example, **`Target-Subnet-Route-Table`**).
4. For **VPC**, choose your VPC.
5. Choose **Create route table**.
6. Select your route table and choose **Actions**, **Edit subnet associations**.
7. Select your subnet and choose **Save associations**.

AWS CLI

```
aws ec2 create-route-table \
    --vpc-id vpc-0123456789abcdef0 \
    --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Target-Subnet-
Route-Table}]' \
    --region us-east-1

aws ec2 associate-route-table \
    --route-table-id rtb-0987654321fedcba0 \
    --subnet-id subnet-0123456789abcdef0 \
    --region us-east-1
```

Step 5: Create a security group for the target instance

Complete this step to create a security group that will control network access to your target instance.

AWS console

1. In the VPC console navigation pane, choose **Security Groups**.
2. Choose **Create security group**.
3. For **Security group name**, enter a name (for example, **`IGW-Target-SG`**).
4. For **Description**, enter **`Security group for IGW ingress routing target instance`**.
5. For **VPC**, select your VPC.
6. To add inbound rules, choose **Inbound rules**. For each rule, choose **Add rule** and specify the following:
   - **Type**: All ICMP - IPv4, **Source**: 0.0.0.0/0 (for ping testing).

- **Type**: SSH, **Port**: 22, **Source**: 0.0.0.0/0 (for EC2 Instance Connect).

> **Note**
>
> This security group opens SSH ports to all internet traffic for this tutorial. This tutorial is for educational purposes and should not be configured for production environments. In production, restrict SSH access to specific IP ranges.

- Choose **Create security group**.

AWS CLI

```
aws ec2 create-security-group \
    --group-name IGW-Target-SG \
    --description "Security group for IGW ingress routing target instance" \
    --vpc-id vpc-0123456789abcdef0 \
    --region us-east-1

aws ec2 authorize-security-group-ingress \
    --group-id sg-0123456789abcdef0 \
    --protocol icmp \
    --port -1 \
    --cidr 0.0.0.0/0 \
    --region us-east-1

aws ec2 authorize-security-group-ingress \
    --group-id sg-0123456789abcdef0 \
    --protocol tcp \
    --port 22 \
    --cidr 0.0.0.0/0 \
    --region us-east-1
```

Step 6: Launch target EC2 instance

Complete this step to launch the EC2 instance that will receive traffic from your BYOIP pool.

AWS console

1. Open the Amazon EC2 console.

2. Choose **Launch instance**.
3. For **Name**, enter a name for your instance (for example, **`IGW-Target-Instance`**).
4. For **Application and OS Images (Amazon Machine Image)**, choose **Amazon Linux 2023 AMI**.
5. For **Instance type**, choose **t2.micro** (eligible for free tier).
6. For **Key pair (login)**, select an existing key pair or create a new one.
7. For **Network settings**, choose **Edit** and configure:
   - **VPC**: Select your VPC
   - **Subnet**: Select your subnet
   - **Auto-assign public IP**: Enable
   - **Firewall (security groups)**: Select existing security group and choose your security group
8. Choose **Launch instance**.
9. **Important**: After launch, go to the instance details and note the **Network interface ID** (starts with "eni-") - you'll need this for Step 10.

AWS CLI

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --count 1 \
    --instance-type t2.micro \
    --key-name your-key-pair \
    --security-group-ids sg-0123456789abcdef0 \
    --subnet-id subnet-0123456789abcdef0 \
    --associate-public-ip-address \
    --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=IGW-Target-
Instance}]' \
    --region us-east-1
```

To find the ENI ID in the console:

1. In the EC2 console, select your instance.
2. Go to the **Networking** tab.
3. Note the **Network interface ID** (for example, `eni-0abcdef1234567890`).

To find the ENI ID using AWS CLI:

```
aws ec2 describe-instances --instance-ids i-0123456789abcdef0 --query
 'Reservations[0].Instances[0].NetworkInterfaces[0].NetworkInterfaceId' --output text
 --region us-east-1
```

Step 7: Create the internet gateway route table

Complete this step to create a dedicated route table for the internet gateway that will handle ingress routing.

AWS console

1. In the VPC console, choose **Route tables**.
2. Choose **Create route table**.
3. For **Name**, enter a name for your route table (for example, **`IGW-Ingress-Route-Table`**).
4. For **VPC**, choose your VPC.
5. Choose **Create route table**.
6. Select your route table and choose the **Edge associations** tab.
7. Choose **Edit edge associations**.
8. Select your internet gateway and choose **Save changes**.

AWS CLI

```
aws ec2 create-route-table \
    --vpc-id vpc-0123456789abcdef0 \
    --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=IGW-Ingress-
Route-Table}]' \
    --region us-east-1
```

Step 8: Associate the route table with the internet gateway

Complete this step to associate your route table with the internet gateway to enable ingress routing functionality.

AWS console

1. In the VPC console navigation pane, choose **Route tables**, and then select the route table you created.
2. From the **Edge associations** tab, choose **Edit edge associations**.

3. Select the checkbox for the internet gateway.
4. Choose **Save changes**.

AWS CLI

```
aws ec2 associate-route-table \
    --route-table-id rtb-0123456789abcdef0 \
    --gateway-id igw-0123456789abcdef0 \
    --region us-east-1
```

Step 9: Associate your BYOIP pool with the internet gateway

Complete this step to associate your BYOIP pool with the internet gateway route table, enabling the VPC to accept traffic for your IP range.

AWS console

1. In the VPC console navigation pane, choose **Route tables**, and select the internet gateway route table you created.
2. Click on the **IPv4 pool associations** tab.
3. Click on the **Edit associations** button.
4. Select your BYOIP pool (for example, `pool-12345678901234567`).
5. Click on the **Save associations** button.

AWS CLI

```
aws ec2 associate-route-table \
    --route-table-id rtb-0123456789abcdef0 \
    --public-ipv4-pool pool-12345678901234567 \
    --region us-east-1
```

Note: Replace `rtb-0123456789abcdef0` with your internet gateway route table ID and `pool-12345678901234567` with your BYOIP pool ID.

Step 10: Add static route to target your instance

Complete this step to add a route that directs traffic from your BYOIP range to your target instance's network interface.

AWS console

1. In the VPC console navigation pane, choose **Route tables**, and select the internet gateway route table you created.
2. Choose **Actions**, **Edit routes**.
3. Choose **Add route**.
4. For **Destination**, enter your BYOIP CIDR or a subset (for example, **`203.0.113.0/24`**). The prefix must be between /23 and /28.
5. For **Target**, select **Network interface** and enter your instance's ENI ID (for example, `eni-0abcdef1234567890`).
6. Choose **Save changes**.

AWS CLI

```
aws ec2 create-route \
    --route-table-id rtb-0123456789abcdef0 \
    --destination-cidr-block 203.0.113.0/24 \
    --network-interface-id eni-0abcdef1234567890 \
    --region us-east-1
```

Step 11: Configure the target instance

Complete this step to configure your target instance to properly handle traffic destined for BYOIP addresses.

Important: Complete this instance configuration step before testing connectivity (Step 12). The instance must be configured to respond to BYOIP addresses for the ingress routing to work properly.

AWS console

1. Connect to your target instance using EC2 Instance Connect:
   - In the EC2 console, select your instance.
   - Choose **Actions** > **Connect**.
   - Select **EC2 Instance Connect** tab.
   - Choose **Connect**.
2. Add specific BYOIP IP address to your instance interface:

First, find your network interface name:

```
ip link show
```

Then add the IP address (replace `203.0.113.10` with an IP from your BYOIP range):

```
sudo ip addr add 203.0.113.10/32 dev eth0
```

Note: Replace `203.0.113.10` with any IP address from your BYOIP range that you want to test. The interface name may be `eth0`, `ens5`, or similar depending on your instance type.

3. In the EC2 console, disable source/destination check:

   - Select your instance.
   - Go to **Networking** tab, click on the network interface.
   - Choose **Actions**, **Change source/dest check**, **Disable**.

AWS CLI

```
aws ec2 modify-network-interface-attribute \
    --network-interface-id eni-0abcdef1234567890 \
    --no-source-dest-check \
    --region us-east-1
```

Step 12: Configure instance for traffic handling

Complete this step to add BYOIP addresses to your instance and disable source/destination checking to enable proper traffic handling.

AWS console

1. Connect to your target instance using EC2 Instance Connect:

   - In the EC2 console, select your instance.
   - Choose **Actions** > **Connect**.
   - Select **EC2 Instance Connect** tab.
   - Choose **Connect**.
2. Add specific BYOIP IP address to your instance interface:

First, find your network interface name:

```
ip link show
```

Then add the IP address (replace `ens5` with your actual interface name):

```
sudo ip addr add 203.0.113.10/32 dev ens5
```

Note: Replace `203.0.113.10` with any IP address from your BYOIP range that you want to test. The interface name may be `eth0`, `ens5`, or similar depending on your instance type.

3. In the EC2 console, disable source/destination check:

   - Select your instance.
   - Go to **Networking** tab, click on the network interface.
   - Choose **Actions**, **Change source/dest check**, **Disable**.

AWS CLI

```
aws ec2 modify-network-interface-attribute \
    --network-interface-id eni-0abcdef1234567890 \
    --no-source-dest-check \
    --region us-east-1
```

Step 13: Test connectivity

Complete this step to verify that internet traffic is properly routed to your target instance through the BYOIP addresses.

1. On your target instance, monitor incoming traffic using tcpdump:

```
sudo tcpdump -i any icmp
```

2. From another terminal or computer, test connectivity to your BYOIP IP address:

```
ping 203.0.113.10
```

3. Expected results:

- Ping should succeed and show responses from your BYOIP IP address.
- tcpdump should show incoming packets for the BYOIP address, similar to:

```
12:34:56.789012 IP 203.0.113.100 > 203.0.113.10: ICMP echo request, id 1234, seq
 1, length 64
12:34:56.789123 IP 203.0.113.10 > 203.0.113.100: ICMP echo reply, id 1234, seq 1,
 length 64
```

- Traffic should appear to come from external IP addresses, proving the internet gateway ingress routing is delivering internet traffic to your instance.

Troubleshooting

Use this section to resolve common issues you might encounter when setting up internet gateway ingress routing.

Traffic not reaching the instance

- Verify the route table has the correct ENI ID as the target.
- Confirm the BYOIP pool is associated with the internet gateway route table.
- Check that source/destination check is disabled on the instance.
- Ensure security groups allow the traffic type you're testing.

Route creation fails

- Verify the BYOIP pool is properly associated with the route table.
- Confirm the destination CIDR is within your BYOIP range.
- Check that the target ENI exists and is attached to a running instance.
- Ensure your BYOIP prefix is between /23 and /28 (prefixes outside this range are not supported).

Ping/connectivity fails

- Verify the IP addresses are added to the instance interface.
- Check security groups allow ICMP (for ping) or relevant ports.
- Confirm the instance is in a running state.
- Test from multiple external locations.

Advanced option: Route server integration for dynamic routing

For environments requiring automatic failover, this feature integrates with VPC Route Server to:

- **Dynamically update routes** during instance failures.
- **Eliminate manual intervention** for route management.
- **Provide enterprise-grade availability** for critical workloads.

This is particularly important for Telco and IoT use cases where high availability is essential.

> **Note**
>
> When using Route Server with multiple BGP peers, be aware that a maximum of 32 BGP peers can advertise the same prefix to the same route table using route server.

For environments requiring dynamic routing, automatic failover, and load distribution across multiple instances, consider integrating with AWS Route Server. Route Server enables BGP-based dynamic routing instead of static routes, providing:

- **Dynamic route advertisement** from instances via BGP.
- **Automatic failover** between multiple target instances.
- **Load distribution** across multiple endpoints.
- **Centralized route management** through BGP protocols.

This is an important use case for enterprise deployments where high availability and dynamic routing capabilities are required. For detailed Route Server setup instructions, see the [AWS Route Server documentation].

Clean up

To avoid ongoing charges, delete the resources you created in this tutorial:

Step 1: Terminate EC2 instance

Complete this step to terminate the EC2 instance and stop incurring charges for compute resources.

AWS console

1. Open the Amazon EC2 console.
2. In the EC2 console navigation pane, choose **Instances**.
3. Select your instance and choose **Instance state**, **Terminate instance**.
4. Choose **Terminate** to confirm.

AWS CLI

```
aws ec2 terminate-instances --instance-ids i-0123456789abcdef0 --region us-east-1
```

Step 2: Detach internet gateway from VPC

Complete this step to detach and delete the internet gateway from your VPC.

AWS console

1. Open the Amazon VPC console.
2. In the VPC console navigation pane, choose **Internet gateways**.
3. Select your internet gateway and choose **Actions**, **Detach from VPC**.
4. Choose **Detach internet gateway**.
5. After detaching, choose **Actions**, **Delete internet gateway**.
6. Choose **Delete internet gateway**.

AWS CLI

```
aws ec2 detach-internet-gateway --internet-gateway-id igw-0123456789abcdef0 --vpc-id
 vpc-0123456789abcdef0 --region us-east-1

aws ec2 delete-internet-gateway --internet-gateway-id igw-0123456789abcdef0 --region
 us-east-1
```

Step 3: Delete VPC

Complete this step to delete the VPC and all associated resources to complete the cleanup process.

AWS console

1. In the VPC console, choose **Your VPCs**.

2. Select your VPC and choose **Actions**, **Delete VPC**.
3. Type **delete** to confirm and choose **Delete**.

AWS CLI

```
aws ec2 delete-vpc --vpc-id vpc-0123456789abcdef0 --region us-east-1
```

> **Note**
>
> Deleting the VPC will also delete associated subnets, route tables, and security groups.

> **Note**
>
> Your BYOIP pool remains available for future use and is not deleted as part of this cleanup process.

Dynamic routing in your VPC using VPC Route Server

Amazon VPC Route Server simplifies routing for traffic between workloads that are deployed within a VPC and its internet gateways. With this feature, VPC Route Server dynamically updates VPC and internet gateway route tables with your preferred IPv4 or IPv6 routes to achieve routing fault tolerance for those workloads. This enables you to automatically reroute traffic within a VPC, which increases the manageability of VPC routing and interoperability with third-party workloads.

Route server supports the following route table types:

- VPC route tables not associated with subnets
- Subnet route tables
- Internet gateway route tables

Route server does not support route tables associated with virtual private gateways. To propagate routes into a transit gateway route table, use Transit Gateway Connect.

Quotas

For quotas associated with Amazon VPC Route Server, see Route server quotas.

Pricing

For information about costs associated with Amazon VPC Route Server, see the VPC Route Server tab on the Amazon VPC pricing page.

Contents

- Terminology
- How Amazon VPC Route Server works
- Route server peer logging
- Get started tutorial

Terminology

The following terms are used in this guide:

- **FIB**: The Forwarding Information Base (FIB) serves as a forwarding table for what route server has determined are the best-path routes in the RIB after evaluating all available routing information and policies. The FIB routes that are installed on the route tables. The FIB is recomputed whenever there are changes to the RIB.
- **RIB**: The Routing Information Base (RIB) serves as a database that stores all the routing information and network topology data collected by a router or routing system, such as routes learned from BGP peers. The RIB is constantly updated as new routing information is received or existing routes change. This ensures that the route server always has the most current view of the network topology and can make optimal routing decisions.
- **Route server**: The route server component updates your VPC and internet gateway route tables with the IPv4 or IPv6 routes in your Forwarding Information Base (FIB). The route server represents a single FIB and Routing Information Base (RIB).
- **Route server association**: A route server association is the connection established between a route server and a VPC.
- **Route server endpoint**: A route server endpoint is an AWS-managed component inside a subnet that facilitates BGP (Border Gateway Protocol) connections between your route server and your BGP peers.
- **Route server peer**: A route server peer is a session between a route server endpoint and the device deployed in AWS (such as a firewall appliance or other network security function running on an EC2 instance). The device must meet these requirements:

  - Have an elastic network interface in the VPC
  - Support BGP (Border Gateway Protocol)
  - Can initiate BGP sessions
- **Route server propagation**: When enabled, route server propagation installs the routes in the FIB on the route table you've specified. Route server supports IPv4 and IPv6 route propagation.

How Amazon VPC Route Server works

This section explains how Amazon VPC Route Server works and helps you understand how it achieves routing fault tolerance for your workloads running in subnets.

Contents

- Overview
- Diagrams

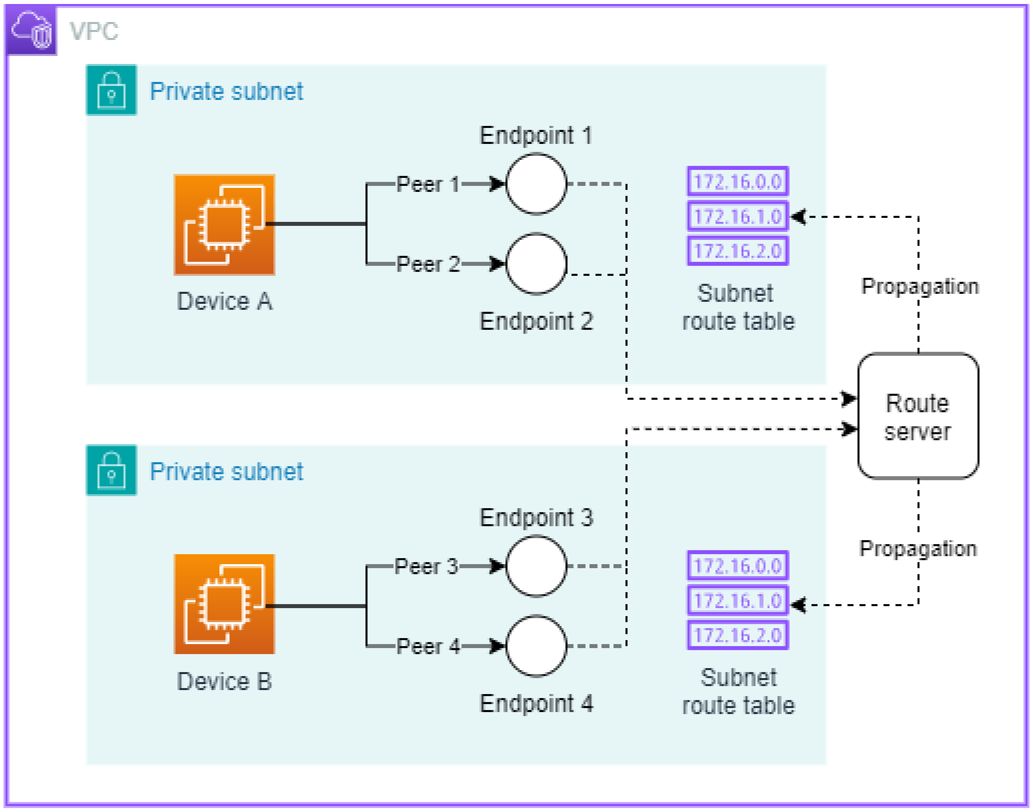
Overview

How Amazon VPC Route Server works:

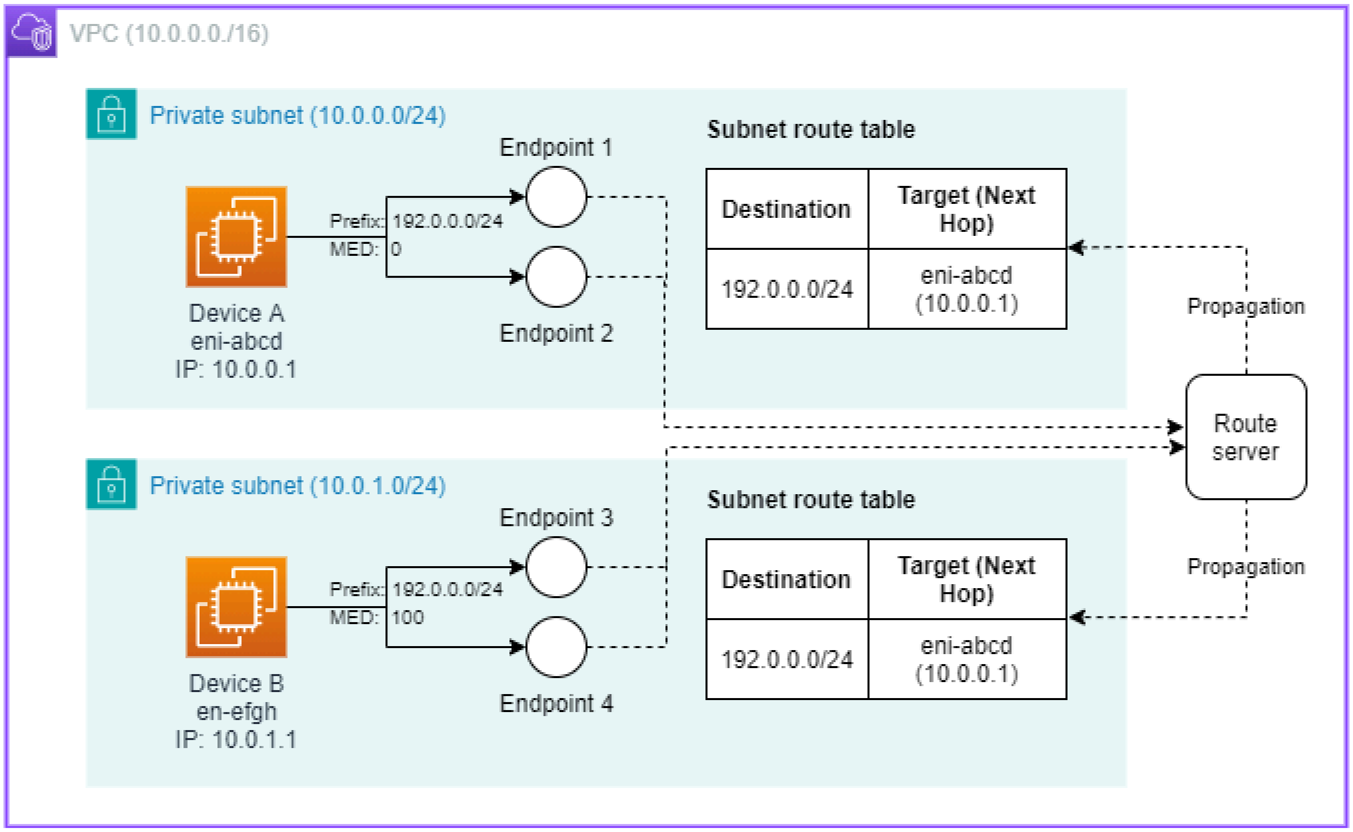
1. You configure a network device (like a firewall running on an EC2 instance in the VPC) to use Amazon VPC Route Server.
2. The network device fails.
3. The route server endpoints detect the failure through BFD (Bidirectional Forwarding Detection) configured on the route server peer.
4. The route server endpoints update the route server to withdraw routes in a Routing Information Base (RIB) where the failed device is the next hop.
5. The route server computes a Forwarding Information Base (FIB) from the RIB, selecting the best available routes.
6. Route server updates the configured route tables with the routes from the FIB.
7. All new traffic is forwarded to the standby device.

Diagrams

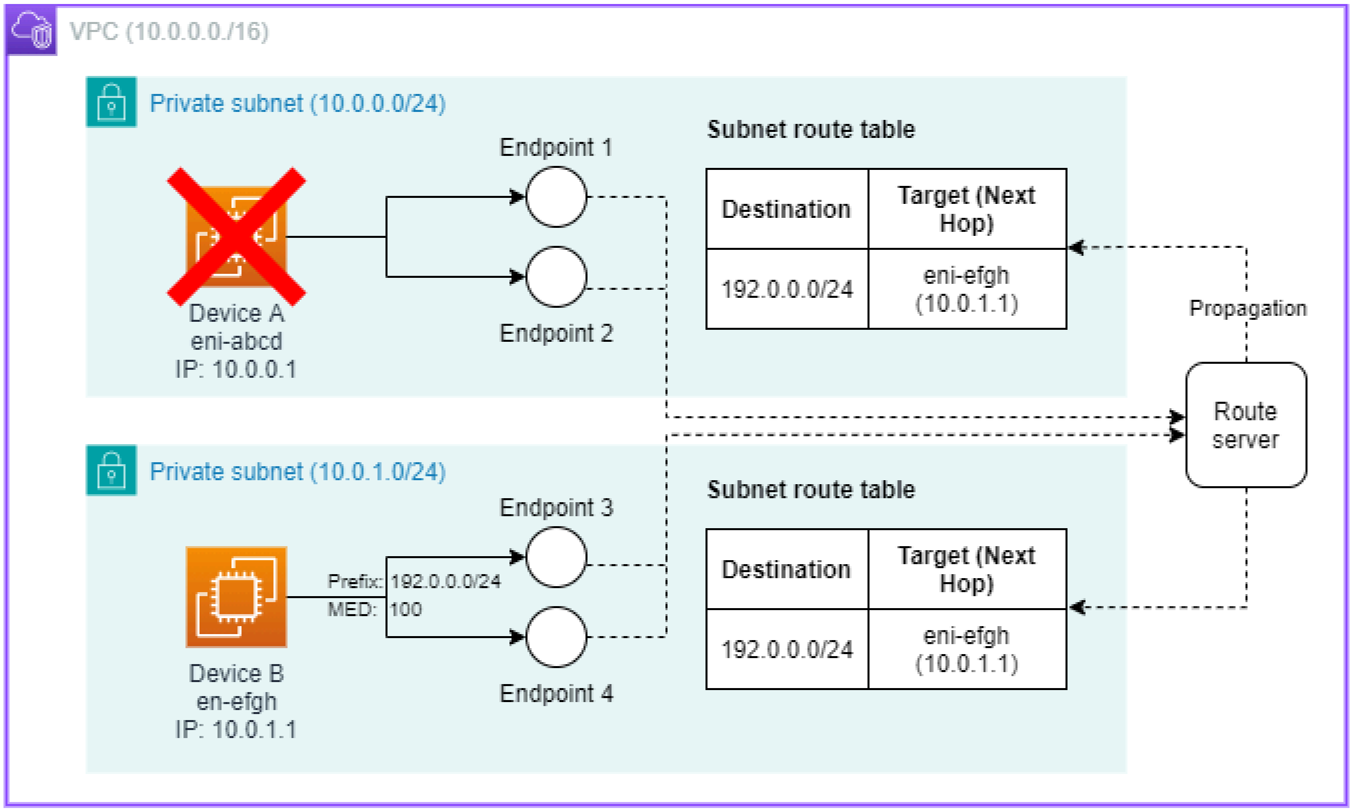
The following is an example diagram of VPC route server with route server endpoints configured for devices in two subnets.



Starting with the example above as a baseline, the example below shows a more detailed design, where both Device A and Device B advertise over BGP that they can accept any traffic with a destination IP in the range of 192.0.0.0/24 (from 192.0.0.0 to 192.0.0.255). The MED (Multi-Exit Discriminator) attribute of 0 tells route server that Device A should be preferred over Device B. The route server receives the route and the MED attribute from Device A and installs that route in the subnet route tables with the network interface of Device A as the "next hop". As a result, any traffic within the subnet with a destination IP in the 192.0.0.0/24 range is sent to Device A. Device A then processes the traffic and sends it onward. Traffic within either subnet (10.0.0.0/24 or 10.0.1.0/24) that is bound for 192.0.0.0/24 will be routed to Device A eni-abcd (10.0.0.1) as the next hop.



This last example below shows how route server handles failover. While the higher MED attribute tells route server that Device B is less preferred than Device A, if Device A eni-abcd (10.0.0.1) goes down, route server updates the subnet route tables, and traffic to 192.0.0.0/24 is routed to Device B eni-efgh (10.0.1.1) as the next hop.



Route server peer logging

Use VPC Route Server peer logging when you need to:

- Monitor BGP and BFD session health
- Troubleshoot connection issues
- Review historical session changes
- Track network status

Pricing

- **CloudWatch**: Data ingestion and archival charges for vended logs apply when you publish route server peer logs to CloudWatch Logs.
- **S3**: Data ingestion and archival charges for vended logs apply when you publish route server peer logs to Amazon S3.
- **Data Firehose**: Standard ingestion and delivery charges apply.

Vended logs are logs from specific AWS services that are available at volume tiered pricing and delivered to CloudWatch Logs, Amazon S3, or Amazon Data Firehose. For more information, open Amazon CloudWatch Pricing, select **Logs** and find **Vended Logs**.

Example log format

```
{
    "resource_arn": "arn:aws:ec2:us-east-1:111122223333:route-server-peer/
rsp-1234567890abcdef0",
    "event_timestamp": 1746643505367,
    "type": "RouteStatus",
    "status": "ADVERTISED",
    "message": {
        "prefix": "10.24.34.0/32",
        "asPath": "65000",
        "med": 100,
        "nextHopIp": "10.24.34.1"
    }
}

{
    "resource_arn": "arn:aws:ec2:us-east-1:111122223333:route-server-peer/
rsp-1234567890abcdef0",
    "event_timestamp": 1746643490000,
    "type": "BGPStatus",
    "status": "UP",
    "message": null
}
```

Where:

- The `resource_arn` is the ARN for the route server peer.
- The `event_timestamp` is the timestamp of the event.
- The `type` of log events we produce (`RouteStatus`, `BGPStatus`, `BFDStatus`).
- The `status` field is the status update.
  - For `RouteStatus` type messages
    - `ADVERTISED` (route was advertised by the peer)
    - `UPDATED` (existing route was updated by the peer)
    - `WITHDRAWN` (route was withdrawn by peer)

- For `BFDStatus` and `BGPStatus` updates
  - `UP`, `DOWN`.
- The `message` field is currently only used for route attributes for the RouteStatus message type but may be populated with relevant information for any type.

AWS Management Console

To create route server peer logs:

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **Virtual private cloud**, choose **Route servers**.
3. On the **Route servers** page, choose **Route server peers**.
4. Choose the **Log delivery** tab.
5. Choose **Add log delivery**.
6. Choose a destination and configure the settings:
   - Amazon CloudWatch Logs
     - **Log type**: Types of logs to deliver. The only supported log type is EVENT_LOGS.
     - **Destination log group**: The CloudWatch log group where logs will be sent. You can pick an existing log group or create a new one (example: /aws/vpc/route-server-peers).
     - **Field selection**: Data fields to include in your logs.
     - **Output format**: How logs are formatted:
       - JSON: Structured format for computer processing
       - Text: Plain text format
     - **Field delimiter**: When using Text format, this is the character that separates fields (example: comma, tab, space).
   - Amazon S3
     - Cross account - Sending logs to different AWS accounts
       - **Log type**: Types of logs to deliver. The only supported log type is EVENT_LOGS.
       - **Delivery destination ARN**: The Amazon Resource Name of the S3 bucket in another AWS account where logs will be sent.
       - **Field selection**: Data fields to include in your logs.
       - **Suffix**: The ending added to log file names (example: .log, .txt).

        - **Hive-compatible**: When turned on, organizes logs in a folder structure that works with Hive-based tools for easier searching with services like Amazon Athena.
        - **Field delimiter**: When using Text format, this is the character that separates fields.
    - In current account
        - **Log type**: Types of logs to deliver. The only supported log type is EVENT_LOGS.
        - **Destination S3 bucket**: The S3 bucket in your account where logs will be sent. You can specify a subfolder path.
        - **Field selection**: Data fields to include in your logs.
        - **Suffix**: The ending added to log file names (example: .log, .txt).
        - **Hive-compatible**: When turned on, organizes logs in a folder structure that works with Hive-based tools for easier searching.
        - **Field delimiter**: When using Text format, this is the character that separates fields.
- Amazon Data Firehose
    - Cross account
        - **Log type**: Types of logs to deliver. The only supported log type is EVENT_LOGS.
        - **Delivery destination ARN**: The Amazon Resource Name of the Firehose delivery stream in another AWS account.
        - **Field selection**: Data fields to include in your logs.
        - **Field delimiter**: When using Text format, this is the character that separates fields.
    - In current account
        - **Log type**: Types of logs to deliver. The only supported log type is EVENT_LOGS.
        - **Delivery destination stream**: The Firehose delivery stream in your account where logs will be sent. The stream must use the "Direct Put" source type.
        - **Field selection**: Data fields to include in your logs.
        - **Output format**: How logs are formatted:
            - JSON: Structured format for computer processing
            - Text: Plain text format
        - **Field delimiter**: When using Text format, this is the character that separates fields.

Command line

The commands in this section link to the AWS CLI Reference documentation. The documentation provides detailed descriptions of the options that you can use when you run the commands.

To create route server peer logs:

1. Use the put-delivery-source command.

   - Example request

```
aws logs put-delivery-source --name "source-rsp-1234567890abcdef0" --
resource-arn "arn:aws:ec2:us-east-1:111122223333:route-server-peer/
rsp-1234567890abcdef0" --log-type "EVENT_LOGS"
```

   - Example response

```
{
     "deliverySource": {
        "name": "source-rsp-1234567890abcdef0",
        "arn": "arn:aws:logs:us-east-1:111122223333:delivery-source:source-
rsp-1234567890abcdef0",
        "resourceArns": [
            "arn:aws:ec2:us-east-1:111122223333:route-server-peer/
rsp-1234567890abcdef0"
        ],
        "service": "ec2",
        "logType": "EVENT_LOGS"
    }
}
```

2. Use the put-delivery-destination command.

   - The following AWS CLI example creates a route server log. The logs are delivered to the specified log group.
   - Example request

```
aws logs put-delivery-destination --name "destination-rsp-abcdef01234567890"
 --destination-resource-arn "arn:aws:logs:us-east-1:111122223333:log-group:/
aws/vendedlogs/ec2/route-server-peer/EVENT_LOGS/rsp-abcdef01234567890"
```

- Example response

```
{
     "deliveryDestination": {
        "name": "destination-rsp-abcdef01234567890",
        "arn": "arn:aws:logs:us-east-1:111122223333:delivery-
destination:destination-rsp-abcdef01234567890",
        "deliveryDestinationType": "CWL",
        "deliveryDestinationConfiguration": {
            "destinationResourceArn": "arn:aws:logs:us-
east-1:111122223333:log-group:/aws/vendedlogs/ec2/route-server-peer/
EVENT_LOGS/rsp-abcdef01234567890"
        }
    }
}
```

3. Use the create-delivery command.

   - Example request

```
aws logs create-delivery --delivery-source-name "source-rsp-1234567890abcdef0"
 --delivery-destination-arn "arn:aws:logs:us-east-1:111122223333:delivery-
destination:destination-rsp-abcdef01234567890"
```

   - Example response

```
{
     "delivery": {
        "id": "1234567890abcdef0",
        "arn": "arn:aws:logs:us-
east-1:111122223333:delivery:1234567890abcdef0",
        "deliverySourceName": "source-rsp-1234567890abcdef0",
        "deliveryDestinationArn": "arn:aws:logs:us-
east-1:111122223333:delivery-destination:destination-rsp-abcdef01234567890",
        "deliveryDestinationType": "CWL",
        "recordFields": [
            "resource_arn",
            "event_timestamp",
            "type",
            "status",
            "message"
        ]
    }
```

```
}
```

Get started tutorial

This tutorial walks you through the process of setting up and configuring VPC Route Server to enable dynamic routing in your VPC. You'll learn how to create and configure all the necessary components, establish BGP peering, and verify proper operation. The tutorial covers everything from initial IAM setup through testing and cleanup.

Before beginning this tutorial, ensure you have:

- Administrative access to your AWS account
- A VPC with at least two subnets where you want to enable dynamic routing
- Network devices (like firewalls running on EC2 instances) that support BGP and can serve as route server peer devices
- Basic familiarity with BGP concepts and AWS networking

The steps can be completed using either the AWS Management Console or AWS CLI. Both methods are provided for each step.

Estimated time to complete: 15-30 minutes

Steps

- Step 1: Configure required IAM Role permissions
- Step 2: Create a route server
- Step 3: Associate route server with a VPC
- Step 4: Create route server endpoints
- Step 5: Enable route server propagation
- Step 6: Create route server peer
- Step 7: Initiate BGP sessions from the devices
- Step 8: Cleanup

Step 1: Configure required IAM Role permissions

To use VPC Route Server, ensure that the IAM user or role you are using has the required IAM permissions. Below is a guide to which permissions are required for each API:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateRouteServer",
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DeleteRouteServer",
            "Effect": "Allow",
            "Action": [
                "sns:DeleteTopic"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateRouteServerEndpoint",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:CreateSecurityGroup",
                "ec2:DescribeSecurityGroups",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DeleteRouteServerEndpoint",
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteSecurityGroup",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:CreateTags",
```

```
                "ec2:DeleteTags"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateRouteServerPeer",
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DeleteRouteServerPeer",
            "Effect": "Allow",
            "Action": [
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "*"
        }
    ]
}
```

Step 2: Create a route server

Complete the steps in this section to create a route server.

The route server component updates your VPC and internet gateway route tables with the IPv4 or IPv6 routes in your Forwarding Information Base (FIB). The route server represents a single FIB and Routing Information Base (RIB).

AWS Management Console

To create a route server

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **Virtual private cloud**, choose **Route servers**.
3. On the **Route servers** page, choose **Create route server**.
4. On the **Create route server** page, configure the following settings:

- For **Name**, enter a name for your route server (e.g., "my-route-server-01"). The name must be 255 characters or less in length.
- For **Amazon Side ASN**, enter a BGP ASN value. This value must be in the range of 1-4294967295. We recommend using a private ASN in the 64512–65534 (16-bit ASN) or 4200000000–4294967294 (32-bit ASN) range.
- For **Persist routes**, choose either **Enable** or **Disable**. This option determines whether routes should be maintained after all BGP sessions are terminated:
  - If enabled: Routes will be preserved in the route server's routing database even if all BGP sessions end
  - If disabled: Routes will be removed from the routing database when all BGP sessions end
- If you enabled persist routes, for **Persist duration**, enter a value between 1-5 minutes. This duration specifies how long the route server will wait after BGP is re-established to unpersist the routes. For example, if you set it to 1 minute, your device has 1 minute after re-establishing BGP to relearn and advertise its routes before the route server resumes normal functionality. While 1 minute is typically sufficient, you can set up to 5 minutes if your BGP network needs more time to fully re-establish and re-learn all routes.
- (Optional) To enable SNS notifications for BGP status changes, toggle the **Enable SNS notifications** switch. Enabling SNS notifications persists BGP or BFD session status changes on route server peers and maintenance notifications for route server endpoints to an SNS topic provisioned by AWS. For details about these notifications, see the **SNS notification details** table below.

5. (Optional) To add tags to your route server, scroll down to the **Tags - optional** section and choose **Add new tag**. Enter a key and an optional value for each tag. You can add up to 50 tags.
6. Review your settings and choose **Create route server**.
7. Wait for the route server to be created. Once complete, you will be redirected to the **Route servers** page, where you can see your new route server listed with a status of *Available*.

Command line

Use the following procedure to create a new route server to manage dynamic routing in a VPC.

For `--amazon-side-asn`, enter a BGP ASN value. This value must be in the range of 1-4294967295. We recommend using a private ASN in the 64512–65534 (16-bit ASN) or 4200000000–4294967294 (32-bit ASN) range.

1. Command:

   ```
   aws ec2 create-route-server --amazon-side-asn 65000
   ```

   Response:

   ```
   {
       "RouteServer": {
           "RouteServerId": "rs-1",
           "AmazonSideAsn": 65000,
           "State": "pending"
       }
   }
   ```

2. Wait for the route server to be available.

   Command:

   ```
   aws ec2 describe-route-servers
   ```

   Response:

   ```
   {
       "RouteServer": {
           "RouteServerId": "rs-1",
           "AmazonSideAsn": 65000,
           "State": "available"
       }
   }
   ```

SNS notification details

The following table shows details about the messages that Amazon VPC Route Server will send using Amazon SNS:

Standard fields		**Message attributes (Metadata)**			
Message	**When it is sent**	**timestamp**	**eventCode**	**routeServerEndpointId**	**affectedRouteServerPeerIds**
Route Server Endpoint [ENDPOINT ID] is now undergoing maintenance. BFD and BGP sessions may be impacted.	Route server endpoint maintenance	Format: 2025-02-17T15:55:00Z	ROUTE_SERVER_ENDPOINT_MAINTENANCE	Affected endpoint ID	List of affected peer IDs
Message	**When it is sent**	**timestamp**	**eventCode**	**routeServerPeerId**	**newBgpStatus**
BGP for Route Server Peer [PEER ID] is now [UP/DOWN].	Route server peer BGP status change	Format: 2025-02-17T15:55:00Z	ROUTE_SERVER_PEER_BGP_STATUS_CHANGE	Affected peer ID	UP or DOWN
Message	**When it is sent**	**timestamp**	**eventCode**	**routeServerPeerId**	**newBfdStatus**
BFD for Route Server Peer [PEER ID] is now [UP/DOWN].	Route server peer BFD status change	Format: 2025-02-17T15:55:00Z	ROUTE_SERVER_PEER_BFD_STATUS_CHANGE	Affected peer ID	UP or DOWN

Step 3: Associate route server with a VPC

Complete the steps in this section to associate the route server with a VPC.

A route server association is the connection established between a route server and a VPC. This is a fundamental configuration step that enables the route server to work with appliances in your VPC.

When you create a route server association:

- It links the route server to a specific VPC.
- It enables the route server to interact with route tables within the VPC's subnets.
- It allows the route server to receive and propagate routes within the associated VPC.
- It establishes the scope of where the route server can operate.

Key aspects of a route server association:

- Each route server can be associated with one VPC. Each VPC can have up to 5 separate route server associations by default. For more information about quotas, see Route server quotas.
- The association must be created before the route server can manage routes.
- The association can be monitored to track its state (such as associating and associated).
- The association can be removed (disassociated) if you no longer want the route server to operate in that VPC.

AWS Management Console

Associate a route server with a VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **Virtual private cloud**, choose **Route servers**.
3. Select the route server you want to associate with a VPC.
4. On the **Association tab**, choose **Associate route server**.
5. In the Associate route server dialog box:
   - The **Route server ID** field is automatically populated with your selected route server
   - For **VPC ID**, choose the VPC you want to associate from the dropdown list
6. Choose **Associate route server**.

7. Wait for the association to complete. Once finished, the **State** will show as *Associated* on the **Association** tab.

Command line

Use the following procedure to associate a route server with a VPC.

1. Command:

```
aws ec2 associate-route-server --route-server-id rs-1 --vpc-id vpc-1
```

   Response:

```
{
    "RouteServerAssociation": {
        "RouteServerId": "rs-1",
        "VpcId": "vpc-1",
        "State": "associating"
    }
}
```

2. Wait for the association to complete.

   Command:

```
aws ec2 get-route-server-associations --route-server-id rs-1
```

   Response:

```
{
    "RouteServerAssociation": {
        "RouteServerId": "rs-1",
        "VpcId": "vpc-1",
        "State": "associated"
    }
}
```

Step 4: Create route server endpoints

Complete the steps in this section to create route server endpoints. Create two endpoints per subnet for redundancy.

A route server endpoint is an AWS-managed component inside a subnet that facilitates BGP (Border Gateway Protocol) connections between your route server and your BGP peers.

Route server endpoints are the "contact points" where your network devices establish BGP sessions with the route server. They're the components that actually handle the BGP connections, while the route server itself manages the routing decisions and route propagation.

> **Note**
>
> Route server endpoints are charged $0.75 per hour.

AWS Management Console

To create route server endpoints

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **Virtual private cloud**, choose **Route servers**.
3. Select the route server for which you want to create endpoints.
4. In the lower pane, choose the **Route server endpoints** tab.
5. Choose **Create route server endpoint**.
6. On the **Create route server endpoint** page, configure the following settings:
   - For **Name**, enter a descriptive name for your endpoint.
   - For **Route server**, confirm that the correct route server is selected.
   - For **Subnet**, select the subnet in which you want to create the endpoint.
7. (Optional) To add tags to your route server endpoint, scroll down to the **Tags - optional** section and choose **Add new tag**. Enter a key and an optional value for each tag.
8. Review your settings and choose **Create route server endpoint**.
9. Wait for the endpoint to be created. Once complete, you will see a success message.
10. Repeat steps 5-9 to create a second endpoint in the same subnet, using a different name.

11. Repeat steps 5-10 for each subnet where you need route server endpoints.
12. After creating the endpoints, return to the **Route server endpoints** tab for your route server.
13. Verify that you see two endpoints listed for each subnet.
14. Check that the **State** for each endpoint is *Available*.

Command line

Use the following procedure to create a route server endpoint.

1. Command:

   ```
   aws ec2 create-route-server-endpoint --route-server-id rs-1 --subnet-id subnet-1
   ```

   Response:

   ```
   {
       "RouteServerEndpoint": {
           "RouteServerId": "rs-1",
           "RouteServerEndpointId": "rse-1",
           "VpcId": "vpc-1",
           "SubnetId": "subnet-1",
           "State": "pending"
       }
   }
   ```

2. You may need to wait a few minutes for the endpoints to become fully available after creation.

   Command:

   ```
   aws ec2 describe-route-server-endpoints
   ```

   Response:

   ```
   {
       "RouteServerEndpoint": {
           "RouteServerId": "rs-1",
           "RouteServerEndpointId": "rse-1",
   ```

```
        "VpcId": "vpc-1",
        "SubnetId": "subnet-1",
        "EniId": "eni-123",
        "EniAddress": "10.1.2.3",
        "State": "available"
    }
}
```

Repeat the steps to create a second endpoint in the same subnet using a different name and create endpoints for each subnet where you need route server endpoints.

Step 5: Enable route server propagation

Complete this step to enable route server propagation.

When enabled, route server propagation installs the routes in the FIB on the route table you've specified. Route server supports IPv4 and IPv6 route propagation.

Route server propagation is the mechanism that automates route table updates - instead of manually updating route tables, the route server automatically propagates the appropriate routes to the configured route tables with routes from the FIB.

Key aspects of route server propagation:

- Configuration
  - Links a route server to specific route tables
  - Determines which route tables will receive dynamic route updates
  - Can be enabled or disabled per route table
- Functionality
  - Automatically updates route tables with routes learned from BGP peers
  - Propagates the best available routes based on BGP attributes
  - Maintains route consistency across specified route tables
  - Updates routes dynamically when network conditions change
- States
  - Can be enabled (routes are being propagated)
  - Can be disabled (routes are not being propagated)

AWS Management Console

To enable route server propagation

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Select the route server for which you want to enable propagation.
3. Choose the **Propagations** tab in the route server details panel.
4. Choose **Enable propagation**.
5. In the **Enable propagation** dialog:
   - The **Route server ID** will be pre-populated.
   - Under **Route table**, select the destination route table from the dropdown menu for newly propagated routes.
6. Choose **Enable propagation** to confirm.
7. Wait for the propagation status to change to Available in the **Propagations** list.
8. Verify that the selected route table appears in the **Propagations** list with a state of *Available*.

Command line

Use the following procedure to enable route server propagation.

1. Command:

   ```
   aws ec2 enable-route-server-propagation --route-table-id rtb-1 --route-server-id
    rs-1
   ```

   Response:

   ```
   {
       "RouteServerRoutePropagation": {
           "RouteServerId": "rs-1",
           "RouteTableId": "rtb-1",
           "State": "pending"
       }
   }
   ```
2. Wait for the propagation state to change to available.

Command:

```
aws ec2 get-route-server-propagations --route-server-id rs-1
```

Response:

```
{
    "RouteServerRoutePropagation": {
        "RouteServerId": "rs-1",
        "RouteTableId": "rtb-1",
        "State": "available"
    }
}
```

Step 6: Create route server peer

A route server peer is a session between a route server endpoint and the device deployed in AWS (such as a firewall appliance or other network security function running on an EC2 instance). The device must meet these requirements:

- Have an elastic network interface in the VPC
- Support BGP (Border Gateway Protocol)
- Can initiate BGP sessions

> **Note**
>
> We recommend you create one route server peer per route server endpoint for redundancy.

AWS Management Console

To create a route server peer

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation path, choose **VPC** > **Route server peers** > **Create route server peer**.
3. Under **Details**, configure the following:

- **Name**: Enter a name for your route server peer (up to 255 characters). Example: my-route-server-peer-01
- **Route server endpoint ID**: Choose a route server endpoint from the dropdown. Optionally, choose **Create a route server endpoint** to create a new one.
- **Peer address**: Enter the IPv4 address of the peer. Must be a valid IP address. The peer address must be reachable from the route server endpoint.
- **Peer ASN**: Enter the ASN (Autonomous System Number) for the BGP peer. Value must be in range of 1-4294967295. The ASN should typically use private ranges (64512-65534 for 16-bit or 4200000000-4294967294 for 32-bit)
- **Peer liveness detection**:
  - **BGP keepalive** (default): Standard BGP keep alive mechanism
  - **BFD**: Bidirectional Forwarding Detection for faster failover
- (Optional) Under **Tags**, choose **Add new tag** to add key-value pair tags. Tags help identify and track AWS resources.

4. Review your settings and choose **Create route server peer**.

Command line

Use the following procedure to create a route server peer.

1. Command:

```
aws ec2 create-route-server-peer --route-server-endpoint-id rse-1 --peer-address
 10.0.2.3 --bgp-options PeerAsn=65001,PeerLivenessDetection=bfd
```

Response:

In the response, the state values can be `pending|available|deleting|deleted`.

```
{
    "RouteServerPeer": {
        "RouteServerPeerId": "rsp-1",
        "RouteServerId": "rs-1",
        "VpcId": "vpc-1",
        "SubnetId": "subnet-1",
        "State": "pending",
        "EndpointEniId": "eni-2,
```

```
        "EndpointEniAddress": "10.0.2.4",
        "PeerEniId": "eni-1",
        "PeerAddress": "10.0.2.3",
        "BgpOptions": {
            "PeerAsn": 65001,
   "PeerLivenessDetection": "bfd"
        },
        "BgpStatus": {
            "Status": "Up"
        }
    }
}
```

2. Wait for the propagation state to change to available.

   Command:

```
aws ec2 describe-route-server-peers
```

   Response:

```
{
    "RouteServerPeer": {
        "RouteServerPeerId": "rsp-1",
        "RouteServerId": "rs-1",
        "VpcId": "vpc-1",
        "SubnetId": "subnet-1",
        "State": "available",
        "EndpointEniId": "eni-2,
        "EndpointEniAddress": "10.0.2.4",
        "PeerEniId": "eni-1",
        "PeerAddress": "10.0.2.3",
        "BgpOptions": {
            "PeerAsn": 65001,
   "PeerLivenessDetection": "bfd"
        },
        "BgpStatus": {
            "Status": "down"
        }
    }
}
```

Step 7: Initiate BGP sessions from the devices

When the status of route server peer is available, configure your workload to initiate the BGP session with the route server endpoint.

Initiating a BGP session from the devices in your subnets is outside the scope of this guide. The route server endpoint does not initiate the BGP session.

You can check that the VPC Route Server feature is working by verifying that the route table contains the best routes propagated by route server.

Step 8: Cleanup

The building portion of the tutorial is complete. Complete the steps in this section to remove the VPC Route Server components that you created.

7.1: Withdraw BGP advertisement on the devices

Withdrawing BGP advertisement on the devices in your subnets is outside the scope of this guide. Refer to your third-party vendor for your BGP configurations as needed.

7.2: Disable route server propagation

Use the following procedure to disable route server propagation.

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Select the route server for which you want to disable propagation.
3. Choose **Actions > Modify route server**.
4. Choose the **Propagations** tab in the route server details panel.
5. Choose the propagation you want to disable and then choose **Disable propagation**.
6. In the dialog box, choose **Disable route server propagation**.

Command line

1. Disable propagation:

```
aws ec2 disable-route-server-route-propagation --route-table-id rtb-1 --route-server-id rs-1
```

2. Confirm that the propagation has been deleted:

```
aws ec2 get-route-server-route-propagations --route-server-id rs-1 [--route-
table-id rtb-1]
```

7.3: Delete route server peers

Use the following procedure to delete route server peers.

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation path, choose **Route servers** > **Route server peers**.
3. Select a route server peer.
4. Choose **Actions** > **Delete route server peer**.

Command line

1. Delete peers:

```
aws ec2 delete-route-server-peer --route-server-peer-id rsp-1
```

2. Confirm the deletion:

```
aws ec2 describe-route-server-peers [--route-server-peer-ids rsp-1] [--filters
 Key=RouteServerId|RouteServerEndpointId|VpcId]
```

7.4: Delete route server endpoints

Use the following procedure to delete route server endpoints.

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Select the route server for which you want to delete endpoints.
3. Choose **Route server endpoints**.

4. Select the endpoint and choose **Actions** > **Delete route server endpoint**.
5. Enter delete and choose **Delete**.

Command line

1. Describe endpoints:

```
aws ec2 describe-route-server-endpoints
```

2. Delete route server endpoints:

```
aws ec2 delete-route-server-endpoint --route-server-endpoint-id rse-1
```

3. Confirm that the endpoints have been deleted:

```
aws ec2 describe-route-server-endpoints [--route-server-endpoint-ids rsp-1] [--
filters Key=RouteServerId|VpcId|SubnetId]
```

7.5: Disassociate route server from VPC

Use the following procedure to disassociate the route server from the VPC.

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Select the route server for which you want to disassociate.
3. Choose **Association**.
4. Choose **Disassociate route server**.
5. Confirm the changes that will be made and choose **Disassociate route server**.

Command line

1. Disassociate route server from the VPC:

```
aws ec2 disassociate-route-server --route-server-id rs-1 --vpc-id vpc-1
```

2. Confirm the disassociation:

```
aws ec2 get-route-server-associations --route-server-id rs-1
```

7.6 Delete route server

Use the following procedure to delete the route server.

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Select the route server to delete.
3. Choose **Actions** > **Delete route server**.
4. Enter *delete* and choose **Delete**.

Command line

1. Delete route server:

```
aws ec2 delete-route-server --route-server-id rs-1
```

2. Confirm the deletion:

```
aws ec2 describe-route-servers [--route-server-ids rs-1] [--filters Key=VpcId]
```

The Amazon VPC Route Server tutorial is complete.

Troubleshoot reachability issues in your VPC

Reachability Analyzer is a static configuration analysis tool. Use Reachability Analyzer to analyze and debug network reachability between two resources in your VPC. Reachability Analyzer produces hop-by-hop details of the virtual path between these resources when they are reachable, and identifies the blocking component otherwise. For example, it can identify missing or misconfigured route table routes.

For more information, see the Reachability Analyzer Guide.

Middlebox routing wizard

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

The middlebox routing wizard can help you configure routing for the following scenarios:

- Routing traffic to a middlebox appliance, for example, an Amazon EC2 instance that's configured as a security appliance.
- Routing traffic to a Gateway Load Balancer. For more information, see the User Guide for Gateway Load Balancers.

For more information, see the section called "Middlebox scenarios".

Contents

- Middlebox routing wizard prerequisites
- Redirect VPC traffic to a security appliance
- Middlebox routing wizard considerations
- Middlebox scenarios

Middlebox routing wizard prerequisites

Review the section called "Middlebox routing wizard considerations". 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.

Redirect VPC traffic to a security appliance

The middlebox routing wizard is available in the Amazon Virtual Private Cloud Console.

Contents

- 1. Create routes using the middlebox routing wizard
- 2. Modify middlebox routes
- 3. Delete the middlebox routing wizard configuration

1. 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**.

2. 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**.

3. 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.

   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.

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*.
- The route tables and routes that the middlebox routing wizard creates count toward your quotas. For more information, see the section called "Route tables".
- If you delete a resource, for example a network interface, the route table associations with the resource are removed. If the resource is a target, the route destination is set to blackhole. The route tables are not deleted.
- The middlebox subnet and the destination subnet must be associated with a non-default route table.

> **Note**
>
> We recommend that you use the middlebox routing wizard to modify or delete any route tables that you created using the middlebox routing wizard.

- If you use middlebox routing to route through a security appliance, security group referencing between the source and ultimate destination after inspection is not supported.

Middlebox scenarios

Amazon Virtual Private Cloud (VPC) provides a wide range of networking capabilities that allow you to customize and control the routing of traffic within your virtual network. One such feature is the middlebox routing wizard, which enables fine-grained control over the routing path of traffic entering or leaving your VPC.

If you need to redirect traffic to a security appliance, load balancer, or other network device for inspection, monitoring, or optimization purposes, the middlebox routing wizard can simplify the process. This wizard automatically creates the necessary route tables and routes (hops) to redirect the specified traffic as needed, eliminating the manual effort required to set up complex routing configurations.

The middlebox routing wizard supports several different scenarios. For example, you can use it to inspect traffic destined for a particular subnet, configure middlebox traffic routing and inspection across your entire VPC, or selectively inspect traffic between specific subnets. This granular control over traffic routing allows you to implement advanced security policies, enable centralized network monitoring, or optimize the performance of your cloud-based applications.

The following examples describe scenarios for the middlebox routing wizard.

Contents

- Inspect traffic destined for a subnet
- Configure middlebox traffic routing and inspection in a VPC
- Inspect traffic between subnets

Inspect traffic destined for a subnet

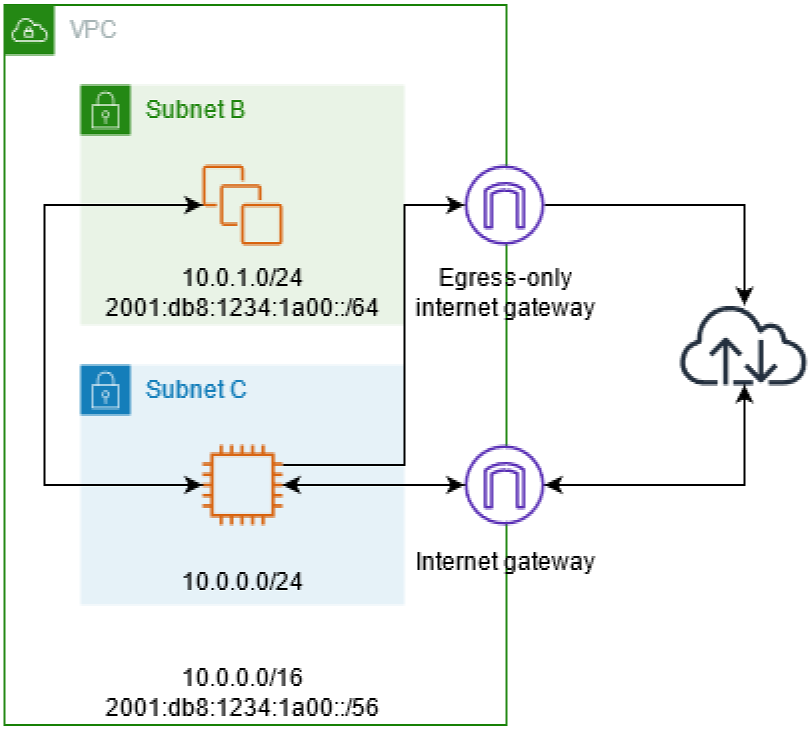
Consider the scenario where you have traffic coming into the VPC through an internet gateway and you want to inspect all traffic that is destined for a subnet, say subnet B, using a firewall appliance installed on an EC2 instance. The firewall appliance should be installed and configured on an EC2 instance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

The middlebox routing wizard, automatically performs the following operations:

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

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

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



Internet gateway route table

Add the following routes to the route table for the internet gateway.

Destination	Target	Purpose
10.0.0.0/16	Local	Local route for IPv4
10.0.1.0/24	*appliance-eni*	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	*appliance-eni*	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, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"

- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Destination subnet route table

Add the following routes to the route table for the destination subnet (subnet B in the example diagram).

Destination	Target	Purpose
10.0.0.0/16	Local	Local route for IPv4
0.0.0.0/0	*appliance-eni*	Route IPv4 traffic destined for the internet to the middlebox
2001:db8:1234:1a00::/56	Local	Local route for IPv6
::/0	*appliance-eni*	Route IPv6 traffic destined for the internet to the middlebox

There is a subnet association with the middlebox subnet.

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Middlebox subnet route table

Add the following routes to the route table for the middlebox subnet (subnet C in the example diagram).

Destination	Target	Purpose
10.0.0.0/16	Local	Local route for IPv4

Destination	Target	Purpose
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 IPv6 traffic to the egress-only internet gateway

There is a subnet association with the destination subnet.

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Configure middlebox traffic routing and inspection in a VPC

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

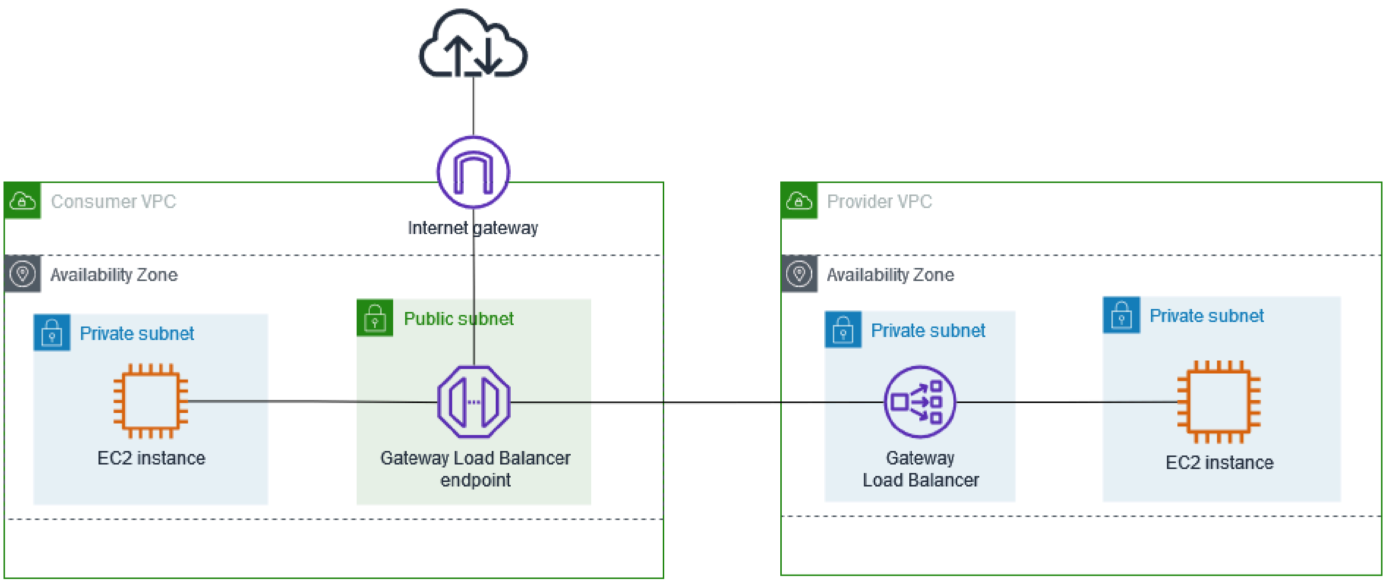
The middlebox routing wizard automatically performs the following operations:

- Creates the route tables.
- Adds the necessary routes to the new route tables.
- Disassociates the current route tables associated with the subnets.
- Associates the route tables that the middlebox routing wizard creates with the subnets.

- Creates a tag that indicates it was created by the middlebox routing wizard, and a tag that indicates the creation date.

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

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



Internet gateway route table

The route table for the internet gateway has the following routes.

Destination	Target	Purpose
Consumer VPC CIDR	Local	Local route
Application subnet CIDR	*endpoint-id*	Routes traffic destined for the application subnet to the Gateway Load Balancer endpoint

There is an edge association with the gateway.

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Application subnet route table

The route table for the application subnet has the following routes.

Destination	Target	Purpose
Consumer VPC CIDR	Local	Local route
0.0.0.0/0	*endpoint-id*	Route traffic from the application servers to the Gateway Load Balancer endpoint before it is routed to the internet

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Provider subnet route table

The route table for the provider subnet has the following routes.

Destination	Target	Purpose
Provider VPC CIDR	Local	Local route. Ensures that traffic originati ng from the internet is routed to the application servers
0.0.0.0/0	*igw-id*	Routes all traffic to the internet gateway

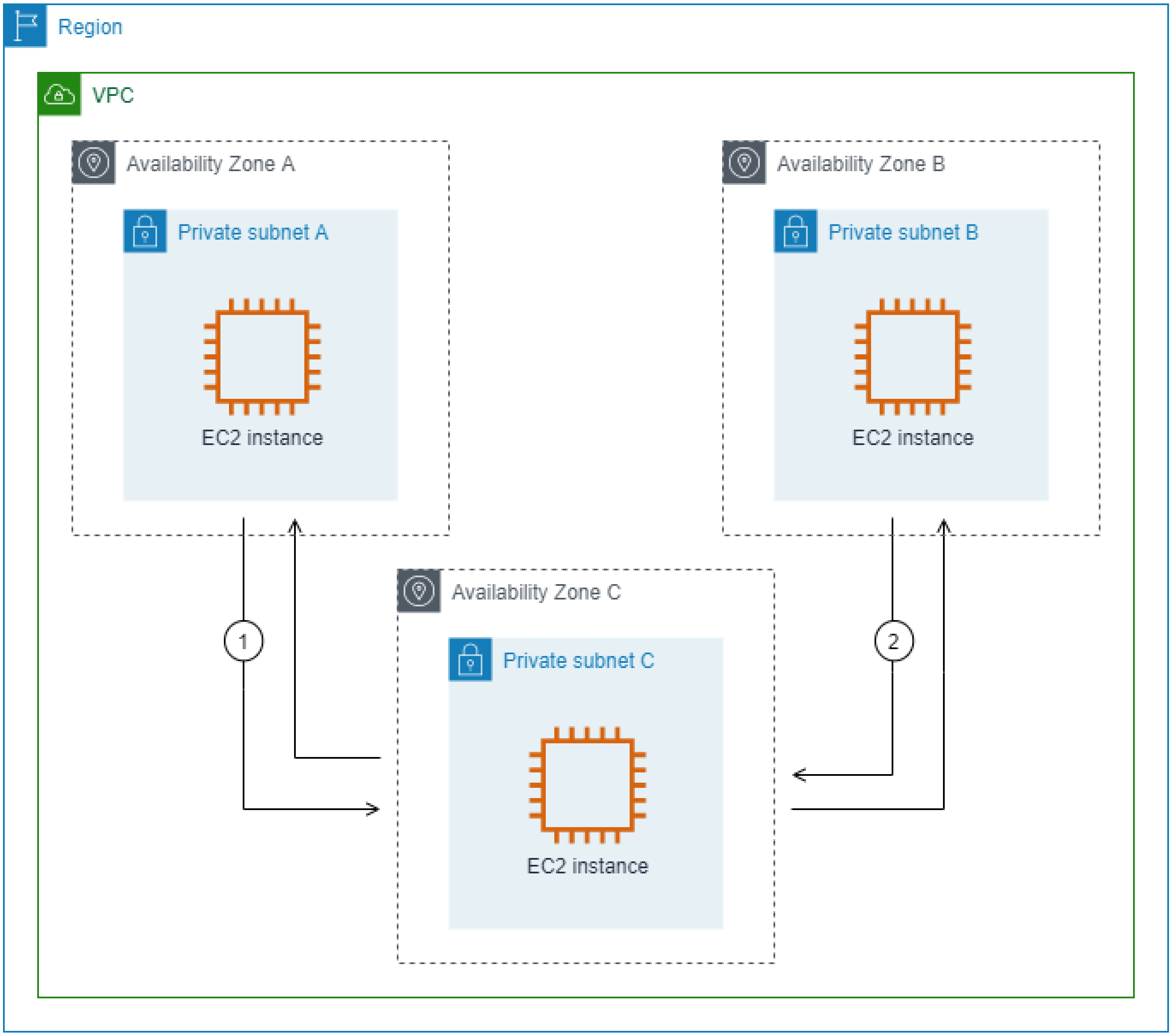
When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Inspect traffic between subnets

Consider the scenario where you have multiple subnets in a VPC and you want to inspect the traffic between them using a firewall appliance. Configure and install the firewall appliance on an EC2 instance in a separate subnet in your VPC.

The following diagram shows a firewall appliance installed on an EC2 instance in subnet C. The appliance inspects all traffic that travels from subnet A to subnet B (see 1) and from subnet B to subnet A (see 2).



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

The middlebox routing wizard, automatically performs the following operations:

- Creates the route tables.
- Adds the necessary routes to the new route tables.
- Disassociates the current route tables associated with the subnets.
- Associates the route tables that the middlebox routing wizard creates with the subnets.

- Creates a tag that indicates it was created by the middlebox routing wizard, and a tag that indicates the creation date.

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

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

Custom route table for subnet A

The route table for subnet A has the following routes.

Destination	Target	Purpose
VPC CIDR	Local	Local route
Subnet B CIDR	*appliance-eni*	Route traffic destined for subnet B to the middlebox

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Custom route table for subnet B

The route table for subnet B has the following routes.

Destination	Target	Purpose
VPC CIDR	Local	Local route

Destination	Target	Purpose
Subnet A CIDR	*appliance-eni*	Route traffic destined for subnet A to the middlebox

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Main route table

Subnet C uses the main route table. The main route table has the following route.

Destination	Target	Purpose
VPC CIDR	Local	Local route

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Delete a subnet

If you no longer need a subnet, you can delete it. You cannot delete a subnet if it contains any network interfaces. For example, you must terminate any instances in a subnet before you can delete it.

When you delete a subnet, the CIDR block associated with that subnet is returned to the VPC's available IP address pool. This means that the IP addresses within the subnet's CIDR range can be reallocated to other subnets or resources within the same VPC.

It's important to note that deleting a subnet does not automatically delete the resources within it. You must first terminate any EC2 instances, delete any network interfaces, and remove any other resources associated with the subnet before you can proceed with the subnet deletion.

To delete a subnet using the console

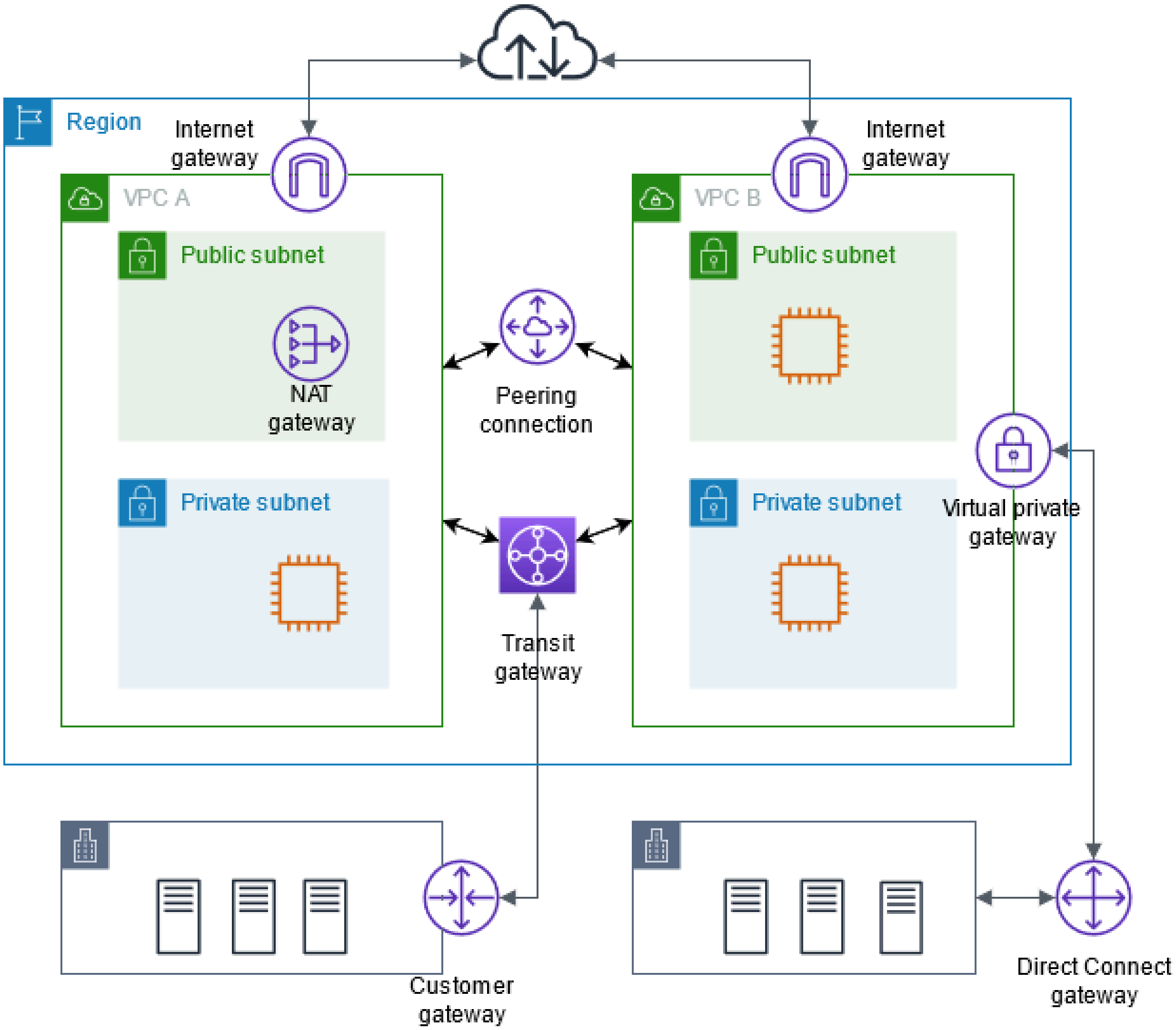
1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Terminate all instances in the subnet. For more information, see Terminate your instance in the *Amazon EC2 User Guide*.
3. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
4. In the navigation pane, choose **Subnets**.
5. Select the subnet and choose **Actions**, **Delete subnet**.
6. When prompted for confirmation, type **`delete`** and then choose **Delete**.

To delete a subnet using the AWS CLI

Use the delete-subnet command.

Connect your VPC to other networks

You can connect your virtual private cloud (VPC) to other networks, such as other VPCs, the internet, or your on-premises network.



You can connect your virtual private cloud (VPC) to other networks, such as other VPCs, the internet, or your on-premises network.

The diagram demonstrates some of these connectivity options. VPC A is connected to the internet through an internet gateway, and the EC2 instance in the private subnet can connect to the internet using a NAT gateway in the public subnet. VPC B is also connected to the internet, but through a direct internet gateway, allowing the EC2 instance in the public subnet to access the internet.

Moreover, VPC A and VPC B are connected to each other through a VPC peering connection and a transit gateway. The transit gateway has a VPN attachment to a data center, and VPC B has an AWS Direct Connect connection to the same data center. This interconnectivity enables organizations to integrate their cloud resources with on-premises infrastructure, creating a hybrid cloud environment.

Connecting VPCs to other networks is an important aspect of building cloud infrastructure within AWS. It offers organizations flexibility and control over their networking configurations, allowing them to design VPC architectures that align with their business requirements and security needs. These connectivity options facilitate efficient data flow between various components of a distributed IT landscape, whether they are within the cloud or on-premises.

AWS provides a range of tools and features to enable these VPC connections, including internet gateways, NAT gateways, VPC peering, transit gateways, and AWS Direct Connect. By leveraging these capabilities, organizations can create secure and integrated cloud environments that seamlessly integrate with their existing IT infrastructure.

You can connect your virtual private cloud (VPC) to other networks. For example, other VPCs, the internet, or your on-premises network.

For more information, see Amazon Virtual Private Cloud Connectivity Options.

Contents

- Enable internet access for a VPC using an internet gateway
- Enable outbound IPv6 traffic using an egress-only internet gateway
- Connect to the internet or other networks using NAT devices
- Associate Elastic IP addresses with resources in your VPC
- Connect your VPC to other VPCs and networks using a transit gateway
- Connect your VPC to remote networks using AWS Virtual Private Network
- Connect VPCs using VPC peering

Enable internet access for a VPC using an internet gateway

An internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between your VPC and the internet. It supports IPv4 and IPv6 traffic. It does not cause availability risks or bandwidth constraints on your network traffic.

An internet gateway enables resources in your public subnets (such as EC2 instances) to connect to the internet if the resource has a public IPv4 address or an IPv6 address. Similarly, resources on the internet can initiate a connection to resources in your subnet using the public IPv4 address or IPv6 address. For example, an internet gateway enables you to connect to an EC2 instance in AWS using your local computer.

An internet gateway provides a target in your VPC route tables for internet-routable traffic. For communication using IPv4, the internet gateway also performs network address translation (NAT). For more information, see IP addresses and NAT.

Pricing

There is no charge for an internet gateway, but there are data transfer charges for EC2 instances that use internet gateways. For more information, see Amazon EC2 On-Demand Pricing.

Contents

- Internet gateway basics
- Add internet access to a subnet
- Delete an internet gateway

Internet gateway basics

To use an internet gateway, you must attach it to a VPC and configure routing.

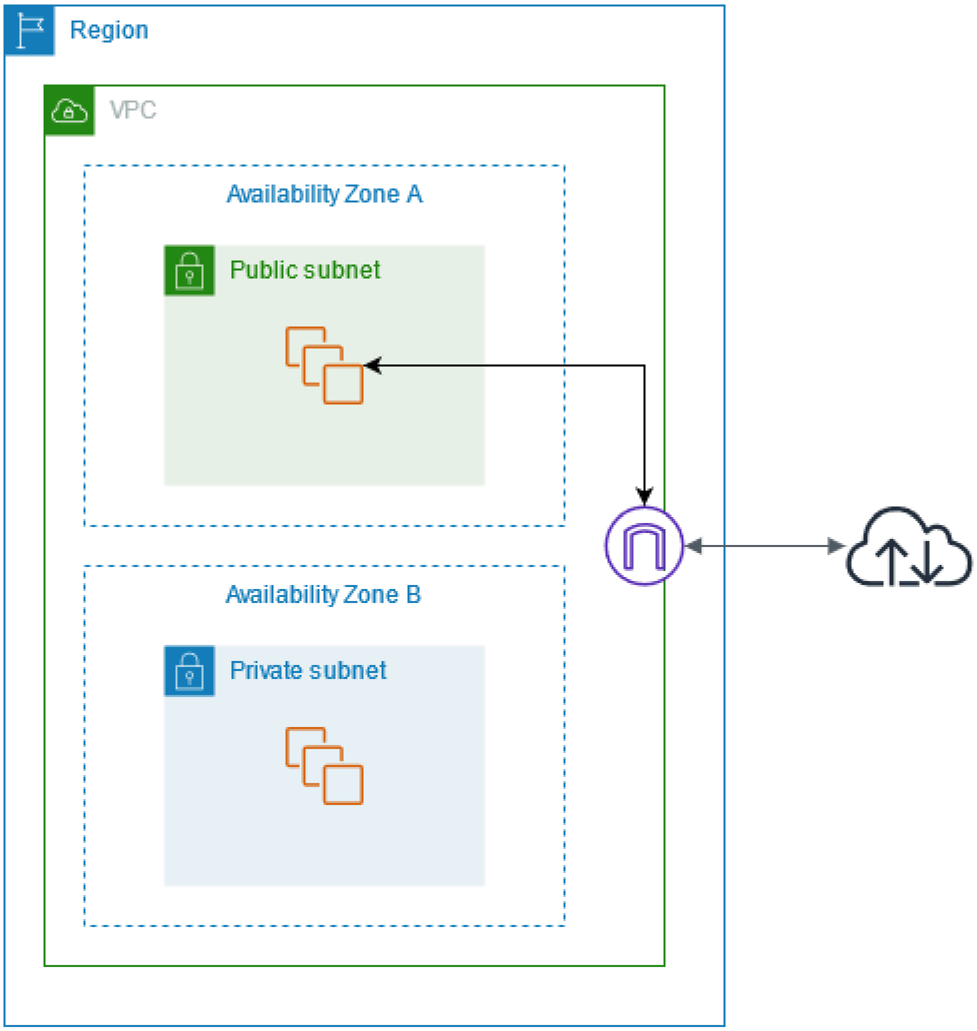
Routing configuration

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.

Internet gateway diagram

In the following diagram, the subnet in Availability Zone A is a public subnet because its route table has a route that sends all internet-bound IPv4 traffic to the internet gateway. The instances in the public subnet must have public IP addresses or Elastic IP addresses to enable communication with the internet over the internet gateway. For comparison, the subnet in Availability Zone B is a private subnet because its route table does not have a route to the internet gateway. Because there is no route to the internet gateway, instances in the private subnet can't communicate with the internet, even if they have public IP addresses.



IP addresses and NAT

To enable communication over the internet for IPv4, your instance must have a public IPv4 address. You can either configure your VPC to automatically assign public IPv4 addresses to your instances, or you can assign Elastic IP addresses to your instances. Your instance is only aware of the private (internal) IP address space defined within the VPC and subnet. The internet gateway logically provides the one-to-one NAT on behalf of your instance, so that when traffic leaves your VPC subnet and goes to the internet, the reply address field is set to the public IPv4 address or Elastic IP address of your instance, and not its private IP address. Conversely, traffic that's destined for the public IPv4 address or Elastic IP address of your instance has its destination address translated into the instance's private IPv4 address before the traffic is delivered to the VPC.

To enable communication over the internet for IPv6, your VPC and subnet must have an associated IPv6 CIDR block, and your instance must be assigned an IPv6 address from the range of the subnet. IPv6 addresses are globally unique, and therefore public by default.

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	No
Route table with route to internet gateway for IPv4 traffic (0.0.0.0/0)	Yes	No
Route table with route to internet gateway for IPv6 traffic (::/0)	No	No
Public IPv4 address automatic ally 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)

Add internet access to a subnet

The following describes how to support internet access from a subnet in a nondefault VPC using an internet gateway. You must create the internet gateway, attach it to the VPC, and configure routing for the subnet.

After you configure internet access for your subnet, you must ensure that resources in the subnet can access the internet. For example, your EC2 instances must have a public IPv4 or IPv6 address, and the security groups for your instances must allow specific traffic to and from the internet.

Alternatively, to provide your instances with internet access without assigning them a public IP address, use a NAT device instead. For more information, see NAT devices.

To remove internet access, you can detach the internet gateway from your VPC and then delete it. For more information, see the section called "Delete an internet gateway".

Tasks

- Step 1: Create an internet gateway
- Step 2: Attach the internet gateway to the VPC
- Step 3: Add a route to the subnet route table

Step 1: Create an internet gateway

Use the following procedure to create an internet gateway.

To create an internet gateway using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Internet gateways**.
3. Choose **Create internet gateway**.
4. (Optional) Enter a name for your internet gateway.
5. (Optional) To add a tag, choose **Add new tag** and enter the tag key and value.
6. Choose **Create internet gateway**.
7. (Optional) To attach the internet gateway to a VPC now, choose **Attach to a VPC** from the banner at the top of the screen, select an available VPC, and then choose **Attach internet gateway**. Otherwise, you can attach your internet gateway to a VPC at another time.

To create an internet gateway using the command line

- create-internet-gateway (AWS CLI)
- New-EC2InternetGateway (AWS Tools for Windows PowerShell)

Step 2: Attach the internet gateway to the VPC

To use an internet gateway, you must attach it to a VPC.

To attach an internet gateway to a VPC using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Internet gateways**.
3. Select the check box for the internet gateway.
4. To attach it, choose **Actions**, Attach to VPC, select an available VPC, and choose **Attach internet gateway**.
5. To detach it, choose **Actions**, **Detach from VPC** and choose **Detach internet gateway**. When prompted for confirmation, choose **Detach internet gateway**.

To attach an internet gateway to a VPC using the command line

- attach-internet-gateway (AWS CLI)
- Add-EC2InternetGateway (AWS Tools for Windows PowerShell)

Step 3: Add a route to the subnet route table

The route table for the subnet must have a route that sends internet traffic to the internet gateway.

To configure the subnet 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 for the subnet. By default, a subnet uses the main route table for the VPC. Alternatively, you can create a route table and then associate the subnet with the new route table.
4. On the **Routes** tab, choose **Edit routes** and then choose **Add route**.
5. Enter 0.0.0.0/0 for **Destination** and select the internet gateway for **Target**.
6. Choose **Save changes**.

To configure the subnet route table using the command line

- create-route (AWS CLI)
- New-EC2Route (AWS Tools for Windows PowerShell)

Delete an internet gateway

If you no longer need internet access for a VPC, you can detach the internet gateway from the VPC and then delete it. You can't delete an internet gateway if it's still attached to 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 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 **Internet gateways**.
3. Select the check box for the internet gateway.
4. To attach it, choose **Actions**, Attach to VPC, select an available VPC, and choose **Attach internet gateway**.
5. To detach it, choose **Actions**, **Detach from VPC** and choose **Detach internet gateway**. When prompted for confirmation, choose **Detach internet gateway**.

To describe your internet gateways, including attachments, using the command line

- describe-internet-gateways (AWS CLI)
- Get-EC2InternetGateway (AWS Tools for Windows PowerShell)

To detach an internet gateway from a VPC using the command line

- detach-internet-gateway (AWS CLI)
- Dismount-EC2InternetGateway (AWS Tools for Windows PowerShell)

To delete an internet gateway using the console

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

3. Select the check box for the internet gateway.
4. Choose **Actions**, **Delete internet gateway**.
5. When prompted for confirmation, enter **`delete`**, and then choose **Delete internet gateway**.

To delete an internet gateway using the command line

- delete-internet-gateway (AWS CLI)
- Remove-EC2InternetGateway (AWS Tools for Windows PowerShell)

Enable outbound IPv6 traffic using an egress-only internet gateway

An egress-only internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows outbound communication over IPv6 from instances in your VPC to the internet, and prevents the internet from initiating an IPv6 connection with your instances.

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.

Pricing

There is no charge for an egress-only internet gateway, but there are data transfer charges for EC2 instances that use internet gateways. For more information, see Amazon EC2 On-Demand Pricing.

Contents

- Egress-only internet gateway basics
- Add egress-only internet access to a subnet

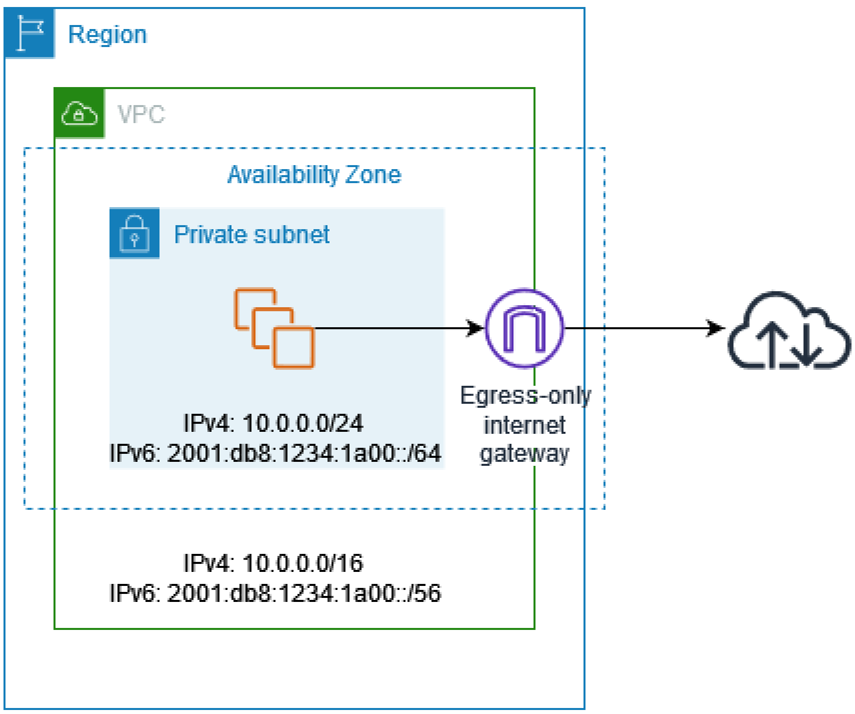
Egress-only internet gateway basics

IPv6 addresses are globally unique, and are therefore public by default. If you want your instance to be able to access the internet, but you want to prevent resources on the internet from initiating communication with your instance, you can use an egress-only internet gateway. To do this, create an egress-only internet gateway in your VPC, and then add a route to your route table that points all IPv6 traffic (`::/0`) or a specific range of IPv6 address to the egress-only internet gateway. IPv6 traffic in the subnet that's associated with the route table is routed to the egress-only internet gateway.

An egress-only internet gateway is stateful: it forwards traffic from the instances in the subnet to the internet or other AWS services, and then sends the response back to the instances.

You can't associate a security group with an egress-only internet gateway to control the traffic that is allowed to reach or leave the egress-only internet gateway. You can use a network ACL to control the traffic to and from the subnet for which the egress-only internet gateway routes traffic.

In the following diagram, the VPC has both IPv4 and IPv6 CIDR blocks, and the subnet both IPv4 and IPv6 CIDR blocks. The VPC has an egress-only internet gateway.



The following is an example of the route table associated with the subnet. There is a route that sends all internet-bound IPv6 traffic (::/0) to the egress-only internet gateway.

Destination	Target
10.0.0.0/16	Local
2001:db8:1234:1a00:/64	Local
::/0	*eigw-id*

Add egress-only internet access to a subnet

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

- 1. Create an egress-only internet gateway
- 2. Create a custom route table
- 3. Delete an egress-only internet gateway
- Command line overview

1. 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**.

2. 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 Configure route tables.

3. 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.

Command line overview

You can perform the tasks described on this page using the command line.

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)

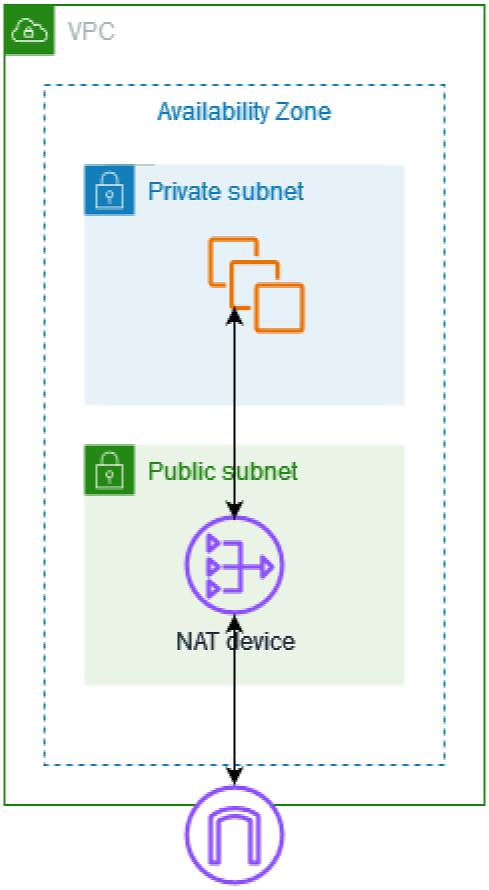
Delete an egress-only internet gateway

- delete-egress-only-internet-gateway (AWS CLI)
- Remove-EC2EgressOnlyInternetGateway (AWS Tools for Windows PowerShell)

Connect to the internet or other networks using NAT devices

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

For example, the following diagram shows a NAT device in a public subnet that allows the EC2 instances in a private subnet to connect to the internet through an internet gateway. 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.



> **Important**
>
> - 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).
> - 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.

Contents

- NAT gateways
- NAT instances
- Compare NAT gateways and NAT instances

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 can't 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 the instances can't 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, but the instances can't receive unsolicited inbound connections from the other VPCs or the on-premises network. You can route traffic from the NAT gateway through a transit gateway or a virtual private gateway. You can't 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.

A NAT gateway is for use with IPv4 or IPv6 traffic (using DNS64 and NAT64). Another option for enabling outbound-only internet communication over IPv6 is using an egress-only internet gateway.

Both private and public NAT gateways map the source private IPv4 address of the instances to the private IPv4 address of the NAT gateway, but in the case of a public NAT gateway, the internet gateway then maps the private IPv4 address of the public NAT gateway to the Elastic IP address associated with the NAT gateway. When sending response traffic to the instances, whether it's a public or private NAT gateway, the NAT gateway translates the address back to the original source IP address.

> **Important**
>
> Connections must always be initiated from within the VPC containing the NAT Gateway.

You can use either a public or private NAT gateway to route traffic to transit gateways and virtual private gateways.
If you use a private NAT gateway to connect to a transit gateway or virtual private gateway, traffic to the destination will come from the private IP address of the private NAT gateway.
If you use a public NAT gateway to connect to a transit gateway or virtual private gateway, traffic to the destination will come from the private IP address of the public NAT gateway. The public NAT gateway will only use its EIP as the source IP address when used in conjunction with an internet gateway in the same VPC.
NAT gateways support traffic with a maximum transmission unit (MTU) of 8500. For more information, see NAT gateway basics.

Contents

- NAT gateway basics
- Work with NAT gateways
- NAT gateway use cases
- DNS64 and NAT64
- Monitor NAT gateways with Amazon CloudWatch
- Troubleshoot NAT gateways
- Pricing for NAT gateways

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.

If you have resources in multiple Availability Zones and they share one NAT gateway, and if the NAT gateway's Availability Zone is down, resources in the other Availability Zones lose internet access. To improve resiliency, create a NAT gateway in each Availability Zone, and configure your routing to ensure that resources use the NAT gateway in the same Availability Zone.

The following characteristics and rules apply to NAT gateways:

- A NAT gateway supports the following protocols: TCP, UDP, and ICMP.

- NAT gateways are supported for IPv4 or IPv6 traffic. For IPv6 traffic, NAT gateway performs NAT64. By using this in conjunction with DNS64 (available on Route 53 resolver), your IPv6 workloads in a subnet in Amazon VPC can communicate with IPv4 resources. These IPv4 services may be present in the same VPC (in a separate subnet) or a different VPC, on your on-premises environment or on the internet.
- A NAT gateway supports 5 Gbps of bandwidth and automatically scales up to 100 Gbps. If you require more bandwidth, you can split your resources into multiple subnets and create a NAT gateway in each subnet.
- A NAT gateway can process one million packets per second and automatically scales up to ten million packets per second. Beyond this limit, a NAT gateway will drop packets. To prevent packet loss, split your resources into multiple subnets and create a separate NAT gateway for each subnet.
- Each IPv4 address can support up to 55,000 simultaneous connections to each unique destination. A unique destination is identified by a unique combination of destination IP address, the destination port, and protocol (TCP/UDP/ICMP). You can increase this limit by associating up to 8 IPv4 addresses to your NAT gateways (1 primary IPv4 address and 7 secondary IPv4 addresses). You are limited to associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see Elastic IP addresses.
- You can pick the private IPv4 address to assign to the NAT gateway or have it automatically assigned from the IPv4 address range of the subnet. The assigned private IPv4 address persists until you delete the private NAT gateway. You can't detach the private IPv4 address and you can't attach additional private IPv4 addresses.
- You can't associate a security group with a NAT gateway. You can associate security groups with your instances to control inbound and outbound traffic.
- You can use a network ACL to control the traffic to and from the subnet for your NAT gateway. NAT gateways use ports 1024–65535. For more information, see Control subnet traffic with network access control lists.
- A NAT gateway receives a network interface. You can pick the private IPv4 address to assign to the interface or have it automatically assigned from the IPv4 address range of the subnet. You can view the network interface for the NAT gateway using the Amazon EC2 console. For more information, see Viewing details about a network interface. You can't modify the attributes of this network interface.
- You can't route traffic to a NAT gateway through a VPC peering connection.

- You can't route traffic to a NAT gateway from Site-to-Site VPN or Direct Connect using a virtual private gateway. You can route traffic to a NAT gateway from Site-to-Site VPN or Direct Connect if you use a transit gateway instead of a virtual private gateway.
- NAT gateways support traffic with a maximum transmission unit (MTU) of 8500, but it's important to note the following:
  - The MTU of a network connection is the size, in bytes, of the largest permissible packet that can be passed over the connection. The larger the MTU of a connection, the more data that can be passed in a single packet.
  - Packets larger than 8500 bytes that arrive at the NAT gateway are dropped (or fragmented, if applicable).
  - To prevent potential packet loss when communicating with resources over the internet using a public NAT gateway, the MTU setting for your EC2 instances should not exceed 1500 bytes. For more information about checking and setting the MTU on an instance, see Check and set the MTU on your Linux instance in the *Amazon EC2 User Guide*.
  - NAT gateways support Path MTU Discovery (PMTUD) via FRAG_NEEDED ICMPv4 packets and Packet Too Big (PTB) ICMPv6 packets.
  - NAT gateways enforce Maximum Segment Size (MSS) clamping for all packets. For more information, see RFC879.

Work with NAT gateways

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

Tasks

- Control the use of NAT gateways
- Create a NAT gateway
- Edit secondary IP address associations
- Tag a NAT gateway
- Delete a NAT gateway
- Command line overview

Control the use of NAT gateways

By default, users do not have permission to work with NAT gateways. You can create an IAM role with a policy attached that grants users permissions to create, describe, and delete NAT gateways. For more information, see Identity and access management for Amazon VPC.

Create a NAT gateway

Use the following procedure to create a NAT gateway.

Related quotas

- You won't be able to create a public NAT gateway if you've exhausted the number of EIPs allocated to your account. For more information on EIP quotas and how to adjust them, see Elastic IP addresses.
- You can assign up to 8 private IPv4 addresses to your private NAT gateway. This limit is not adjustable.
- You are limited to associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see Elastic IP addresses.

To create a NAT gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **NAT gateways**.
3. Choose **Create NAT gateway**.
4. (Optional) Specify a name for the NAT gateway. This creates a tag where the key is **Name** and the value is the name that you specify.
5. Select the subnet in which to create the NAT gateway.
6. For **Connectivity type**, leave the default **Public** selection to create a public NAT gateway or choose **Private** to create a private NAT gateway. For more information about the difference between a public and private NAT gateway, see NAT gateways.
7. If you chose **Public**, do the following; otherwise, skip to step 8:

    1. Choose an **Elastic IP allocation ID** to assign an EIP to the NAT gateway or choose **Allocate Elastic IP** to automatically allocate an EIP for the public NAT gateway. You are limited to

associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see Elastic IP addresses.

> **Important**
>
> When you assign an EIP to a public NAT gateway, the network border group of the EIP must match the network border group of the Availability Zone (AZ) that you're launching the public NAT gateway into. If it's not the same, the NAT gateway will fail to launch. You can see the network border group for the subnet's AZ by viewing the details of the subnet. Similarly, you can view the network border group of an EIP by viewing the details of the EIP address. For more information about network border groups and EIPs, see 1. Allocate an Elastic IP address.

2. (Optional) Choose **Additional settings** and, under **Private IP address - optional**, enter a private IPv4 address for the NAT gateway. If you don't enter an address, AWS will automatically assign a private IPv4 address to your NAT gateway at random from the subnet that your NAT gateway is in.
3. Skip to step 11.

8. If you chose **Private**, for **Additional settings**, **Private IPv4 address assigning method**, choose one of the following:

   - **Auto-assign**: AWS chooses the primary private IPv4 address for the NAT gateway. For **Number of auto-assigned private IPv4 addresses**, you can optionally specify the number of secondary private IPv4 addresses for the NAT gateway. AWS chooses these IP addresses at random from the subnet for your NAT gateway.
   - **Custom**: For **Primary private IPv4 address**, choose the primary private IPv4 address for the NAT gateway. For **Secondary private IPv4 addresses**, you can optionally specify up to 7 secondary private IPv4 addresses for the NAT gateway.

9. If you chose **Custom** in Step 8, skip this step. If you chose **Auto-assign**, under **Number of auto-assigned private IP addresses**, choose the number of secondary IPv4 addresses that you want AWS assign to this private NAT gateway. You can choose up to 7 IPv4 addresses.

> **Note**
>
> Secondary IPv4 addresses are optional and should be assigned or allocated when your workloads that use a NAT gateway exceed 55,000 concurrent connections to a single destination (the same destination IP, destination port, and protocol). Secondary IPv4

> addresses increase the number of available ports, and therefore they increase the limit on the number of concurrent connections that your workloads can establish using a NAT gateway.

10. If you chose **Auto-assign** in Step 9, skip this step. If you chose **Custom**, do the following:
    1. Under **Primary private IPv4 address**, enter a private IPv4 address.
    2. Under **Secondary private IPv4 address**, enter up to 7 secondary private IPv4 addresses.
11. (Optional) To add a tag to the NAT gateway, choose **Add new tag** and enter the key name and value. You can add up to 50 tags.
12. Choose **Create a NAT gateway**.
13. The initial status of the NAT gateway is `Pending`. After the status changes to `Available`, the NAT gateway is ready for you to use. Be sure to update your route tables as needed. For examples, see the section called “Use cases”.

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

Edit secondary IP address associations

Each IPv4 address can support up to 55,000 simultaneous connections to each unique destination. A unique destination is identified by a unique combination of destination IP address, the destination port, and protocol (TCP/UDP/ICMP). You can increase this limit by associating up to 8 IPv4 addresses to your NAT gateways (1 primary IPv4 address and 7 secondary IPv4 addresses). You are limited to associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see Elastic IP addresses.

You can use the NAT gateway CloudWatch metrics *ErrorPortAllocation* and *PacketsDropCount* to determine if your NAT gateway is generating port allocation errors or dropping packets. To resolve this issue, add secondary IPv4 addresses to your NAT gateway.

Considerations

- You can add secondary private IPv4 addresses when you create a private NAT gateway or after you create the NAT gateway using the procedure in this section. You can add secondary EIP addresses to public NAT gateways only after you create the NAT gateway by using the procedure in this section.

- Your NAT gateway can have up to 8 IPv4 addresses associated with it (1 primary IPv4 address and 7 secondary IPv4 addresses). You can assign up to 8 private IPv4 addresses to your private NAT gateway. You are limited to associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see [Elastic IP addresses](#).

To edit secondary IPv4 address associations

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **NAT gateways**.
3. Select the NAT gateway whose secondary IPv4 address associations you want to edit.
4. Choose **Actions**, and then choose **Edit secondary IP address associations**.
5. If you are editing the secondary IPv4 address associations of a private NAT gateway, under **Action**, choose **Assign new IPv4 addresses** or **Unassign existing IPv4 addresses**. If you are editing the secondary IPv4 address associations of a public NAT gateway, under **Action**, choose **Associate new IPv4 addresses** or **Disassociate existing IPv4 addresses**.
6. Do one of the following:
   - If you chose to assign or associate new IPv4 addresses, do the following:
     1. This step is required. You must select a private IPv4 address. Choose the **Private IPv4 address assigning method**:
        - **Auto-assign**: AWS automatically chooses a primary private IPv4 address and you choose if you want AWS to assign up to 7 secondary private IPv4 addresses to assign to the NAT gateway. AWS automatically chooses and assigns them for you at random from the subnet that your NAT gateway is in.
        - **Custom**: Choose the primary private IPv4 address and up to 7 secondary private IPv4 addresses to assign to the NAT gateway.
     2. Under **Elastic IP allocation ID**, choose an EIP to add as a secondary IPv4 address. This step is required. You must select an EIP along with a private IPv4 address. If you chose **Custom** for the **Private IP address assigning method**, you also must enter a private IPv4 address for each EIP that you add.

> **Important**
>
> When you assign a secondary EIP to a public NAT gateway, the network border group of the EIP must match the network border group of the Availability Zone (AZ) that the public NAT gateway is in. If it's not the same, the EIP will fail to assign. You can see the network border group for the subnet's AZ by viewing the details of the subnet. Similarly, you can view the network border group of an EIP by viewing the details of the EIP address. For more information about network border groups and EIPs, see 1. Allocate an Elastic IP address.

  Your NAT gateway can have up to 8 IP addresses associated with it. If this is a public NAT gateway, there is a default quota limit for EIPs per Region. For more information, see Elastic IP addresses.

- If you chose to unassign or disassociate new IPv4 addresses, complete the following:
  1. Under **Existing secondary IP address to unassign**, select the secondary IP addresses that you want to unassign.
  2. (optional) Under **Connection drain duration**, enter the maximum amount of time to wait (in seconds) before forcibly releasing the IP addresses if connections are still in progress. If you don't enter a value, the default value is 350 seconds.

7. Choose **Save changes**.

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

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*.

Cost allocation tags are supported for NAT gateways. Therefore, you can also use tags to organize your AWS bill and reflect your own cost structure. For more information, see Using cost allocation tags in the *AWS Billing User Guide*. For more information about setting up a cost allocation report with tags, see Monthly cost allocation report in *About AWS Account Billing*.

To tag a NAT gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **NAT gateways**.
3. Select the NAT gateway that you want to tag and choose **Actions**. Then choose **Manage tags**.
4. Choose **Add new tag**, and define a **Key** and **Value** for the tag. You can add up to 50 tags.
5. Choose **Save**.

Delete a NAT gateway

If you no longer need a NAT gateway, you can delete it. After you delete a NAT gateway, its entry remains visible in the Amazon VPC console for about an hour, after which it's automatically removed. You can't 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 5. Release an Elastic IP address.

Command line overview

You can perform the tasks described on this page using the command line.

Assign a private IPv4 address to a private NAT gateway

- assign-private-nat-gateway-address (AWS CLI)
- Register-EC2PrivateNatGatewayAddress (AWS Tools for Windows PowerShell)

Associate Elastic IP addresses (EIPs) and private IPv4 addresses with a public NAT gateway

- associate-nat-gateway-address (AWS CLI)
- Register-EC2NatGatewayAddress (AWS Tools for Windows PowerShell)

Create a NAT gateway

- create-nat-gateway (AWS CLI)
- New-EC2NatGateway (AWS Tools for Windows PowerShell)

Delete a NAT gateway

- delete-nat-gateway (AWS CLI)
- Remove-EC2NatGateway (AWS Tools for Windows PowerShell)

Describe a NAT gateway

- describe-nat-gateways (AWS CLI)
- Get-EC2NatGateway (AWS Tools for Windows PowerShell)

Disassociate secondary Elastic IP addresses (EIPs) from a public NAT gateway

- disassociate-nat-gateway-address (AWS CLI)
- Unregister-EC2NatGatewayAddress (AWS Tools for Windows PowerShell)

Tag a NAT gateway

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

Unassign secondary IPv4 addresses from a private NAT gateway

- unassign-private-nat-gateway-address (AWS CLI)
- Unregister-EC2PrivateNatGatewayAddress (AWS Tools for Windows PowerShell)

NAT gateway use cases

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

Scenarios

- Access the internet from a private subnet
- Access your network using allow-listed IP addresses
- Enable communication between overlapping networks

Access the internet from a private subnet

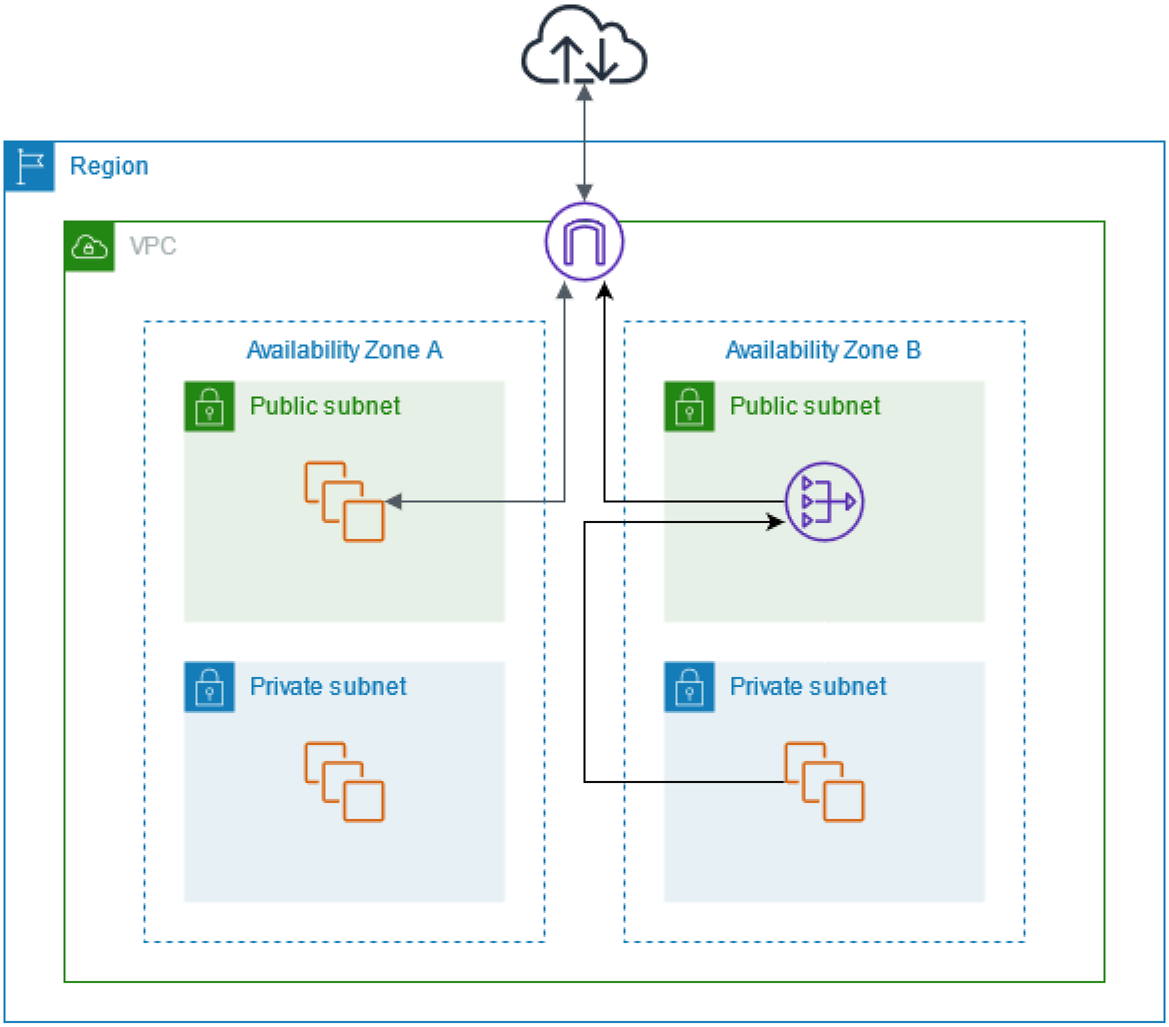
You can use a public NAT gateway to enable instances in a private subnet to send outbound traffic to the internet, while preventing the internet from establishing connections to the instances.

Contents

- Overview
- Routing
- Test the public NAT gateway

Overview

The following diagram illustrates this use case. There are two Availability Zones, with two subnets in each Availability Zone. The route table for each subnet determines how traffic is routed. In Availability Zone A, the instances in the public subnet can reach the internet through a route to the internet gateway, while the instances in the private subnet have no route to the internet. In Availability Zone B, the public subnet contains a NAT gateway, and the instances in the private subnet can reach the internet through a route to the NAT gateway in the public subnet. Both private and public NAT gateways map the source private IPv4 address of the instances to the private IPv4 address of the private NAT gateway, but in the case of a public NAT gateway, the internet gateway then maps the private IPv4 address of the public NAT gateway to the Elastic IP address associated with the NAT gateway. When sending response traffic to the instances, whether it's a public or private NAT gateway, the NAT gateway translates the address back to the original source IP address.



Note that if the instances in the private subnet in Availability Zone A also need to reach the internet, you can create a route from this subnet to the NAT gateway in Availability Zone B. Alternatively, you can improve resiliency by creating a NAT gateway in each Availability Zone that contains resources that require internet access. For an example diagram, see [the section called "Private servers"].

Routing

The following is the route table associated with the public subnet in Availability Zone A. The first entry is the local route; it enables the instances in the subnet to communicate with other instances in the VPC using private IP addresses. The second entry sends all other subnet traffic to the internet gateway, which enables the instances in the subnet to access the internet.

Destination	Target
VPC CIDR	local
0.0.0.0/0	*internet-gateway-id*

The following is the route table associated with the private subnet in Availability Zone A. The entry is the local route, which enables the instances in the subnet to communicate with other instances in the VPC using private IP addresses. The instances in this subnet have no access to the internet.

Destination	Target
VPC CIDR	local

The following is the route table associated with the public subnet in Availability Zone B. The first entry is the local route, which enables the instances in the subnet to communicate with other instances in the VPC using private IP addresses. The second entry sends all other subnet traffic to the internet gateway, which enables the NAT gateway in the subnet to access the internet.

Destination	Target
VPC CIDR	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 local route; it enables the instances in the subnet to communicate with other instances in the VPC using private IP addresses. The second entry sends all other subnet traffic to the NAT gateway.

Destination	Target
VPC CIDR	local
0.0.0.0/0	*nat-gateway-id*

For more information, see the section called “Manage subnet route tables”.

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.

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.

Test the internet connection

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

1. Launch an instance in your public subnet (use this as a bastion host). In the launch wizard, ensure that you select an Amazon Linux AMI, and assign a public IP address to your instance. Ensure that your security group rules allow inbound SSH traffic from the range of IP addresses for your local network, and outbound SSH traffic to the IP address range of your private subnet (you can also use `0.0.0.0/0` for both inbound and outbound SSH traffic for this test).
2. Launch an instance in your private subnet. In the launch wizard, ensure that you select an Amazon Linux AMI. Do not assign a public IP address to your instance. Ensure that your security group rules allow inbound SSH traffic from the private IP address of your instance that you launched in the public subnet, and all outbound ICMP traffic. You must choose the same key pair that you used to launch your instance in the public subnet.
3. Configure SSH agent forwarding on your local computer, and connect to your bastion host in the public subnet. For more information, see To configure SSH agent forwarding for Linux or macOS or To configure SSH agent forwarding for Windows.

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.

To configure SSH agent forwarding for Linux or macOS

1. From your local machine, add your private key to the authentication agent.

   For Linux, use the following command.

   ```
   ssh-add -c mykeypair.pem
   ```

   For macOS, use the following command.

   ```
   ssh-add -K mykeypair.pem
   ```

2. Connect to your instance in the public subnet using the `-A` option to enable SSH agent forwarding, and use the instance's public address, as shown in the following example.

   ```
   ssh -A ec2-user@54.0.0.123
   ```

To configure SSH agent forwarding for Windows

You can use the OpenSSH client available in Windows, or install your preferred SSH client (for example, PuTTY).

OpenSSH

Install OpenSSH for Windows as described in this article: Getting started with OpenSSH for Windows. Then add your key to the authentication agent. For more information, see Key-based authentication in OpenSSH for Windows.

PuTTY

1. Download and install Pageant from the PuTTY download page, if not already installed.
2. Convert your private key to .ppk format. For more information, see Convert your private key using PuTTYgen in the *Amazon EC2 User Guide*.

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 Connect to your Linux instance using PuTTY. 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.

3. (Optional) If you no longer require your instances, terminate them. For more information, see Terminate your instance in the *Amazon EC2 User Guide*.

Access your network using allow-listed IP addresses

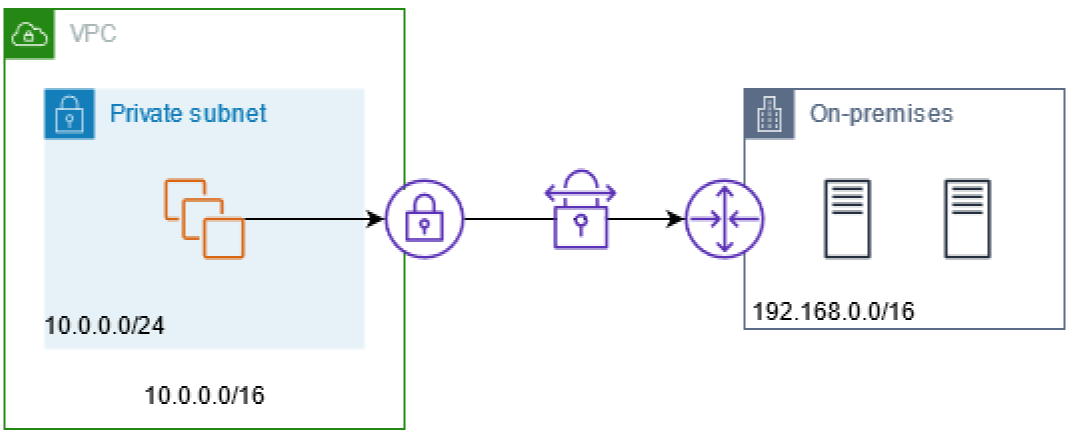
You can use a private NAT gateway to enable communication from your VPCs to your on-premises network using a pool of allow-listed addresses. Instead of assigning each instance a separate IP address from the allow-listed IP address range, you can route traffic from the subnet that is destined for the on-premises network through a private NAT gateway with an IP address from the allow-listed IP address range.

Contents

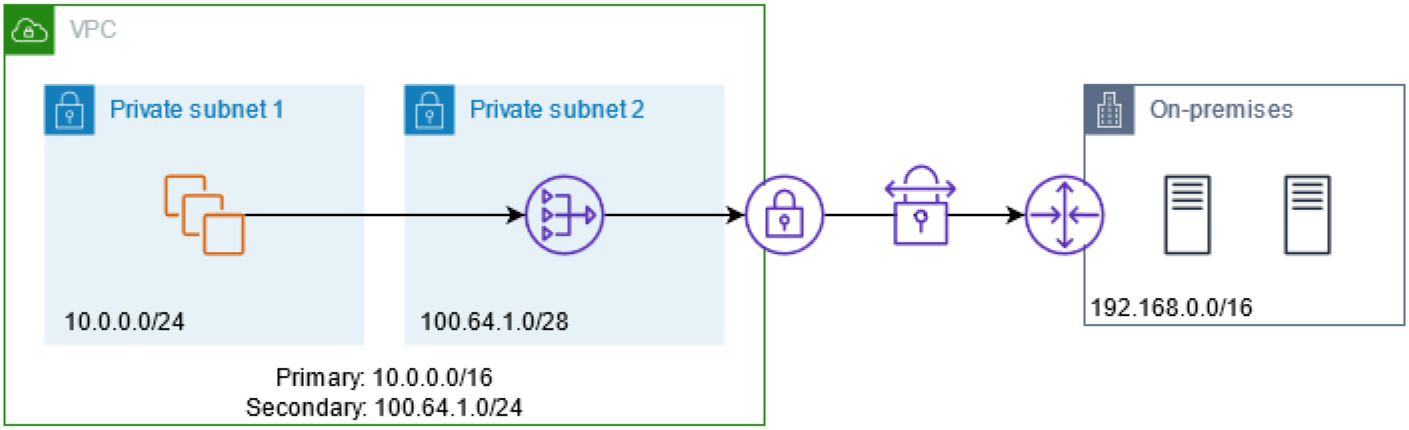
- Overview
- Resources
- Routing

Overview

The following diagram shows how instances can access on-premises resources through AWS VPN. Traffic from the instances is routed to a virtual private gateway, over the VPN connection, to the customer gateway, and then to the destination in the on-premises network. However, suppose that the destination allows traffic only from a specific IP address range, such as 100.64.1.0/28. This would prevent traffic from these instances from reaching the on-premises network.



The following diagram shows the key components of the configuration for this scenario. The VPC has its original IP address range plus the allowed IP address range. The VPC has a subnet from the allowed IP address range with a private NAT gateway. Traffic from the instances that is destined for the on-premises network is sent to the NAT gateway before being routed to the VPN connection. The on-premises network receives the traffic from the instances with the source IP address of the NAT gateway, which is from the allowed IP address range.



Resources

Create or update resources as follows:

- Associate the allowed IP address range with the VPC.
- Create a subnet in the VPC from the allowed IP address range.
- Create a private NAT gateway in the new subnet.
- Update the route table for the subnet with the instances to send traffic destined for the on-premises network to the NAT gateway. Add a route to the route table for the subnet with the private NAT gateway that sends traffic destined for the on-premises network to the virtual private gateway.

Routing

The following is the route table associated with the first subnet. There is a local route for each VPC CIDR. Local routes enable resources in the subnet to communicate with other resources in the VPC using private IP addresses. The third entry sends traffic destined for the on-premises network to the private NAT gateway.

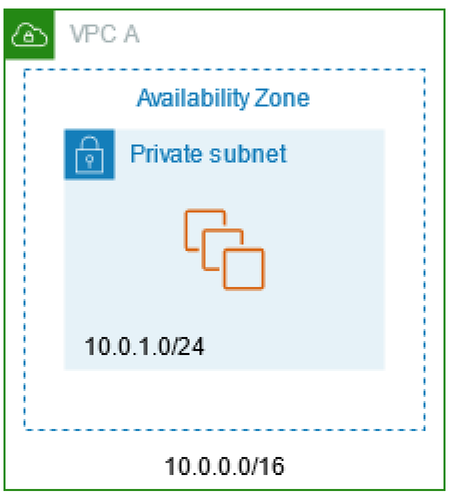
Destination	Target
10.0.0.0/16	local
100.64.1.0/24	local
192.168.0.0/16	*nat-gateway-id*

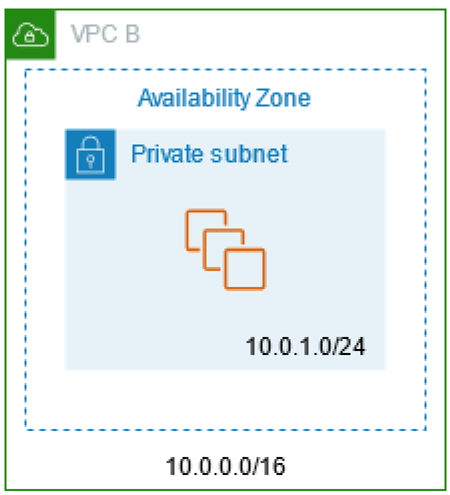
The following is the route table associated with the second subnet. There is a local route for each VPC CIDR. Local routes enable resources in the subnet to communicate with other resources in the VPC using private IP addresses. The third entry sends traffic destined for the on-premises network to the virtual private gateway.

Destination	Target
10.0.0.0/16	local
100.64.1.0/24	local
192.168.0.0/16	*vgw-id*

Enable communication between overlapping networks

You can use a private NAT gateway to enable communication between networks even if they have overlapping CIDR ranges. For example, suppose that the instances in VPC A need to access the services provided by the instances in VPC B.





Contents

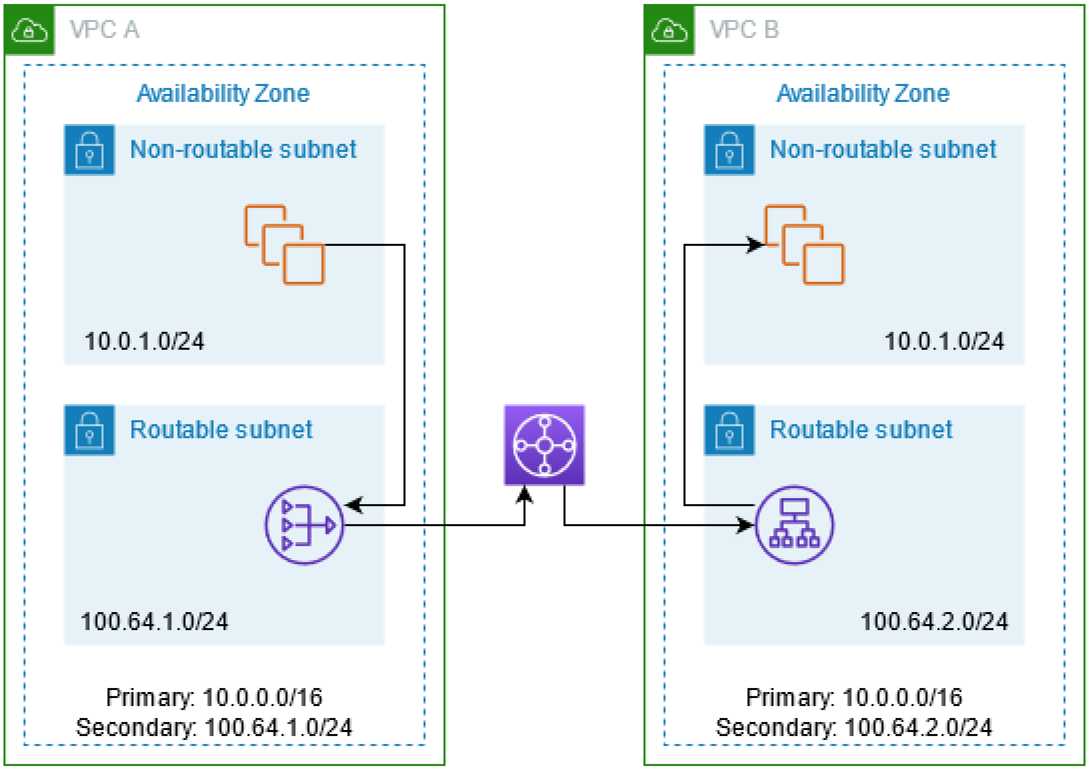
- Overview
- Resources
- Routing

Overview

The following diagram shows the key components of the configuration for this scenario. First, your IP management team determines which address ranges can overlap (non-routable address ranges) and which can't (routable address ranges). The IP management team allocates address ranges from the pool of routable address ranges to projects on request.

Each VPC has its original IP address range, which is non-routable, plus the routable IP address range assigned to it by the IP management team. VPC A has a subnet from its routable range with a private NAT gateway. The private NAT gateway gets its IP address from its subnet. VPC B has a subnet from its routable range with an Application Load Balancer. The Application Load Balancer gets its IP addresses from its subnets.

Traffic from an instance in the non-routable subnet of VPC A that is destined for the instances in the non-routable subnet of VPC B is sent through the private NAT gateway and then routed to the transit gateway. The transit gateway sends the traffic to the Application Load Balancer, which routes the traffic to one of the target instances in the non-routable subnet of VPC B. The traffic from the transit gateway to the Application Load Balancer has the source IP address of the private NAT gateway. Therefore, response traffic from the load balancer uses the address of the private NAT gateway as its destination. The response traffic is sent to the transit gateway and then routed to the private NAT gateway, which translates the destination to the instance in the non-routable subnet of VPC A.



Resources

Create or update resources as follows:

- Associate the assigned routable IP address ranges with their respective VPCs.
- Create a subnet in VPC A from its routable IP address range, and create a private NAT gateway in this new subnet.
- Create a subnet in VPC B from its routable IP address range, and create an Application Load Balancer in this new subnet. Register the instances in the non-routable subnet with the target group for the load balancer.
- Create a transit gateway to connect the VPCs. Be sure to disable route propagation. When you attach each VPC to the transit gateway, use the routable address range of the VPC.
- Update the route table of the non-routable subnet in VPC A to send all traffic destined for the routable address range of VPC B to the private NAT gateway. Update the route table of the routable subnet in VPC A to send all traffic destined for the routable address range of VPC B to the transit gateway.
- Update the route table of the routable subnet in VPC B to send all traffic destined for the routable address range of VPC A to the transit gateway.

Routing

The following is the route table for the non-routable subnet in VPC A.

Destination	Target
10.0.0.0/16	local
100.64.1.0/24	local
100.64.2.0/24	*nat-gateway-id*

The following is the route table for the routable subnet in VPC A.

Destination	Target
10.0.0.0/16	local
100.64.1.0/24	local
100.64.2.0/24	*transit-gateway-id*

The following is the route table for the non-routable subnet in VPC B.

Destination	Target
10.0.0.0/16	local
100.64.2.0/24	local

The following is the route table for the routable subnet in VPC B.

Destination	Target
10.0.0.0/16	local
100.64.2.0/24	local

Destination	Target
100.64.1.0/24	*transit-gateway-id*

The following is the transit gateway route table.

CIDR	Attachment	Route type
100.64.1.0/24	*Attachment for VPC A*	Static
100.64.2.0/24	*Attachment for VPC B*	Static

DNS64 and NAT64

A NAT gateway supports network address translation from IPv6 to IPv4, popularly known as NAT64. NAT64 helps your IPv6 AWS resources communicate with IPv4 resources in the same VPC or a different VPC, in your on-premises network or over the internet. You can use NAT64 with DNS64 on Amazon Route 53 Resolver or use your own DNS64 server.

Contents

- What is DNS64?
- What is NAT64?
- Configure DNS64 and NAT64

What is DNS64?

Your IPv6-only workloads running in VPCs can only send and receive IPv6 network packets. Without DNS64, a DNS query for an IPv4-only service will yield an IPv4 destination address in response and your IPv6-only service can't communicate with it. To bridge this communication gap, you can enable DNS64 for a subnet and it applies to all the AWS resources within that subnet. With DNS64, the Amazon Route 53 Resolver looks up the DNS record for the service you queried for and does one of the following:

- If the record contains an IPv6 address, it returns the original record and the connection is established without any translation over IPv6.

- If there is no IPv6 address associated with the destination in the DNS record, the Route 53 Resolver synthesizes one by prepending the well-known `/96` prefix, defined in RFC6052 (`64:ff9b::/96`), to the IPv4 address in the record. Your IPv6-only service sends network packets to the synthesized IPv6 address. You will then need to route this traffic through the NAT gateway, which performs the necessary translation on the traffic to allow IPv6 services in your subnet to access IPv4 services outside that subnet.

You can enable or disable DNS64 on a subnet using the [modify-subnet-attribute](https://docs.aws.amazon.com/cli/latest/reference/ec2/modify-subnet-attribute.html) using the AWS CLI or with the VPC console by selecting a subnet and choosing **Actions** > **Edit subnet settings**.

What is NAT64?

NAT64 enables your IPv6-only services in Amazon VPCs to communicate with IPv4-only services within the same VPC (in different subnets) or connected VPCs, in your on-premises networks, or over the internet.

NAT64 is automatically available on your existing NAT gateways or on any new NAT gateways you create. It's not a feature you enable or disable. The subnet that the NAT gateway is in does not need to be a dual-stack subnet for NAT64 to work.

After you enable DNS64, if your IPv6-only service sends network packets to a synthesized IPv6 address through the NAT gateway, the following happens:

- From the `64:ff9b::/96` prefix, the NAT gateway recognizes that the original destination is IPv4 and translates the IPv6 packets to IPv4 by replacing:
  - Source IPv6 with its own private IP which is translated to Elastic IP address by the internet gateway.
  - Destination IPv6 to IPv4 by truncating the `64:ff9b::/96` prefix.
- The NAT gateway sends the translated IPv4 packets to the destination through the internet gateway, virtual private gateway, or transit gateway and initiates a connection.
- The IPv4-only host sends back IPv4 response packets. After a connection is established, NAT gateway accepts the response IPv4 packets from the external hosts.
- The response IPv4 packets are destined for NAT gateway, which receives the packets and de-NATs them by replacing its IP (destination IP) with the host's IPv6 address and prepending back `64:ff9b::/96` to the source IPv4 address. The packet then flows to the host following the local route.

In this way, the NAT gateway enables your IPv6-only workloads in a subnet to communicate with IPv4-only services outside the subnet.

Configure DNS64 and NAT64

Follow the steps in this section to configure DNS64 and NAT64 to enable communication with IPv4-only services.

Contents

- Enable communication with IPv4-only services on the internet with the AWS CLI
- Enable communication with IPv4-only services in your on-premises environment

Enable communication with IPv4-only services on the internet with the AWS CLI

If you have a subnet with IPv6-only workloads that needs to communicate with IPv4-only services outside the subnet, this example shows you how to enable these IPv6-only services to communicate with IPv4-only services on the internet.

You should first configure a NAT gateway in a public subnet (separate from the subnet containing the IPv6-only workloads). For example, the subnet containing the NAT gateway should have a `0.0.0.0/0` route pointing to the internet gateway.

Complete these steps to enable these IPv6-only services to connect with IPv4-only services on the internet:

1. Add the following three routes to the route table of the subnet containing the IPv6-only workloads:

   - IPv4 route (if any) pointing to the NAT gateway.
   - `64:ff9b::/96` route pointing to the NAT gateway. This will allow traffic from your IPv6-only workloads destined for IPv4-only services to be routed through the NAT gateway.
   - IPv6 `::/0` route pointing to the egress-only internet gateway (or the internet gateway).

   Note that pointing `::/0` to the internet gateway will allow external IPv6 hosts (outside the VPC) to initiate connection over IPv6.

   ```
   aws ec2 create-route --route-table-id rtb-34056078 --destination-cidr-block
   ```

```
0.0.0.0/0 --nat-gateway-id nat-05dba92075d71c408
```

```
aws ec2 create-route --route-table-id rtb-34056078 --destination-ipv6-cidr-block
64:ff9b::/96 --nat-gateway-id nat-05dba92075d71c408
```

```
aws ec2 create-route --route-table-id rtb-34056078 --destination-ipv6-cidr-block
::/0 --egress-only-internet-gateway-id eigw-c0a643a9
```

2. Enable DNS64 capability in the subnet containing the IPv6-only workloads.

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --enable-dns64
```

Now, resources in your private subnet can establish stateful connections with both IPv4 and IPv6 services on the internet. Configure your security group and NACLs appropriately to allow egress and ingress traffic to `64:ff9b::/96` traffic.

Enable communication with IPv4-only services in your on-premises environment

Amazon Route 53 Resolver enables you to forward DNS queries from your VPC to an on-premises network and vice versa. You can do this by doing the following:

- You create a Route 53 Resolver outbound endpoint in a VPC and assign it the IPv4 addresses that you want Route 53 Resolver to forward queries from. For your on-premises DNS resolver, these are the IP addresses that the DNS queries originate from and, therefore, should be IPv4 addresses.
- You create one or more rules which specify the domain names of the DNS queries that you want Route 53 Resolver to forward to your on-premises resolvers. You also specify the IPv4 addresses of the on-premises resolvers.
- Now that you have set up a Route 53 Resolver outbound endpoint, you need to enable DNS64 on the subnet containing your IPv6-only workloads and route any data destined for your on-premises network through a NAT gateway.

How DNS64 works for IPv4-only destinations in on-premises networks:

1. You assign an IPv4 address to the Route 53 Resolver outbound endpoint in your VPC.

2. The DNS query from your IPv6 service goes to Route 53 Resolver over IPv6. Route 53 Resolver matches the query against the forwarding rule and gets an IPv4 address for your on-premises resolver.
3. Route 53 Resolver converts the query packet from IPv6 into IPv4 and forwards it to the outbound endpoint. Each IP address of the endpoint represents one ENI that forwards the request to the on-premises IPv4 address of your DNS resolver.
4. The on-premises resolver sends the response packet over IPv4 back through the outbound endpoint to Route 53 Resolver.
5. Assuming the query was made from a DNS64-enabled subnet, Route 53 Resolver does two things:

    a. Checks the content of the response packet. If there's an IPv6 address in the record, it keeps the content as is, but if it contains only an IPv4 record. It synthesizes an IPv6 record as well by prepending `64:ff9b::/96` to the IPv4 address.

    b. Repackages the content and sends it to the service in your VPC over IPv6.

Monitor NAT gateways with Amazon CloudWatch

You can monitor your NAT gateway using CloudWatch, which collects information from your NAT gateway and creates readable, near real-time metrics. You can use this information to monitor and troubleshoot your NAT gateway. These metrics give you visibility into the health and performance of your NAT gateway, enabling you to closely monitor its operation and quickly troubleshoot any issues.

The NAT gateway metrics collected by CloudWatch include data points such as bytes processed, packet counts, connection counts, and error rates. This enables you to thoroughly understand the traffic flowing through your NAT gateway and identify any anomalies or bottlenecks. CloudWatch delivers this metric data at 1-minute intervals, giving you a granular, up-to-the-minute view of your NAT gateway's behavior.

Additionally, CloudWatch retains this NAT gateway metric data for an extended period of 15 months, enabling you to analyze trends and patterns over time. You can use this historical data for capacity planning, performance optimization, and understanding the long-term evolution of your NAT gateway usage.

To leverage these powerful monitoring capabilities, you can create custom CloudWatch dashboards and alarms tailored to your specific needs. For example, you could set up alerts to notify you whenever your NAT gateway's outbound data transfer exceeds a certain threshold, allowing you to proactively address potential bandwidth constraints.

For more information about pricing, see Amazon CloudWatch Pricing.

Contents

- NAT gateway metrics and dimensions
- View NAT gateway CloudWatch metrics
- Create CloudWatch alarms to monitor a NAT gateway

NAT gateway metrics and dimensions

The following metrics are available for your NAT gateways. The description column includes a description of each metrics as well as the units and statistics.

Metric	Description
`ActiveConnectionCount`	The total number of concurrent active TCP connections through the NAT gateway. A value of zero indicates that there are no active connections through the NAT gateway. Units: Count Statistics: The most useful statistic is `Max`.
`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 might 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`.

Metric	Description
`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 might 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 might 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 might 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. This includes only the initial SYN. In some cases, `ConnectionAttemptCount` may be lower than `ConnectionEstablishedCount` due to SYN retransmission. 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`.

Metric	Description
`ConnectionEstablishedCount`	The number of connections established through the NAT gateway. This includes SYN and SYN retransmissions. 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 might 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. To calculate the number of dropped packets as a percentage of the overall packet traffic, use this formula: `PacketsDropCount/(PacketsInFromSource+PacketsInFromDestination)*100` . If this value exceeds 0.01 percent of the total traffic on the NAT gateway, there may be an issue with Amazon VPC service. Use the AWS service health dashboard to identify any issues with the service that may be causing NAT gateways to drop packets. Units: Count Statistics: The most useful statistic is `Sum`.

Metric	Description
`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 might be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Unit: Count Statistics: The most useful statistic is `Sum`.
`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 might 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 might 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 might be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Unit: Count Statistics: The most useful statistic is `Sum`.

Metric	Description
PeakBytesPerSecond	This metric reports the highest 10-second bytes per second average in a given minute. Units: Count Statistics: The most useful statistic is Maximum.
PeakPacketsPerSecond	This metric calculates the average packet rate (packets processed per second) every 10 seconds for 60 seconds and then reports the maximum of the six rates (the highest average packet rate). Units: Count Statistics: The most useful statistic is Maximum.

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. Metrics are grouped first by the service namespace, and then by the possible combinations of dimensions within each namespace. You can view the metrics for your NAT gateways as follows.

To view metrics using the CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**, **All metrics**.
3. Choose the **NATGateway** metric namespace.

4. Choose a metric dimension.

To view metrics using the AWS CLI

At a command prompt, use the following command to list the metrics that are available for the NAT gateway service.

```
aws cloudwatch list-metrics --namespace "AWS/NATGateway"
```

Create CloudWatch alarms to monitor a NAT gateway

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

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

To create an alarm for outbound traffic through the NAT gateway

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **All alarms**.
3. Choose **Create alarm**.
4. Choose **Select metric**.
5. Choose the **NATGateway** metric namespace and then choose a metric dimension. When you get to the metrics, select the check box next to the **BytesOutToDestination** metric for the NAT gateway, and then choose **Select metric**.
6. Configure the alarm as follows, and then choose **Next**:
   - For **Statistic**, choose **Sum**.
   - For **Period**, choose **15 minutes**.
   - For **Whenever**, choose **Greater/Equal** and enter `5000000` for the threshold.
7. For **Notification**, select an existing SNS topic or choose **Create new topic** to create a new one. Choose **Next**.

8. Enter a name and description for the alarm and choose **Next**.
9. When you done configuring the alarm, choose **Create alarm**.

As another example, you can create an alarm that monitors port allocation errors and sends a notification when the value is greater than zero (0) for three consecutive 5-minute periods.

To create an alarm to monitor port allocation errors

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **All alarms**.
3. Choose **Create alarm**.
4. Choose **Select metric**.
5. Choose the **NATGateway** metric namespace and then choose a metric dimension. When you get to the metrics, select the check box next to the **ErrorPortAllocation** metric for the NAT gateway, and then choose **Select metric**.
6. Configure the alarm as follows, and then choose **Next**:
   - For **Statistic**, choose **Maximum**.
   - For **Period**, choose **5 minutes**.
   - For **Whenever**, choose **Greater** and enter 0 for the threshold.
   - For **Additional configuration**, **Datapoints to alarm**, enter 3.
7. For **Notification**, select an existing SNS topic or choose **Create new topic** to create a new one. Choose **Next**.
8. Enter a name and description for the alarm and choose **Next**.
9. When you are done configuring the alarm, choose **Create alarm**.

For more information, see Using Amazon CloudWatch alarms in the *Amazon CloudWatch User Guide*.

Troubleshoot NAT gateways

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

Issues

- NAT gateway creation fails
- NAT gateway quota
- Elastic IP address quota
- Availability Zone is unsupported
- NAT gateway is no longer visible
- NAT gateway doesn't respond to a ping command
- Instances cannot access the internet
- TCP connection to a destination fails
- Traceroute output does not display NAT gateway private IP address
- Internet connection drops after 350 seconds
- IPsec connection cannot be established
- Cannot initiate more connections

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 Add internet access to a subnet.
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. Alternati vely, allocate a new Elastic IP address to your account. For more information, see Start using Elastic IP addresses.

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.

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, 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.

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.
- Check that you've configured your route tables correctly:
  - The NAT gateway must be in a public subnet with a route table that routes internet traffic to an internet gateway.
  - Your instance must be in a private subnet with a route table that routes internet traffic to the NAT gateway.

  - Check that there are no other route table entries that route all or part of the internet traffic to another device instead of the NAT gateway.

- Ensure that your security group rules for your private instance allow outbound internet traffic. For the `ping` command to work, the rules must also allow outbound ICMP traffic.

  The NAT gateway itself allows all outbound traffic and traffic received in response to an outbound request (it is therefore stateful).

- Ensure that the network ACLs that are associated with the private subnet and public subnets do not have rules that block inbound or outbound internet traffic. For the `ping` command to work, the rules must also allow inbound and outbound ICMP traffic.

  You can enable flow logs to help you diagnose dropped connections because of network ACL or security group rules. For more information, see Logging IP traffic using VPC Flow Logs.

- 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.

- The `tcp_tw_recycle` option is enabled on the remote server, which is known to cause issues when there are multiple connections from behind a NAT device.

Solutions

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

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

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

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

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

1. From the server, run the following command.

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

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

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

Traceroute output does not display NAT gateway private IP address

Problem

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

Cause

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

Solution

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

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

Internet connection drops after 350 seconds

Problem

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

Cause

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

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

Solution

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

IPsec connection cannot be established

Problem

You cannot establish an IPsec connection to a destination.

Cause

NAT gateways currently do not support the IPsec protocol.

Solution

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

Cannot initiate more connections

Problem

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

Cause

You might have reached the limit for simultaneous connections for a single NAT gateway. For more information, see NAT gateway basics. If your instances in the private subnet create a large number of connections, you might reach this limit.

Solution

Do one of the following:

- Create a NAT gateway per Availability Zone and spread your clients across those zones.
- Create additional NAT gateways in the public subnet and split your clients into multiple private subnets, each with a route to a different NAT gateway.
- Limit the number of connections your clients can create to the destination.
- Use the `IdleTimeoutCount` metric in CloudWatch to monitor for increases in idle connections. Close idle connections to release capacity.
- Create a NAT gateway with multiple IP addresses or add secondary IP addresses to an existing NAT gateway. Each new IPv4 address can support up to 55,000 concurrent connections. For more information, see Create a NAT gateway or Edit secondary IP address associations.

Pricing for NAT gateways

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

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

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

NAT instances

A NAT instance provides network address translation (NAT). You can use a NAT instance to allow resources in a private subnet to communicate with destinations outside the virtual private cloud (VPC), such as the internet or an on-premises network. The resources in the private subnet can initiate outbound IPv4 traffic to the internet, but they can't receive inbound traffic initiated on the internet.

> **Important**
>
> NAT AMI is built on the last version of the Amazon Linux AMI, 2018.03, which reached the end of standard support on December 31, 2020 and end of maintenance support on December 31, 2023. For more information, see the following blog post: Amazon Linux AMI end of life.
> If you use an existing NAT AMI, AWS recommends that you migrate to a NAT gateway. NAT gateways provide better availability, higher bandwidth, and requires less administrative effort. For more information, see Compare NAT gateways and NAT instances.
> If NAT instances are a better match for your use case than NAT gateways, you can create your own NAT AMI from a current version of Amazon Linux as described in the section called “3. Create a NAT AMI”.

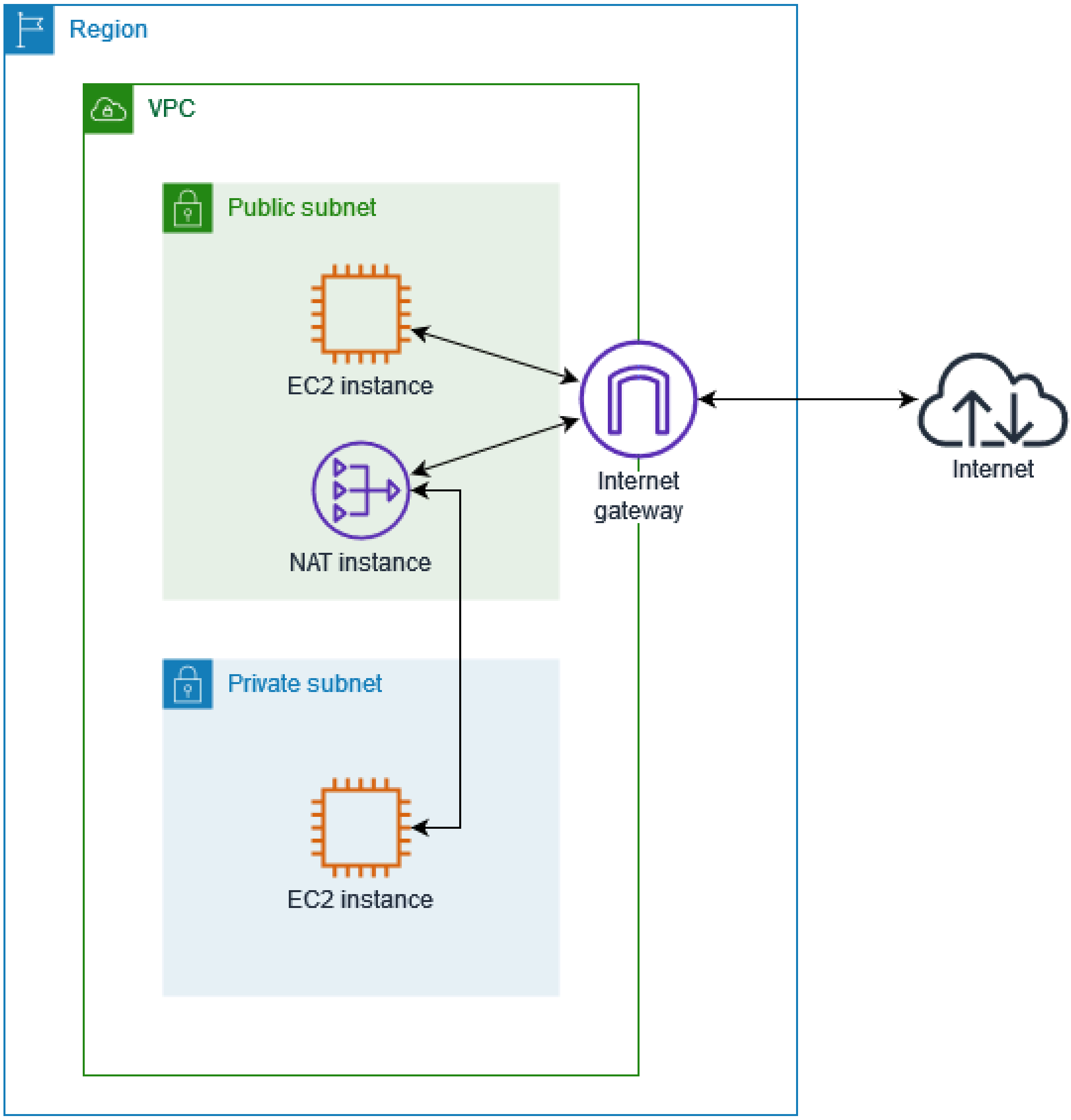
Contents

- NAT instance basics
- Enable private resources to communicate outside the VPC

NAT instance basics

The following figure illustrates the NAT instance basics. The route table associated with the private subnet sends internet traffic from the instances in the private subnet to the NAT instance in the public subnet. The NAT instance then sends the traffic to the internet gateway. The traffic is attributed to the public IP address of the NAT instance. The NAT instance specifies a high port number for the response; if a response comes back, the NAT instance sends it to an instance in the private subnet based on the port number for the response.

The NAT instance must have internet access, so it must be in a public subnet (a subnet that has a route table with a route to the internet gateway), and it must have a public IP address or an Elastic IP address.



To get started with NAT instances, create a NAT AMI, create a security group for the NAT instance, and launch the NAT instance into your VPC.

Your NAT instance quota depends on your instance quota for the Region. For more information, see Amazon EC2 service quotas in the *AWS General Reference*.

Enable private resources to communicate outside the VPC

This section describes how to create and work with NAT instances to enable resources in a private subnet to communicate outside the virtual private cloud.

Tasks

- 1. Create a VPC for the NAT instance

- 2. Create a security group for the NAT instance
- 3. Create a NAT AMI
- 4. Launch a NAT instance
- 5. Disable source/destination checks
- 6. Update the route table
- 7. Test your NAT instance

1. Create a VPC for the NAT instance

Use the following procedure to create a VPC with a public subnet and a private subnet.

To create the VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose **Create VPC**.
3. For **Resources to create**, choose **VPC and more**.
4. For **Name tag auto-generation**, enter a name for the VPC.
5. To configure the subnets, do the following:
   a. For **Number of Availability Zones**, choose **1** or **2**, depending on your needs.
   b. For **Number of public subnets**, ensure that you have one public subnet per Availability Zone.
   c. For **Number of private subnets**, ensure that you have one private subnet per Availability Zone.
6. Choose **Create VPC**.

2. Create a security group for the NAT instance

Create a security group with the rules described in the following table. These rules enable your NAT instance to receive internet-bound traffic from instances in the private subnet, as well as SSH traffic from your network. The NAT instance can also send traffic to the internet, which enables the instances in the private subnet to get software updates.

The following are the inbound recommended rules.

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)

The following are the recommended outbound rules.

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 security group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security groups**.
3. Choose **Create security group**.
4. Enter a name and description for the security group.
5. For **VPC**, select the ID of the VPC for your NAT instance.
6. Add rules for inbound traffic under **Inbound rules** as follows:

   a. Choose **Add rule**. Choose **HTTP** for **Type** and enter the IP address range of your private subnet for **Source**.

   b. Choose **Add rule**. Choose **HTTPS** for **Type** and enter the IP address range of your private subnet for **Source**.

   c. Choose **Add rule**. Choose **SSH** for **Type** and enter the IP address range of your network for **Source**.
7. Add rules for outbound traffic under **Outbound rules** as follows:
   a. Choose **Add rule**. Choose **HTTP** for **Type** and enter 0.0.0.0/0 for **Destination**.
   b. Choose **Add rule**. Choose **HTTPS** for **Type** and enter 0.0.0.0/0 for **Destination**.
8. Choose **Create security group**.

For more information, see Security groups.

3. Create a NAT AMI

A NAT AMI is configured to run NAT on an EC2 instance. You must create a NAT AMI and then launch your NAT instance using your NAT AMI.

If you plan to use an operating system other than Amazon Linux for your NAT AMI, refer to the documentation for this operating system to learn how to configure NAT. Be sure to save these settings so that they persist even after an instance reboot.

To create a NAT AMI for Amazon Linux

1. Launch an EC2 instance running AL2023 or Amazon Linux 2. Be sure to specify the security group that you created for the NAT instance.
2. Connect to your instance and run the following commands on the instance to enable iptables.

```
sudo yum install iptables-services -y
sudo systemctl enable iptables
sudo systemctl start iptables
```

3. Do the following on the instance to enable IP forwarding such that it persists after reboot:
   a. Using a text editor, such as **nano** or **vim**, create the following configuration file: `/etc/sysctl.d/custom-ip-forwarding.conf`.
   b. Add the following line to the configuration file.

   ```
   net.ipv4.ip_forward=1
   ```

   c. Save the configuration file and exit the text editor.
   d. Run the following command to apply the configuration file.

```
sudo sysctl -p /etc/sysctl.d/custom-ip-forwarding.conf
```

4. Run the following command on the instance, and note the name of the primary network interface. You'll need this information for the next step.

```
netstat -i
```

In the following example output, `docker0` is a network interface created by docker, `eth0` is the primary network interface, and `lo` is the loopback interface.

```
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
docker0   1500        0      0      0 0             0      0      0      0 BMU
eth0      9001  7276052      0      0 0       5364991      0      0      0 BMRU
lo       65536   538857      0      0 0        538857      0      0      0 LRU
```

In the following example output, the primary network interface is `enX0`.

```
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
enX0      9001     1076      0      0 0          1247      0      0      0 BMRU
lo       65536       24      0      0 0            24      0      0      0 LRU
```

In the following example output, the primary network interface is `ens5`.

```
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
ens5      9001    14036      0      0 0          2116      0      0      0 BMRU
lo       65536       12      0      0 0            12      0      0      0 LRU
```

5. Run the following commands on the instance to configure NAT. If the primary network interface is not `eth0`, replace *`eth0`* with the primary network interface that you noted in the previous step.

```
sudo /sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo /sbin/iptables -F FORWARD
sudo service iptables save
```

6. Create a NAT AMI from the EC2 instance. For more information, see [Create a Linux AMI from an instance]() in the *Amazon EC2 User Guide*.

4. Launch a NAT instance

Use the following procedure to launch a NAT instance using the VPC, security group, and NAT AMI that you created.

To launch a NAT instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the dashboard, choose **Launch instance**.
3. For **Name**, enter a name for your NAT instance.
4. For **Application and OS Images**, select your NAT AMI (choose **Browse more AMIs**, **My AMIs**).
5. For **Instance type**, choose an instance type that provides the compute, memory, and storage resources that your NAT instance needs.
6. For **Key pair**, select an existing key pair or choose **Create new key pair**.
7. For **Network settings**, do the following:

   a. Choose **Edit**.

   b. For **VPC**, choose the VPC that you created.

   c. For **Subnet**, choose the public subnet that you created.

   d. For **Auto-assign public IP**, choose **Enable**. Alternatively, after you launch the NAT instance, allocate an Elastic IP address and assign it to the NAT instance.

   e. For **Firewall**, choose **Select existing security group** and then choose the security group that you created.
8. Choose **Launch instance**. Choose the instance ID to open the instance details page. Wait for the instance state to change to **Running** and for the status checks to succeed.
9. Disable source/destination checks for the NAT instance (see 5. Disable source/destination checks).
10. Update the route table to send traffic to the NAT instance (see 6. Update the route table).

5. Disable source/destination checks

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

To disable source/destination checking

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Instances**.
3. Select the NAT instance.
4. Choose **Actions**, **Networking**, **Change source/destination check**.
5. For **Source/destination checking**, select **Stop**.
6. Choose **Save**.
7. If the NAT instance has a secondary network interface, choose it from **Network interfaces** on the **Networking** tab. Choose the interface ID to go to the network interfaces page. Choose **Actions**, **Change source/dest. check**, clear **Enable**, and choose **Save**.

6. Update the route table

The route table for the private subnet must have a route that sends internet traffic to the NAT instance.

To update the route table

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route tables**.
3. Select the route table for the private subnet.
4. On the **Routes** tab, choose **Edit routes** and then choose **Add route**.
5. Enter 0.0.0.0/0 for **Destination** and the instance ID of the NAT instance for **Target**.
6. Choose **Save changes**.

For more information, see Configure route tables.

7. Test your NAT instance

After you have launched a NAT instance and completed the configuration steps above, you can test whether an instance in your private subnet can access the internet through the NAT instance by using the NAT instance as a bastion server.

Tasks

- Step 1: Update the NAT instance security group
- Step 2: Launch a test instance in the private subnet
- Step 3: Ping an ICMP-enabled website
- Step 4: Clean up

Step 1: Update the NAT instance security group

To allow instances in your private subnet to send ping traffic to the NAT instance, add a rule to allow inbound and outbound ICMP traffic. To allow the NAT instance to serve as a bastion server, add a rule to allow outbound SSH traffic to the private subnet.

To update your NAT instance security group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security groups**.
3. Select the check box for the security group associated with your NAT instance.
4. On the **Inbound rules** tab, choose **Edit inbound rules**.
5. Choose **Add rule**. Choose **All ICMP - IPv4** for **Type**. Choose **Custom** for **Source** and enter the IP address range of your private subnet. Choose **Save rules**.
6. On the **Outbound rules** tab, choose **Edit outbound rules**.
7. Choose **Add rule**. Choose **SSH** for **Type**. Choose **Custom** for **Destination** and enter the IP address range of your private subnet.
8. Choose **Add rule**. Choose **All ICMP - IPv4** for **Type**. Choose **Anywhere - IPv4** for **Destination**. Choose **Save rules**.

Step 2: Launch a test instance in the private subnet

Launch an instance into your private subnet. You must allow SSH access from the NAT instance, and you must use the same key pair that you used for the NAT instance.

To launch a test instance in the private subnet

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the dashboard, choose **Launch instance**.
3. Select your private subnet.

4. Do not assign a public IP address to this instance.
5. Ensure that the security group for this instance allows inbound SSH access from your NAT instance, or from the IP address range of your public subnet, and outbound ICMP traffic.
6. Select the same key pair that you used for the NAT instance.

Step 3: Ping an ICMP-enabled website

To verify that the test instance in your private subnet can use your NAT instance to communicate with the internet, run the **ping** command.

To test the internet connection from your private instance

1. From your local computer, configure SSH agent forwarding, so that you can use the NAT instance as a bastion server.

   Linux and macOS

   ```
   ssh-add key.pem
   ```

   Windows

   Download and install Pageant, if it is not already installed.

   Convert your private key using PuTTYgen.

   Start Pageant, right-click the **Pageant** icon on the taskbar (it might be hidden), and choose **Add Key**. Select the .ppk file that you created, enter the passphrase if required, and choose **Open**.
2. From your local computer, connect to your NAT instance.

   Linux and macOS

   ```
   ssh -A ec2-user@nat-instance-public-ip-address
   ```

   Windows

   Connect to your NAT instance using PuTTY. For **Auth**, you must select **Allow agent forwarding** and leave **Private key file for authentication** blank.
3. From the NAT instance, run the **ping** command, specifying a website that is enabled for ICMP.

```
[ec2-user@ip-10-0-4-184]$ ping ietf.org
```

To confirm that your NAT instance has internet access, verify that you received output such as the following, and then press **Ctrl+C** to cancel the **ping** command. Otherwise, verify that the NAT instance is in a public subnet (its route table has a route to an internet gateway).

```
PING ietf.org (104.16.45.99) 56(84) bytes of data.
64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=1 ttl=33 time=7.88 ms
64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=2 ttl=33 time=8.09 ms
64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=3 ttl=33 time=7.97 ms
...
```

4. From your NAT instance, connect to your instance in your private subnet by using its private IP address.

```
[ec2-user@ip-10-0-4-184]$ ssh ec2-user@private-server-private-ip-address
```

5. From your private instance, test that you can connect to the internet by running the **ping** command.

```
[ec2-user@ip-10-0-135-25]$ ping ietf.org
```

To confirm that your private instance has internet access through the NAT instance verify that you received output such as the following, and then press **Ctrl+C** to cancel the **ping** command.

```
PING ietf.org (104.16.45.99) 56(84) bytes of data.
64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=1 ttl=33 time=8.76 ms
64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=2 ttl=33 time=8.26 ms
64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=3 ttl=33 time=8.27 ms
...
```

Troubleshooting

If the **ping** command fails from the server in the private subnet, use the following steps to troubleshoot the issue:

- Verify that you pinged a website that has ICMP enabled. Otherwise, your server can't receive reply packets. To test this, run the same **ping** command from a command line terminal on your own computer.
- Verify that the security group for your NAT instance allows inbound ICMP traffic from your private subnet. Otherwise, your NAT instance can't receive the **ping** command from your private instance.
- Verify that you disabled source/destination checking for your NAT instance. For more information, see 5. Disable source/destination checks.
- Verify that you configured your route tables correctly. For more information, see 6. Update the route table.

Step 4: Clean up

If you no longer require the test server in the private subnet, terminate the instance so that you are no longer billed for it. For more information, see Terminate your instance in the *Amazon EC2 User Guide*.

If you no longer require the NAT instance, you can stop or terminate it, so that you are no longer billed for it. If you created a NAT AMI, you can create a new NAT instance whenever you need one.

Compare NAT gateways and NAT instances

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

Attribute	NAT gateway	NAT instance
Availabil ity	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.
Bandwidtl	Scale up to 100 Gbps.	Depends on the bandwidth of the instance type.

Attribute	NAT gateway	NAT instance
Maintenance	Managed by AWS. You do not need to perform any maintenance.	Managed by you, for example, by installing software updates or operating system patches on the instance.
Performance	Software is optimized for handling NAT traffic.	A generic 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 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.

Attribute	NAT gateway	NAT instance
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]().	View CloudWatch metrics for the instance.
Timeout behavior	When a connection times out, a NAT gateway returns an RST packet to any resources behind the NAT gateway that attempt to continue the connection (it does not send a FIN packet).	When a connection times out, a NAT instance sends a FIN packet to resources behind the NAT instance to close the connection.
IP fragmentation	Supports forwarding of IP fragmented packets for the UDP protocol. Does not support fragmentation for the TCP and ICMP protocols. Fragmented packets for these protocols will get dropped.	Supports reassembly of IP fragmented packets for the UDP, TCP, and ICMP protocols.

Migrate from a NAT instance to a NAT gateway

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

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

Associate Elastic IP addresses with resources in your VPC

An Elastic IP address is a static, public IPv4 address designed specifically for the dynamic nature of cloud computing. This feature allows you to associate an Elastic IP address with any instance or network interface within any Virtual Private Cloud (VPC) in your AWS account. By leveraging Elastic IP addresses, you can unlock a host of benefits that simplify the management and resilience of your cloud-based infrastructure.

One of the primary advantages of Elastic IP addresses is their ability to mask the failure of an instance. Should an instance experience an unexpected outage or need to be replaced, you can remap the associated Elastic IP address to another instance within your VPC. This failover process ensures that your applications and services maintain a consistent and reliable public endpoint, minimizing downtime and providing a superior user experience.

Furthermore, Elastic IP addresses offer flexibility in how you manage your network resources. You can programmatically associate and disassociate these addresses as needed, allowing you to direct traffic to different instances based on your evolving business requirements. This dynamic allocation of public IP addresses empowers you to adapt to changing demand, scale your infrastructure, and implement innovative architectures without the constraints of static IP assignments.

Beyond their use for instance failover, Elastic IP addresses can also serve as stable identifiers for your cloud-based resources. This can be beneficial when configuring external services, such as DNS records or firewall rules, to communicate with your AWS-hosted applications. By associating a persistent public IP address, you can future-proof your networking configurations and avoid the need to update external references when underlying instances are replaced or scaled.

Contents

- Elastic IP address concepts and rules
- Start using Elastic IP addresses

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*.

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 primary network interface of your instance, its current public IPv4 address (if it had one) is released to the public IP address pool. If you disassociate the Elastic IP address, the primary network interface is automatically assigned a new public IPv4 address within a few minutes. This doesn't apply if you've attached a second network interface to your instance.
- You're limited to five Elastic IP addresses. To help conserve them, you can use a NAT device. For more information, see Connect to the internet or other networks using NAT devices.
- Elastic IP addresses for IPv6 are not supported.
- You can tag an Elastic IP address that's allocated for use in a VPC, however, cost allocation tags are not supported. If you recover an Elastic IP address, tags are not recovered.
- You can access an Elastic IP address from the internet when the security group and network ACL allow traffic from the source IP address. The reply traffic from within the VPC back to the internet requires an internet gateway. For more information, see Security groups and Network ACLs.
- 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*.

- Public IPv4 addresses support cost allocation tags. If you apply tags to Elastic IP addresses, you can use those tags to track public IPv4 address costs in AWS Cost Explorer.

  Before you can use tags as cost allocation tags, you must activate the tags. For more information, see Activating user-defined cost allocation tags in the *AWS Billing User Guide*. Note that after you create and apply user-defined tags to your resources, it can take up to 24 hours for the tag keys to appear on your cost allocation tags page for activation.

  Once the cost allocation tags are activated...

  - For all public IPv4 addresses (including public IPv4 addresses assigned to EC2 instances and Elastic IP addresses) that are associated with an elastic network interface, you can view the costs associated with public IPv4 addresses in Cost Explorer by choosing **Usage type** > **PublicIPv4InUseAddress (Hrs)**.
  - If a tagged Elastic IP address is not associated with an ENI or is associated with a stopped resource (like a stopped EC2 instance), it's considered an idle IPv4 address. You can view the costs associated with idle IPv4 addresses in Cost Explorer by choosing **Usage type** > **PublicIPv4IdleAddress (Hrs)**.

  For more information about Cost Explorer, see Analyzing your costs with AWS Cost Explorer in the *AWS Billing User Guide*.

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*.

For more information about pricing for Elastic IP addresses, see *Public IPv4 address* in Amazon VPC Pricing.

Start using Elastic IP addresses

The following sections describe how you can get started using Elastic IP addresses.

Tasks

- 1. Allocate an Elastic IP address
- 2. Associate an Elastic IP address
- 3. Disassociate an Elastic IP address
- 4. Transfer Elastic IP addresses

- 5. Release an Elastic IP address
- 6. Recover an Elastic IP address
- Command line overview

1. Allocate an Elastic IP address

Before you use an Elastic IP, you must allocate one for use in your VPC.

To allocate an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Elastic IPs**.
3. Choose **Allocate Elastic IP address**.
4. (Optional) When you allocate an Elastic IP address (EIP), you choose the **Network border group** in which to allocate the EIP. A network border group is a collection of Availability Zones (AZs), Local Zones, or Wavelength Zones from which AWS advertises a public IP address. Local Zones and Wavelength Zones may have different network border groups than the AZs in a Region to ensure minimum latency or physical distance between the AWS network and the customers accessing the resources in these Zones.

> **Important**
>
> You must allocate an EIP in the same network border group as the AWS resource that will be associated with the EIP. An EIP in one network border group can only be advertised in zones in that network border group and not in any other zones represented by other network border groups.

If you have Local Zones or Wavelength Zones enabled (for more information, see Enable a Local Zone or Enable Wavelength Zones), you can choose a network border group for AZs, Local Zones, or Wavelength Zones. Choose the network border group carefully as the EIP and the AWS resource it is associated with must reside in the same network border group. You can use the EC2 console to view the network border group that your Availability Zones, Local Zones, or Wavelength Zones are in (see Local Zones). Typically, all Availability Zones in a Region belong to the same network border group, whereas Local Zones or Wavelength Zones belong to their own separate network border groups.

If you don't have Local Zones or Wavelength Zones enabled, when you allocate an EIP, the network border group that represents all of the AZs for the Region (such as `us-west-2`) is predefined for you and you cannot change it. This means that the EIP that you allocate to this network border group will be advertised in all AZs in the Region you're in.

5. For **Public IPv4 address pool** choose one of the following:

   - **Amazon's pool of IP addresses**—If you want an IPv4 address to be allocated from Amazon's pool of IP addresses.
   - **My pool of public IPv4 addresses**—If you want to allocate an IPv4 address from an IP address pool that you have brought to your AWS account. This option is disabled if you do not have any IP address pools.
   - **Customer owned pool of IPv4 addresses**—If you want to allocate an IPv4 address from a pool created from your on-premises network for use with an Outpost. This option is only available if you have an Outpost.

6. (Optional) Add or remove a tag.

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

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

   [Remove a tag] Choose **Remove** to the right of the tag's Key and Value.

7. Choose **Allocate**.

2. Associate an Elastic IP address

You can associate an Elastic IP with a running instance or network interface in your VPC.

After you associate the Elastic IP address with your instance, the instance receives a public DNS hostname if DNS hostnames are enabled. For more information, see [DNS attributes for your VPC](#).

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**.

3. 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**.

4. Transfer Elastic IP addresses

This section describes how to transfer Elastic IP addresses from one AWS account to another. Transferring Elastic IP addresses can be helpful in the following situations:

- **Organizational restructuring –** Use Elastic IP address transfers to quickly move workloads from one AWS account to another. You don't have to wait for new Elastic IP addresses to be allowlisted in your security groups and NACLs.
- **Centralized security administration –** Use a centralized AWS security account to track and transfer Elastic IP addresses that have been vetted for security compliance.
- **Disaster recovery –** Use Elastic IP address transfers to quickly remap IPs for public-facing internet workloads during emergency events.

There is no charge for transferring Elastic IP addresses.

Tasks

- [Enable Elastic IP address transfer]
- [Disable Elastic IP address transfer]

- Accept a transferred Elastic IP address

Enable Elastic IP address transfer

This section describes how to accept a transferred Elastic IP address. Note the following limitations related to enabling Elastic IP addresses for transfer:

- You can transfer Elastic IP addresses from any AWS account (source account) to any other AWS account in the same AWS Region (transfer account).
- When you transfer an Elastic IP address, there is a two-step handshake between the AWS accounts. When the source account starts the transfer, the transfer accounts have seven days to accept the Elastic IP address transfer. During those seven days, the source account can view the pending transfer (for example in the AWS console or by using the describe-address-transfers AWS CLI command). After seven days, the transfer expires and ownership of the Elastic IP address returns to the source account.
- Accepted transfers are visible to the source account (for example in the AWS console or by using the describe-address-transfers AWS CLI command) for 14 days after the transfers have been accepted.
- AWS does not notify transfer accounts about pending Elastic IP address transfer requests. The owner of the source account must notify the owner of the transfer account that there is an Elastic IP address transfer request that they must accept.
- Any tags that are associated with an Elastic IP address being transferred are reset when the transfer is complete.
- You cannot transfer Elastic IP addresses allocated from public IPv4 address pools that you bring to your AWS account – commonly referred to as Bring Your Own IP (BYOIP) address pools.
- If you attempt to transfer an Elastic IP address that has a reverse DNS record associated with it, you can begin the transfer process, but the transfer account will not be able to accept the transfer until the associated DNS record is removed.
- If you have enabled and configured AWS Outposts, you might have allocated Elastic IP addresses from a customer-owned IP address pool (CoIP). You cannot transfer Elastic IP addresses allocated from a CoIP. However, you can use AWS RAM to share a CoIP with another account. For more information, see Customer-owned IP addresses in the *AWS Outposts User Guide*.
- You can use Amazon VPC IPAM to track the transfer of Elastic IP addresses to accounts in an organization from AWS Organizations. For more information, see View IP address history. If an

Elastic IP address is transferred to an AWS account outside of the organization, the IPAM audit history of the Elastic IP address is lost.

These steps must be completed by the source account.

To enable Elastic IP address transfer

1. Ensure that you're using the source AWS account.
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. In the navigation pane, choose **Elastic IPs**.
4. Select one or more Elastic IP address to enable for transfer and choose **Actions**, **Enable transfer**.
5. If you are transferring multiple Elastic IP addresses, you'll see the **Transfer type** option. Choose one of the following options:
   - Choose **Single account** if you are transferring the Elastic IP addresses to a single AWS account.
   - Choose **Multiple accounts** if you are transferring the Elastic IP addresses to multiple AWS accounts.
6. Under **Transfer account ID**, enter the IDs of the AWS accounts that you want to transfer the Elastic IP addresses to.
7. Confirm the transfer by entering `enable` in the text box.
8. Choose **Submit**.
9. To accept the transfer, see Accept a transferred Elastic IP address. To disable the transfer, see Disable Elastic IP address transfer.

Disable Elastic IP address transfer

This section describes how to disable an Elastic IP transfer after the transfer has been enabled.

These steps must be completed by the source account that enabled the transfer.

To disable an Elastic IP address transfer

1. Ensure that you're using the source AWS account.
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.

3. In the navigation pane, choose **Elastic IPs**.
4. In the resource list of Elastic IPs, ensure that you have the property enabled that shows the column **Transfer status**.
5. Select one or more Elastic IP address that have a **Transfer status** of **Pending**, and choose **Actions**, **Disable transfer**.
6. Confirm by entering **`disable`** in the text box.
7. Choose **Submit**.

Accept a transferred Elastic IP address

This section describes how to accept a transferred Elastic IP address.

When you transfer an Elastic IP address, there is a two-step handshake between the AWS accounts. When the source account starts the transfer, the transfer accounts have seven days to accept the Elastic IP address transfer. During those seven days, the source account can view the pending transfer (for example in the AWS console or by using the describe-address-transfers AWS CLI command). After seven days, the transfer expires and ownership of the Elastic IP address returns to the source account.

When accepting transfers, note the following exceptions that might occur and how to resolve them:

- **AddressLimitExceeded**: If your transfer account has exceeded the Elastic IP address quota, the source account can enable Elastic IP address transfer, but this exception occurs when the transfer account tries to accept the transfer. By default, all AWS accounts are limited to 5 Elastic IP addresses per Region. See Elastic IP address limit in the *Amazon EC2 User Guide* for instructions on increasing the limit.
- **InvalidTransfer.AddressCustomPtrSet**: If you or someone in your organization has configured the Elastic IP address that you are attempting to transfer to use reverse DNS lookup, the source account can enable transfer for the Elastic IP address, but this exception occurs when the transfer account tries to accept the transfer. To resolve this issue, the source account must remove the DNS record for the Elastic IP address. For more information, see Remove a reverse DNS record in the *Amazon EC2 User Guide*.
- **InvalidTransfer.AddressAssociated**: If an Elastic IP address is associated with an ENI or EC2 instance, the source account can enable transfer for the Elastic IP address, but this exception occurs when the transfer account tries to accept the transfer. To resolve this issue, the source

account must disassociate the Elastic IP address. For more information, see Disassociate an Elastic IP address in the *Amazon EC2 User Guide*.

For any other exceptions, contact Support.

These steps must be completed by the transfer account.

To accept an Elastic IP address transfer

1. Ensure that you're using the transfer account.
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. In the navigation pane, choose **Elastic IPs**.
4. Choose **Actions**, **Accept transfer**.
5. No tags that are associated with the Elastic IP address being transferred are transferred with the Elastic IP address when you accept the transfer. If you want to define a **Name** tag for the Elastic IP address that you are accepting, select **Create a tag with a key of 'Name' and a value that you specify**.
6. Enter the Elastic IP address that you want to transfer.
7. If you are accepting multiple transferred Elastic IP addresses, choose **Add address** to enter an additional Elastic IP address.
8. Choose **Submit**.

5. 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 even if it's not associated with an instance. The Elastic IP address must not be associated with an instance or network interface.

To release an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Elastic IPs**.
3. Select the Elastic IP address, and then choose **Actions**, **Release Elastic IP addresses**.
4. When prompted, choose **Release**.

6. Recover an Elastic IP address

If you release an Elastic IP address but change your mind, you might be able to recover it. You cannot recover the Elastic IP address if it has been allocated to another AWS account, or if recovering it results in you exceeding your Elastic IP address quota.

You can recover an Elastic IP address by using the Amazon EC2 API or a command line tool.

To recover an Elastic IP address using the AWS CLI

Use the allocate-address command and specify the IP address using the `--address` parameter.

```
aws ec2 allocate-address --domain vpc --address 203.0.113.3
```

Command line overview

You can perform the tasks described in this section using the command line or an API. For more information about the command line interfaces and a list of available API actions, see Working with Amazon VPC.

Accept Elastic IP address transfer

- accept-address-transfer (AWS CLI)
- Approve-EC2AddressTransfer (AWS Tools for Windows PowerShell)

Allocate an Elastic IP address

- allocate-address (AWS CLI)
- New-EC2Address (AWS Tools for Windows PowerShell)

Associate an Elastic IP address with an instance or network interface

- associate-address (AWS CLI)
- Register-EC2Address (AWS Tools for Windows PowerShell)

Describe Elastic IP address transfers

- describe-address-transfers (AWS CLI)

- Get-EC2AddressTransfer (AWS Tools for Windows PowerShell)

Disable Elastic IP address transfer

- disable-address-transfer (AWS CLI)
- Disable-EC2AddressTransfer (AWS Tools for Windows PowerShell)

Disassociate an Elastic IP address

- disassociate-address (AWS CLI)
- Unregister-EC2Address (AWS Tools for Windows PowerShell)

Enable Elastic IP address transfer

- enable-address-transfer (AWS CLI)
- Enable-EC2AddressTransfer (AWS Tools for Windows PowerShell)

Release an Elastic IP address

- release-address (AWS CLI)
- Remove-EC2Address (AWS Tools for Windows PowerShell)

Tag an Elastic IP address

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

View your Elastic IP addresses

- describe-addresses (AWS CLI)
- Get-EC2Address (AWS Tools for Windows PowerShell)

Connect your VPC to other VPCs and networks using a transit gateway

You can connect your virtual private clouds (VPC) and on-premises networks using a transit gateway, which acts as a central hub, routing traffic between VPCs, VPN connections, and AWS Direct Connect connections.

One of the key benefits of using a transit gateway is the ability to centralize and simplify the management of connectivity between your VPCs and on-premises networks. Rather than configuring multiple VPN connections or Direct Connect links, you can leverage the transit gateway as a single point of integration, which can help reduce the overall complexity and operational overhead of your network architecture.

The pricing for using a transit gateway is based on the volume of data transferred through the gateway. There is a per-GB rate for data transferred in and out of the transit gateway, as well as a separate per-hour rate for the transit gateway resource itself. The specific pricing can vary by AWS Region and is subject to change, so it's important to refer to the current AWS Transit Gateway pricing page for the most up-to-date information. By understanding the pricing model for transit gateways, you can better plan and budget for the ongoing costs associated with this AWS networking service. This, combined with the operational efficiencies and connectivity benefits, makes transit gateways a compelling choice for organizations looking to build scalable and cost-effective hybrid cloud solutions.

The following table describes some common use case for transit gateways. For more information about each use case, see Example transit gateway scenarios in the *AWS Transit Gateway User Guide*.

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.
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.

Example	Usage
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 AWS Transit Gateway.

Connect your VPC to remote networks using AWS Virtual Private Network

You can connect your VPC to remote networks and users using the following VPN connectivity options.

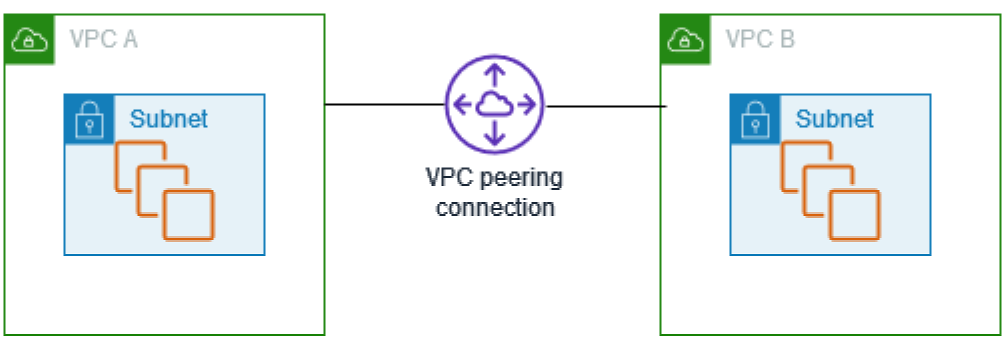
VPN connectivity option	Description
AWS Site-to-Site VPN	You can create an IPsec VPN connection between your VPC and your remote network. On the AWS side of the Site-to-Site VPN connection, a virtual private gateway or transit gateway provides two VPN endpoints (tunnels) for automatic failover. You configure 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

VPN connectivity option	Description
	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*.

Connect VPCs using VPC peering

A VPC peering connection is a networking feature that enables secure and direct communication between two virtual private clouds (VPCs) within the AWS infrastructure. This private connection allows resources in the peered VPCs to interact with each other as if they were part of the same network, eliminating the need to traverse the public internet.



The process of creating a VPC peering connection leverages the existing VPC infrastructure to establish this connection, without the requirement of a gateway, AWS Site-to-Site VPN, or any additional physical hardware. This design ensures that there is no single point of failure or bandwidth bottleneck.

One of the key advantages of a VPC peering connection is the ability to connect VPCs across different AWS accounts or even different AWS Regions. This flexibility allows organizations to seamlessly integrate their cloud resources, whether they are within the same account or spread across multiple accounts and geographic locations. The private nature of the connection also ensures that all data traffic between the peered VPCs remains within the AWS network, without ever traversing the public internet.

The use cases for VPC peering connections are wide-ranging. Organizations can leverage this feature to enable secure communication between different tiers of an application (such as web servers and database servers), facilitate the sharing of resources between multiple teams or business units, or even enable hybrid cloud architectures by connecting on-premises networks to their AWS VPCs.

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them privately. Resources in peered VPCs can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, with a VPC in another AWS account, or with a VPC in a different AWS Region. Traffic between peered VPCs never traverses the public internet.

For more information, see the Amazon VPC Peering Guide.

Monitoring your VPC

You can use the following tools to monitor traffic or network access in your virtual private cloud (VPC).

VPC Flow Logs

You can use VPC Flow Logs to capture detailed information about the traffic going to and from network interfaces in your VPCs.

Amazon CloudWatch Internet Monitor

You can use Internet Monitor for visibility into how internet issues impact the performance and availability between your applications hosted on AWS and your end users. You can also explore, in near real-time, how to improve the projected latency of your application by switching to use other services, or by rerouting traffic to your workload through different AWS Regions. For more information, see Using Amazon CloudWatch Internet Monitor.

Amazon VPC IP Address Manager (IPAM)

You can use IPAM to plan, track, and monitor IP addresses for your workloads. For more information, see IP Address Manager.

Traffic Mirroring

You can use this feature to copy network traffic from a network interface of an Amazon EC2 instance and send it to out-of-band security and monitoring appliances for deep packet inspection. You can detect network and security anomalies, gain operational insights, implement compliance and security controls, and troubleshoot issues. For more information, see Traffic Mirroring.

Reachability Analyzer

You can use this tool to analyze and debug network reachability between two resources in your VPC. After you specify the source and destination resources, Reachability Analyzer produces hop-by-hop details of the virtual path between them when they are reachable, and identifies the blocking component when they are unreachable. For more information, see Reachability Analyzer.

Network Access Analyzer

You can use Network Access Analyzer to understand network access to your resources. This helps you identify improvements to your network security posture and demonstrate that your network meets specific compliance requirements. For more information, see Network Access Analyzer.

CloudTrail logs

AWS CloudTrail logs API calls for Amazon VPC, such as:

- Which API calls were made (such as actions like creating or modifying VPC resources)
- The source IP address of the call
- Who made the call
- When the call was made

Separate logs are created for `CreateVpc`, `DeleteVpc` and `CreateDefaultVpc` actions. These logs also include the default resources (like any default internet gateways or default security groups) created and associated with the VPC.

For more information, see Log Amazon EC2 API calls using AWS CloudTrail in the *Amazon EC2 User Guide*.

Logging IP traffic using VPC Flow Logs

VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data can be published to the following locations: Amazon CloudWatch Logs, Amazon S3, or Amazon Data Firehose. The configured delivery path and permissions that enable network traffic logs to be sent to a destination like CloudWatch Logs or S3 are referred to as *subscriptions*. After you create a flow log, you can retrieve and view the flow log records in the log group, bucket, or delivery stream that you configured.

Flow logs can help you with a number of tasks, such as:

- Diagnosing overly restrictive security group rules
- Monitoring the traffic that is reaching your instance
- Determining the direction of the traffic to and from the network interfaces

Flow log data is collected outside of the path of your network traffic, and therefore does not affect network throughput or latency. You can create or delete flow logs without any risk of impact to network performance.

> **Note**
>
> This section only talks about flow logs for VPCs. For information about flow logs for transit gateways introduced in version 6, see Logging network traffic using Transit Gateway Flow Logs in the *Amazon VPC Transit Gateways User Guide*.

Contents

- Flow logs basics
- Flow log records
- Flow log record examples
- Flow log limitations
- Pricing
- Work with flow logs
- Publish flow logs to CloudWatch Logs
- Publish flow logs to Amazon S3
- Publish flow logs to Amazon Data Firehose
- Query flow logs using Amazon Athena
- Troubleshoot VPC Flow Logs

Flow logs basics

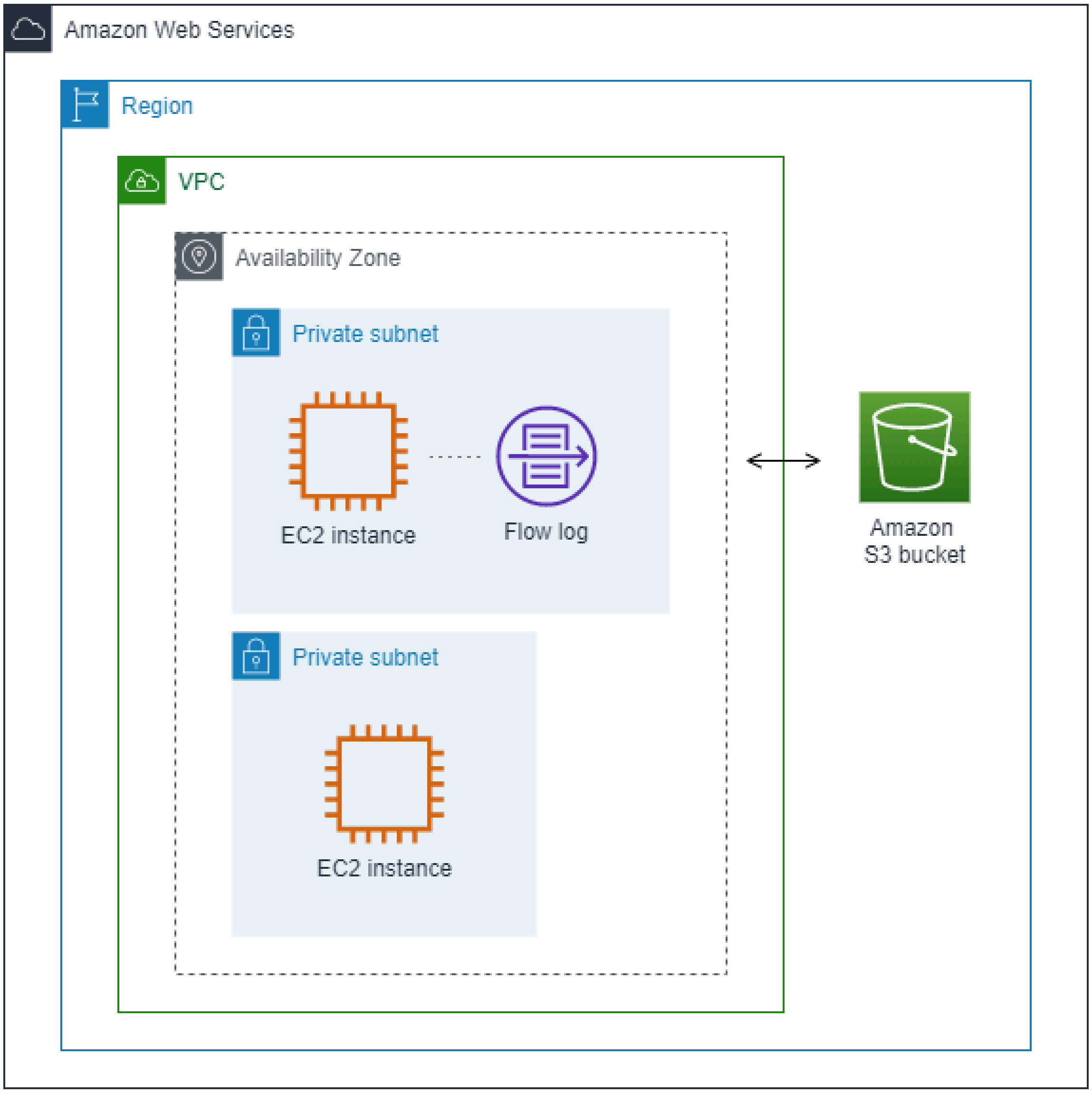
You can create a flow log for a VPC, a subnet, or a network interface. If you create a flow log for a subnet or VPC, each network interface in that subnet or VPC is monitored.

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.

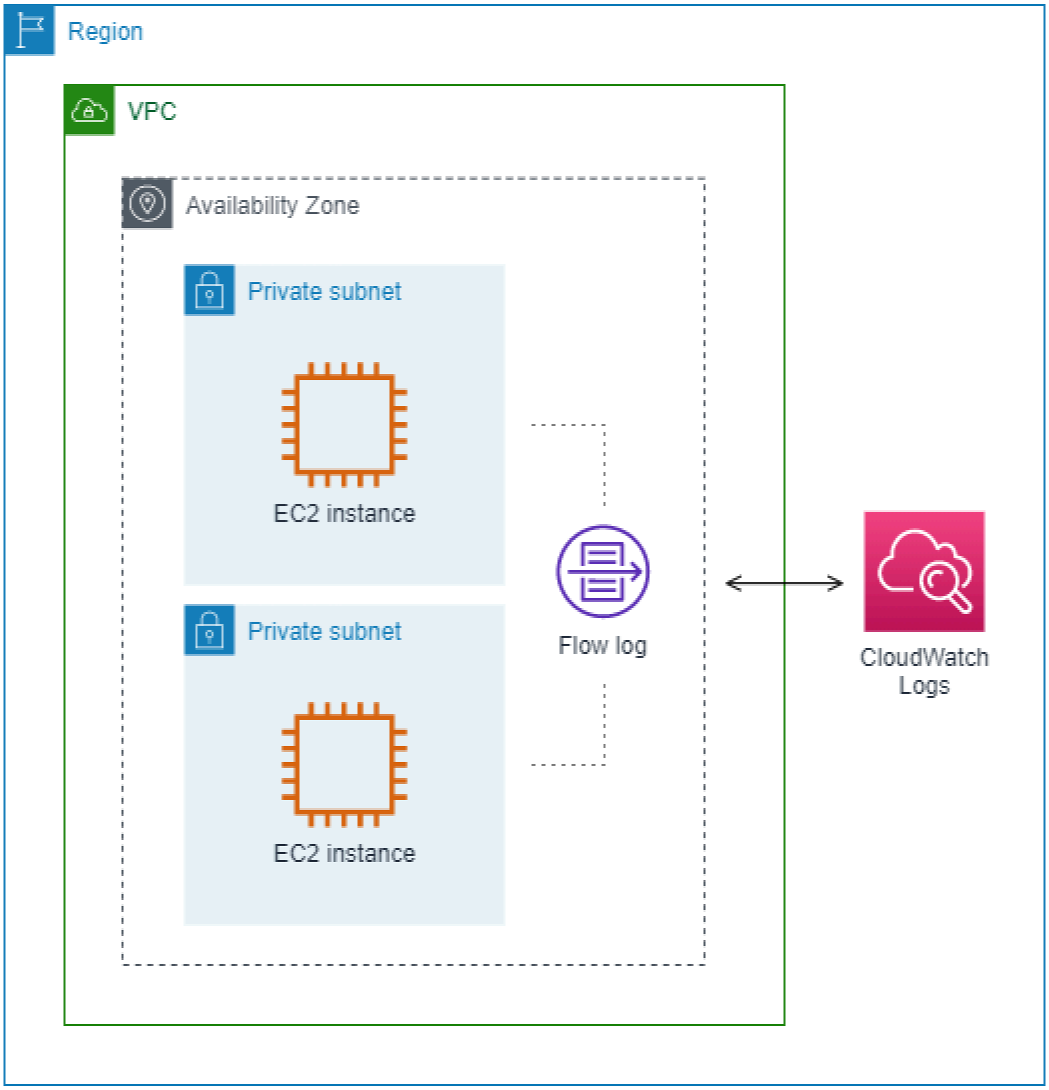
To create a flow log, you specify:

- The resource for which to create the flow log
- The type of traffic to capture (accepted traffic, rejected traffic, or all traffic)
- The destinations to which you want to publish the flow log data

In the following example, you create a flow log that captures accepted traffic for the network interface for one of the EC2 instances in a private subnet and publishes the flow log records to an Amazon S3 bucket.



In the following example, a flow log captures all traffic for a subnet and publishes the flow log records to Amazon CloudWatch Logs. The flow log captures traffic for all network interfaces in the subnet.



After you create a flow log, it can take several minutes to begin collecting and publishing data to the chosen destinations. Flow logs do not capture real-time log streams for your network interfaces. For more information, see 2. Create a flow log.

If you launch an instance into your subnet after you create a flow log for your subnet or VPC, we create a log stream (for CloudWatch Logs) or log file object (for Amazon S3) for the new network interface as soon as there is network traffic for the network interface.

You can create flow logs for network interfaces that are created by other AWS services, such as:

- Elastic Load Balancing
- Amazon RDS
- Amazon ElastiCache
- Amazon Redshift

- 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, so that no new flow log records are created or published. Deleting a flow log does not delete any existing flow log data. After you delete a flow log, you can delete the flow log data directly from the destination when you are finished with it. For more information, see 4. Delete a flow log.

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
- Default format
- Custom format
- Available fields

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 table. You cannot customize or change the default format. To capture additional fields or a different subset of fields, specify a custom format instead.

Custom format

With a custom format, you specify which fields are included in the flow log records and in which order. This enables you to create flow logs that are specific to your needs and to omit fields that are not relevant. Using a custom format can reduce the need for separate processes to extract specific information from the published flow logs. You can specify any number of the available flow log fields, but you must specify at least one.

Available fields

The following table describes all of the available fields for a flow log record. The **Version** column indicates the VPC Flow Logs version in which the field was introduced. The default format includes all version 2 fields, in the same order that they appear in the table.

When publishing flow log data to Amazon S3, the data type for the fields depends on the flow log format. If the format is plain text, all fields are of type STRING. If the format is Parquet, see the table for the field data types.

If a field is not applicable or could not be computed for a specific record, the record displays a '-' symbol for that entry. Metadata fields that do not come directly from the packet header are best effort approximations, and their values might be missing or inaccurate.

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. **Parquet data type:** INT_32	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 might display unknown for this field. **Parquet data type:** STRING	2
interface-id	The ID of the network interface for which the traffic is recorded. **Parquet data type:** STRING	2
srcaddr	For incoming traffic, this is the IP address of the source of traffic. For outgoing traffic, this is the private IPv4 address or the IPv6 address of the network interface sending the traffic. See also pkt-srcaddr. **Parquet data type:** STRING	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. **Parquet data type:** STRING	2
srcport	The source port of the traffic. **Parquet data type:** INT_32	2

Field	Description	Version
dstport	The destination port of the traffic. **Parquet data type:** INT_32	2
protocol	The IANA protocol number of the traffic. For more information, see Assigned Internet Protocol Numbers. **Parquet data type:** INT_32	2
packets	The number of packets transferred during the flow. **Parquet data type:** INT_64	2
bytes	The number of bytes transferred during the flow. **Parquet data type:** INT_64	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. **Parquet data type:** INT_64	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. **Parquet data type:** INT_64	2

Field	Description	Version
action	The action that is associated with the traffic: • ACCEPT — The traffic was accepted. • REJECT — The traffic was rejected. For example, the traffic was not allowed by the security groups or network ACLs, or packets arrived after the connection was closed. **Parquet data type:** STRING	2
log-status	The logging status of the flow log: • OK — Data is logging normally to the chosen destinations. • NODATA — There was no network traffic to or from the network interface during the aggregation interval. • SKIPDATA — Some flow log records were skipped during the aggregation interval. This might be because of an internal capacity constraint, or an internal error. Some flow log records may be skipped during the aggregati on interval (see *log-status* in Available fields). This may be caused by an internal AWS capacity constraint or internal error. If you are using AWS Cost Explorer to view VPC flow log charges and some flow logs are skipped during the flow log aggregati on interval, the number of flow logs reported in AWS Cost Explorer will be higher than the number of flow logs published by Amazon VPC. **Parquet data type:** STRING	2
vpc-id	The ID of the VPC that contains the network interface for which the traffic is recorded. **Parquet data type:** STRING	3

Field	Description	Version
subnet-id	The ID of the subnet that contains the network interface for which the traffic is recorded. **Parquet data type:** STRING	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. **Parquet data type:** STRING	3

Field	Description	Version
tcp-flags	The bitmask value for the following TCP flags: • FIN — 1 • SYN — 2 • RST — 4 • SYN-ACK — 18 If no supported flags are recorded, the TCP flag value is 0. For example, since tcp-flags does not support logging ACK or PSH flags, records for traffic with these unsupported flags will result in tcp-flags value 0. If, however, an unsupported flag is accompanied by a supported flag, we will report the value of the supported flag. For example, if ACK is a part of SYN-ACK, it reports 18. And if there is a record like SYN+ECE, since SYN is a supported flag and ECE is not, the TCP flag value is 2. If for some reason the flag combination is invalid and the value cannot be calculated, the value is '-'. If no flags are sent, the TCP flag value is 0. 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. For general information about TCP flags (such as the meaning of flags like FIN, SYN, and ACK), see TCP segment structure on Wikipedia. **Parquet data type:** INT_32	3
type	The type of traffic. The possible values are: IPv4 | IPv6 | EFA. For more information, see Elastic Fabric Adapter. **Parquet data type:** STRING	3

Field	Description	Version
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, or where the IP address of a pod in Amazon EKS is different from the IP address of the network interface of the instance node on which the pod is running (for communication within a VPC). **Parquet data type:** STRING	3
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, or where the IP address of a pod in Amazon EKS is different from the IP address of the network interface of the instance node on which the pod is running (for communication within a VPC). **Parquet data type:** STRING	3
region	The Region that contains the network interface for which traffic is recorded. **Parquet data type:** STRING	4
az-id	The ID of the Availability Zone that contains the network interface for which traffic is recorded. If the traffic is from a sublocation, the record displays a '-' symbol for this field. **Parquet data type:** STRING	4

Field	Description	Version
sublocation-type	The type of sublocation that's returned in the sublocation-id field. The possible values are: [wavelength](#) | [outpost](#) | [localzone](#). If the traffic is not from a sublocation, the record displays a '-' symbol for this field. **Parquet data type:** STRING	4
sublocation-id	The ID of the sublocation that contains the network interface for which traffic is recorded. If the traffic is not from a sublocation, the record displays a '-' symbol for this field. **Parquet data type:** STRING	4
pkt-src-aws-service	The name of the subset of [IP address ranges](#) for the pkt-srcaddr field, if the source IP address is for an AWS service. If the source IP address belongs to an [overlapped range](#), pkt-src-aws-service shows only one of the AWS service codes. The possible values are: `AMAZON` | `AMAZON_APPFLOW` | `AMAZON_CONNECT` | `API_GATEWAY` | `AURORA_DSQL` | `CHIME_MEETINGS` | `CHIME_VOICECONNECTOR` | `CLOUD9` | `CLOUDFRONT` | `CLOUDFRONT_ORIGIN_FACING` | `CODEBUILD` | `DYNAMODB` | `EBS` | `EC2` | `EC2_INSTANCE_CONNECT` | `GLOBALACCELERATOR` | `IVS_LOW_LATENCY` | `IVS_REALTIME` | `KINESIS_VIDEO_STREAMS` | `MEDIA_PACKAGE_V2` | `ROUTE53` | `ROUTE53_HEALTHCHECKS` | `ROUTE53_HEALTHCHECKS_PUBLISHING` | `ROUTE53_RESOLVER` | `S3` | `WORKSPACES_GATEWAYS` . **Parquet data type:** STRING	5
pkt-dst-aws-service	The name of the subset of IP address ranges for the pkt-dstaddr field, if the destination IP address is for an AWS service. For a list of possible values, see the pkt-src-aws-service field. **Parquet data type:** STRING	5

Field	Description	Version
flow-direction	The direction of the flow with respect to the interface where traffic is captured. The possible values are: ingress | egress. **Parquet data type:** STRING	5
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, including resources that create a network interface in the 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) **Parquet data type:** INT_32	5
ecs-cluster-arn	AWS Resource Name (ARN) of the ECS cluster if the traffic is from a running ECS task. To include this field in your subscription, you need permission to call ecs:ListClusters. **Parquet data type:** STRING	7
ecs-cluster-name	Name of the ECS cluster if the traffic is from a running ECS task. To include this field in your subscription, you need permission to call ecs:ListClusters. **Parquet data type:** STRING	7

Field	Description	Version
ecs-container-instance-arn	ARN of the ECS container instance if the traffic is from a running ECS task on an EC2 instance. If the capacity provider is AWS Fargate, this field will be '-'. To include this field in your subscript ion, you need permission to call ecs:ListClusters and ecs:ListC ontainerInstances. **Parquet data type:** STRING	7
ecs-container-instance-id	ID of the ECS container instance if the traffic is from a running ECS task on an EC2 instance. If the capacity provider is AWS Fargate, this field will be '-'. To include this field in your subscription, you need permission to call ecs:ListClusters and ecs:ListContainerI nstances. **Parquet data type:** STRING	7
ecs-container-id	Docker runtime ID of the container if the traffic is from a running ECS task. If there are one or more containers in the ECS task, this will be the docker runtime ID of the first container. To include this field in your subscription, you need permission to call ecs:ListC lusters. **Parquet data type:** STRING	7
ecs-second-container-id	Docker runtime ID of the container if the traffic is from a running ECS task. If there are more than one containers in the ECS task, this will be the Docker runtime ID of the second container. To include this field in your subscription, you need permission to call ecs:ListClusters. **Parquet data type:** STRING	7
ecs-service-name	Name of the ECS service if the traffic is from a running ECS task and the ECS task is started by an ECS service. If the ECS task is not started by an ECS service, this field will be '-'. To include this field in your subscription, you need permission to call ecs:ListClusters and ecs:ListServices. **Parquet data type:** STRING	7

Field	Description	Version
ecs-task-definition-arn	ARN of the ECS task definition if the traffic is from a running ECS task. To include this field in your subscription, you need permission to call ecs:ListClusters and ecs:ListTaskDefinitions **Parquet data type:** STRING	7
ecs-task-arn	ARN of the ECS task if the traffic is from a running ECS task. To include this field in your subscription, you need permission to call ecs:ListClusters and ecs:ListTasks. **Parquet data type:** STRING	7
ecs-task-id	ID of the ECS task if the traffic is from a running ECS task. To include this field in your subscription, you need permission to call ecs:ListClusters and ecs:ListTasks. **Parquet data type:** STRING	7
reject-reason	Reason why traffic was rejected. Possible values: BPA. Returns a '-' for any other reject reason. For more information about VPC Block Public Access (BPA), see Block public access to VPCs and subnets. **Parquet data type:** STRING	8

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. For information about how to create flow logs, see Work with flow logs.

Contents

- Accepted and rejected traffic
- No data and skipped records
- Security group and network ACL rules
- IPv6 traffic
- TCP flag sequence
- Traffic through a NAT gateway

- Traffic through a transit gateway
- Service name, traffic path, and flow direction

Accepted and rejected traffic

The following are examples of default flow log records.

In this example, SSH traffic (destination port 22, TCP protocol) from IP address 172.31.16.139 to network interface with private IP address is 172.31.16.21 and ID 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
```

VPC Flow Logs skips records when it can't capture flow log data during an aggregation interval because it exceeds internal capacity. A single skipped record can represent multiple flows that were not captured for the network interface during the aggregation interval.

```
2 123456789010 eni-11111111aaaaaaaaa - - - - - - - 1431280876 1431280934 - SKIPDATA
```

Note

Some flow log records may be skipped during the aggregation interval (see *log-status* in Available fields). This may be caused by an internal AWS capacity constraint or internal error. If you are using AWS Cost Explorer to view VPC flow log charges and some flow logs are skipped during the flow log aggregation interval, the number of flow logs reported in AWS Cost Explorer will be higher than the number of flow logs published by Amazon VPC.

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

This section contains examples of custom flow logs that capture the following fields in the following order.

```
version vpc-id subnet-id instance-id interface-id account-id type srcaddr dstaddr
 srcport dstport pkt-srcaddr pkt-dstaddr protocol bytes packets start end action tcp-
flags log-status
```

The tcp-flags field in the examples in this section are represented by the second-to-last value in the flow log. TCP flags can help you identify the direction of the traffic, for example, which server initiated the connection.

> **Note**
>
> For more information about the tcp-flags option and an explanation of each of the TCP flags, see [Available fields](#).

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 server sent a FIN flag (1) to the client for the connection on port 43418. The client responded with a FIN to the server on port 43418.

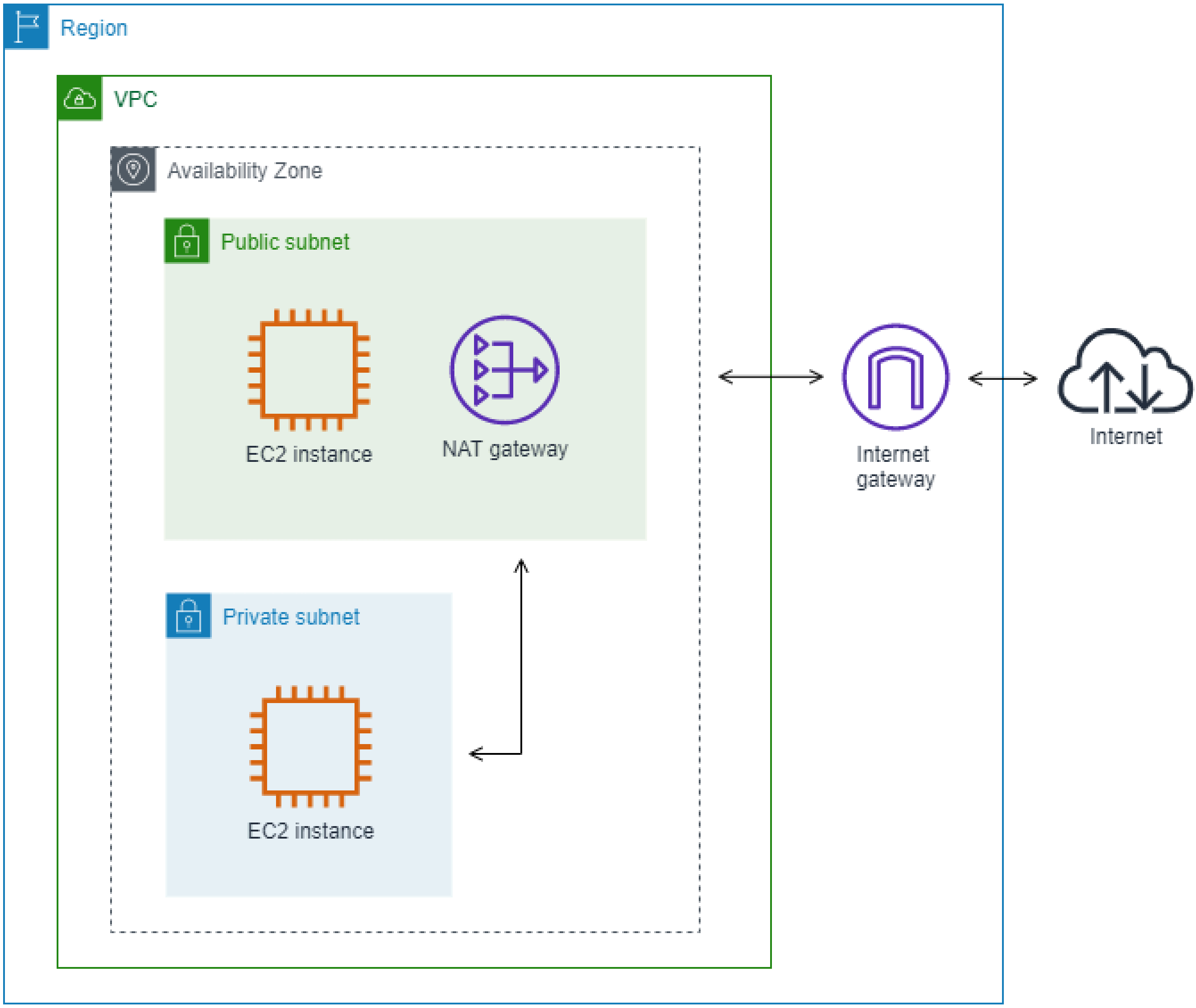
```
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456
 eni-1235b8ca123456789 123456789010 IPv4 10.0.0.62 52.213.180.42 5001 43418 10.0.0.62
 52.213.180.42 6 63388 1219 1566848933 1566849113 ACCEPT 1 OK
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456
 eni-1235b8ca123456789 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.

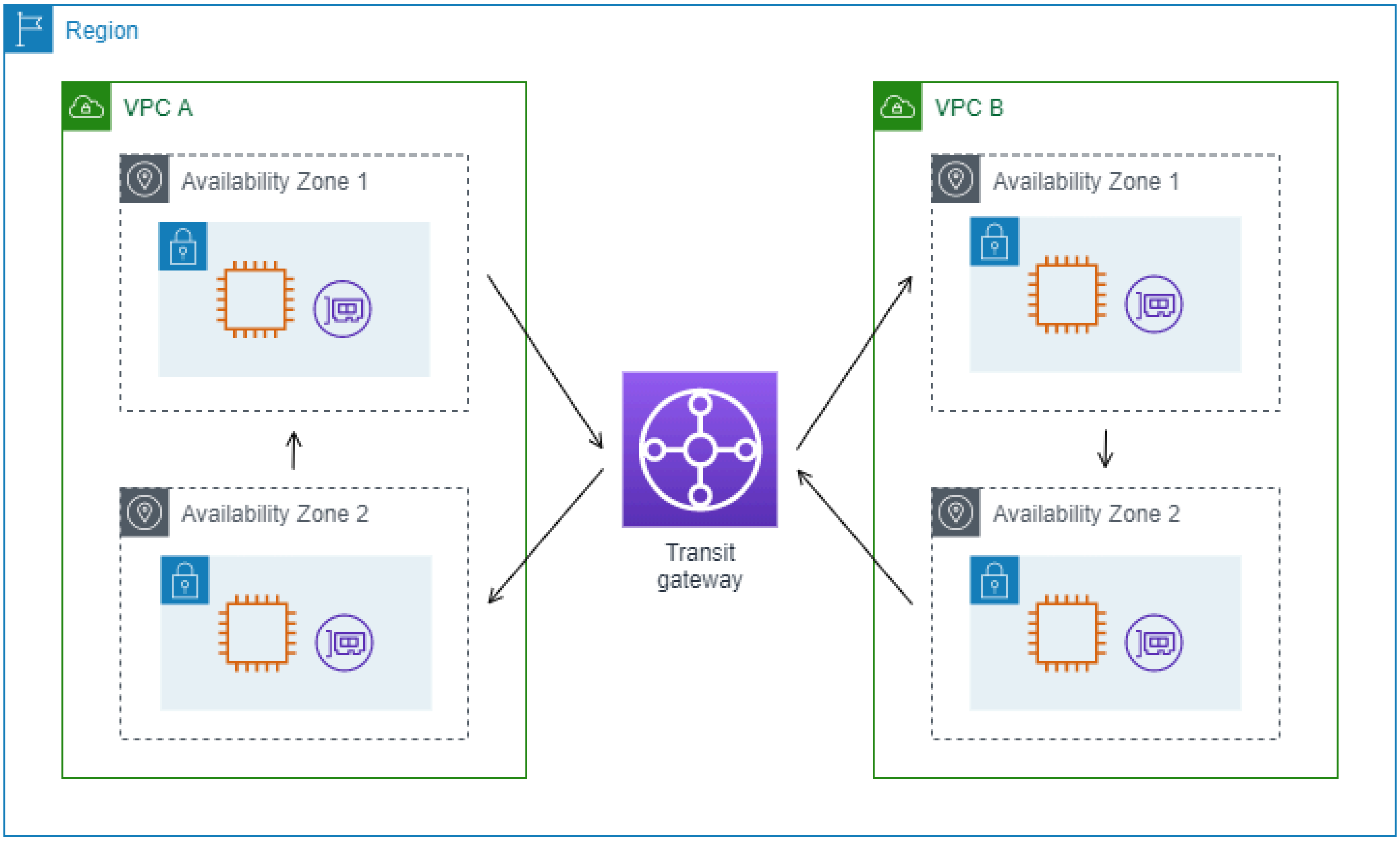
```
- eni-1235b8ca123456789 10.0.0.220 10.0.1.5 203.0.113.5 10.0.1.5
```

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. Traffic arrives at the server in VPC B using one elastic network interface ID (in this example, let's say the ID is eni-11111111111111111) and leaves VPC B using another (for example 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:

- After you create a flow log, you won't see flow log data until there is active traffic for the network interface, subnet, or VPC that you selected.
- You can't enable flow logs for VPCs that are peered with your VPC unless the peer VPC is in your account.
- After you create a flow log, you can't 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, when the log record is for an egress flow, 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 the log record is for an ingress flow into the network interface, the primary private IP of the network interface will not be shown in the `srcaddr` field.
- 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.
- For `pkt-srcaddr` and `pkt-dstaddr` fields, if the intermediate layer has Client IP address Preservation enabled, this field may show the preserved Client IP instead of the IP address of the intermediate layer.
- Some flow log records may be skipped during the aggregation interval (see *log-status* in Available fields). This may be caused by an internal AWS capacity constraint or internal error. If you are using AWS Cost Explorer to view VPC flow log charges and some flow logs are skipped during the flow log aggregation interval, the number of flow logs reported in AWS Cost Explorer will be higher than the number of flow logs published by Amazon VPC.
- If you are using VPC Block Public Access (BPA):
    - Flow logs for VPC BPA do not include skipped records.

- Flow logs for VPC BPA do not include `bytes` even if you include the `bytes` field in your flow log.

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 mirrored source traffic. You will see traffic mirrored target traffic only.
- Traffic to the reserved IP address for the default VPC router.
- Traffic between an endpoint network interface and a Network Load Balancer network interface.
- Address Resolution Protocol (ARP) traffic.

Limitations specific to ECS fields available in version 7:

- To create flow log subscriptions with ECS fields, your account must contain at least one ECS cluster.
- ECS fields are not computed if the underlying ECS tasks are not owned by the owner of the flow log subscription. For example, if you share a subnet (`SubnetA`) with another account (`AccountB`), and then you create a flow log subscription for `SubnetA`, if `AccountB` launches ECS tasks in the shared subnet, your subscription will receive traffic logs from ECS tasks launched by `AccountB` but the ECS fields for these logs will not be computed due to security concerns.
- If you create flow log subscriptions with ECS fields at the VPC/Subnet resource level, any traffic generated for non-ECS network interfaces will also be delivered for your subscriptions. The values for ECS fields will be '-' for non-ECS IP traffic. For example, you have a subnet (`subnet-000000`) and you create a flow log subscription for this subnet with ECS fields (`fl-00000000`). In `subnet-000000`, you launch an EC2 instance (`i-0000000`) that is connected to the internet and is actively generating IP traffic. You also launch a running ECS task (`ECS-Task-1`) in the same subnet. Since both `i-0000000` and `ECS-Task-1` are generating IP traffic, your flow log subscription `fl-00000000` will deliver traffic logs for both entities. However, only

`ECS-Task-1` will have actual ECS metadata for the ECS fields you included in your logFormat. For `i-0000000` related traffic, these fields will have a value of '-'.

- `ecs-container-id` and `ecs-second-container-id` are ordered as the VPC Flow Logs service receives them from the ECS event stream. They are not guaranteed to be in the same order as you see them on ECS console or in the DescribeTask API call. If a container enters a STOPPED status while the task is still running, it may continue to appear in your log.
- The ECS metadata and IP traffic logs are from two different sources. We start computing your ECS traffic as soon as we obtain all required information from upstream dependencies. After you start a new task, we start computing your ECS fields 1) when we receive IP traffic for the underlying network interface and 2) when we receive the ECS event that contains the metadata for your ECS task to indicate the task is now running. After you stop a task, we stop computing your ECS fields 1) when we no longer receive IP traffic for the underlying network interface or we receive IP traffic that is delayed for more than one day and 2) when we receive the ECS event that contains the metadata for your ECS task to indicate your task is no longer running.
- Only ECS tasks launched in `awsvpc` network mode are supported.

Pricing

Data ingestion and archival charges for vended logs apply when you publish flow logs. For more information about pricing when publishing vended logs, open Amazon CloudWatch Pricing, select **Logs** and find **Vended Logs**.

To track charges from publishing flow logs, you can apply cost allocation tags to your destination resource. Thereafter, your AWS cost allocation report includes usage and costs aggregated by these tags. You can apply tags that represent business categories (such as cost centers, application names, or owners) to organize your costs. For more information, see the following:

- Using Cost Allocation Tags in the *AWS Billing User Guide*
- Tag log groups in Amazon CloudWatch Logs in the *Amazon CloudWatch Logs User Guide*
- Using cost allocation S3 bucket tags in the *Amazon Simple Storage Service User Guide*
- Tagging Your Delivery Streams in the *Amazon Data Firehose Developer Guide*

Work with flow logs

You can work with flow logs using consoles for Amazon EC2 and Amazon VPC.

Tasks

- 1. Control the use of flow logs with IAM
- 2. Create a flow log
- 3. Tag a flow log
- 4. Delete a flow log
- Command line overview

1. Control the use of flow logs with IAM

By default, users do not have permission to work with flow logs. You can create an IAM role with a policy attached that grants users the permissions to create, describe, and delete flow logs.

The following is an example policy that grants users full permissions to create, describe, and delete flow logs.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteFlowLogs",
        "ec2:CreateFlowLogs",
        "ec2:DescribeFlowLogs"
      ],
      "Resource": "*"
    }
  ]
}
```

For more information, see the section called “How Amazon VPC works with IAM”.

2. Create a flow log

You can create flow logs for your VPCs, subnets, or network interfaces. When you create a flow log, you must specify a destination for the flow log. For more information, see the following:

- the section called "Create a flow log that publishes to CloudWatch Logs"
- the section called "Create a flow log that publishes to Amazon S3"
- the section called "Create a flow log that publishes to Amazon Data Firehose"

3. Tag a flow log

You can add or remove tags for a flow log at any time.

To manage tags for a flow log

1. Do one of the following:
   - Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. In the navigation pane, choose **Network Interfaces**. Select the checkbox for the network interface.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Your VPCs**. Select the checkbox for the VPC.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Subnets**. Select the checkbox for the subnet.
2. Choose **Flow Logs**.
3. Choose **Actions**, **Manage tags**.
4. To add a new tag, choose **Add new tag** and enter the key and value. To remove a tag, choose **Remove**.
5. When you are finished adding or removing tags, choose **Save**.

4. Delete a flow log

You can delete a flow log at any time. After you delete a flow log, it can take several minutes to stop collecting data.

Deleting a flow log does not delete the log data from the destination or modify the destination resource. You must delete the existing flow log data directly from the destination, and clean up the destination resource, using the console for the destination service.

To delete a flow log

1. Do one of the following:

- Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. In the navigation pane, choose **Network Interfaces**. Select the checkbox for the network interface.
- Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Your VPCs**. Select the checkbox for the VPC.
- Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Subnets**. Select the checkbox for the subnet.

2. Choose **Flow Logs**.
3. Choose **Actions**, **Delete flow logs**.
4. When prompted for confirmation, type **`delete`** and then choose **Delete**.

Command line overview

You can perform the tasks described on this page using the command line.

Create a flow log

- create-flow-logs (AWS CLI)
- New-EC2FlowLog (AWS Tools for Windows PowerShell)

Describe a flow log

- describe-flow-logs (AWS CLI)
- Get-EC2FlowLog (AWS Tools for Windows PowerShell)

Tag a flow log

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

Delete a flow log

- delete-flow-logs (AWS CLI)
- Remove-EC2FlowLog (AWS Tools for Windows PowerShell)

Publish flow logs to CloudWatch Logs

Flow logs can publish flow log data directly to Amazon CloudWatch. Amazon CloudWatch is a comprehensive monitoring and observability service. It collects and tracks metrics, logs, and event data from various AWS resources, as well as your own applications and services. CloudWatch provides visibility into resource utilization, application performance, and operational health, enabling you to detect and respond to system-wide performance changes and potential issues. With CloudWatch, you can set alarms, visualize logs and metrics, and automatically react to collect and optimize your cloud resources. It is an essential tool for ensuring the reliability, availability, and performance of your cloud-based infrastructure and applications.

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.

For more information about CloudWatch Logs, see Logs sent to CloudWatch Logs in the *Amazon CloudWatch Logs User Guide*.

Pricing

Data ingestion and archival charges for vended logs apply when you publish flow logs to CloudWatch Logs. For more information, open Amazon CloudWatch Pricing, select **Logs** and find **Vended Logs**.

Contents

- IAM role for publishing flow logs to CloudWatch Logs
- Create a flow log that publishes to CloudWatch Logs
- View flow log records with CloudWatch Logs
- Search flow log records

- Process flow log records in CloudWatch Logs

IAM role for publishing flow logs to CloudWatch Logs

The IAM role that's associated with your flow log must have sufficient permissions to publish flow logs to the specified log group in CloudWatch Logs. The IAM role must belong to your AWS account.

The IAM policy that's attached to your IAM role must include at least the following permissions.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": "*"
    }
  ]
}
```

Ensure that your role has the following trust policy, which allows the flow logs service to assume the role.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
        "Principal": {
          "Service": "vpc-flow-logs.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
      }
    ]
  }
```

We recommend that you use the `aws:SourceAccount` and `aws:SourceArn` condition keys to protect yourself against [the confused deputy problem](). For example, you could add the following condition block to the previous trust policy. The source account is the owner of the flow log and the source ARN is the flow log ARN. If you don't know the flow log ID, you can replace that portion of the ARN with a wildcard (*) and then update the policy after you create the flow log.

```
"Condition": {
    "StringEquals": {
        "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
        "aws:SourceArn": "arn:aws:ec2:region:account_id:vpc-flow-log/flow-log-id"
    }
}
```

Create an IAM role for flow logs

You can update an existing role as described above. Alternatively, you can use the following procedure to create a new role for use with flow logs. You'll specify this role when you create the flow log.

To create an IAM role for flow logs

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. On the **Create policy** page, do the following:

    a. Choose **JSON**.

    b. Replace the contents of this window with the permissions policy at the start of this section.

c. Choose **Next**.

d. Enter a name for your policy and an optional description and tags, and then choose **Create policy**.

5. In the navigation pane, choose **Roles**.
6. Choose **Create role**.
7. For **Trusted entity type**, choose **Custom trust policy**. For **Custom trust policy**, replace `"Principal": {},` with the following, then and choose **Next**.

```
"Principal": {
   "Service": "vpc-flow-logs.amazonaws.com"
},
```

8. On the **Add permissions** page, select the checkbox for the policy that you created earlier in this procedure, and then choose **Next**.
9. Enter a name for your role and optionally provide a description.
10. Choose **Create role**.

Create a flow log that publishes to CloudWatch Logs

You can create flow logs for your VPCs, subnets, or network interfaces. If you perform these steps as a user using a particular IAM role, ensure that the role has permissions to use the `iam:PassRole` action.

Prerequisite

Verify that the IAM principal that you are using to make the request has permissions to call the `iam:PassRole` action.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
```

```
            ],
            "Resource": "arn:aws:iam::111122223333:role/flow-log-role-name"
        }
    ]
}
```

To create a flow log using the console

1. Do one of the following:
   - Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. In the navigation pane, choose **Network Interfaces**. Select the checkbox for the network interface.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Your VPCs**. Select the checkbox for the VPC.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Subnets**. Select the checkbox for the subnet.
2. Choose **Actions**, **Create flow log**.
3. For **Filter**, specify the type of traffic to log. Choose **All** to log accepted and rejected traffic, **Reject** to log only rejected traffic, or **Accept** to log only accepted traffic.
4. For **Maximum aggregation interval**, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.
5. For **Destination**, choose **Send to CloudWatch Logs**.
6. For **Destination log group**, choose the name of an existing log group or enter the name of a new log group. If you enter a name, we create the log group when there is traffic to log.
7. For **Service access**, choose an existing IAM service role that has permissions to publish logs to CloudWatch Logs or choose to create a new service role.
8. For **Log record format**, select the format for the flow log record.
   - To use the default format, choose **AWS default format**.
   - To use a custom format, choose **Custom format** and then select fields from **Log format**.
9. For **Additional metadata**, select if you want to include metadata from Amazon ECS in the log format.
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-EC2FlowLog (AWS Tools for Windows PowerShell)

The following AWS CLI example creates a flow log that captures all accepted traffic for the specified subnet. The flow logs are delivered to the specified log group. The `--deliver-logs-permission-arn` parameter specifies the IAM role required to publish to CloudWatch Logs.

```
aws ec2 create-flow-logs --resource-type Subnet --resource-ids subnet-1a2b3c4d --
traffic-type ACCEPT --log-group-name my-flow-logs --deliver-logs-permission-arn
 arn:aws:iam::123456789101:role/publishFlowLogs
```

View flow log records with CloudWatch Logs

You can view your flow log records using the CloudWatch Logs console. After you create your flow log, it might take a few minutes for it to be visible in the console.

To view flow log records published to CloudWatch Logs using the console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Logs**, **Log groups**.
3. Select the name of the log group that contains your flow logs to open its details page.
4. Select the name of the log stream that contains the flow log records. For more information, see Flow log records.

To view flow log records published to CloudWatch Logs using the command line

- get-log-events (AWS CLI)
- Get-CWLLogEvent (AWS Tools for Windows PowerShell)

Search flow log records

You can search your flow log records that are published to CloudWatch Logs using the CloudWatch Logs console. You can use metric filters to filter flow log records. Flow log records are space delimited.

To search flow log records using the CloudWatch Logs console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Logs**, **Log groups**.
3. Select the log group that contains your flow log, and then select the log stream, if you know the network interface that you are searching for. Alternatively, choose **Search log group**. This might take some time if there are many network interfaces in your log group, or depending on the time range that you select.
4. Under **Filter events**, enter the string below. This assumes that the flow log record uses the default format.

```
[version, accountid, interfaceid, srcaddr, dstaddr, srcport, dstport, protocol,
 packets, bytes, start, end, action, logstatus]
```

5. Modify the filter as needed by specifying values for the fields. The following examples filter by specific source IP addresses.

```
[version, accountid, interfaceid, srcaddr = 10.0.0.1, dstaddr, srcport, dstport,
 protocol, packets, bytes, start, end, action, logstatus]
[version, accountid, interfaceid, srcaddr = 10.0.2.*, dstaddr, srcport, dstport,
 protocol, packets, bytes, start, end, action, logstatus]
```

The following examples filter by destination port, the number of bytes, and whether the traffic was rejected.

```
[version, accountid, interfaceid, srcaddr, dstaddr, srcport, dstport = 80 ||
 dstport = 8080, protocol, packets, bytes, start, end, action, logstatus]
[version, accountid, interfaceid, srcaddr, dstaddr, srcport, dstport = 80 ||
 dstport = 8080, protocol, packets, bytes >= 400, start, end, action = REJECT,
 logstatus]
```

Process flow log records in CloudWatch Logs

You can process 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 [Creating metrics from log events using filter]() in the *Amazon CloudWatch Logs 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**, **Log groups**.
3. Select the check box for the log group, and then choose **Actions**, **Create metric filter**.
4. For **Filter pattern**, enter the following string.

```
[version, account, eni, source, destination, srcport, destport="22", protocol="6",
 packets, bytes, windowstart, windowend, action="REJECT", flowlogstatus]
```

5. For **Select log data to test**, select the log stream for your network interface. (Optional) To view the lines of log data that match the filter pattern, choose **Test pattern**.
6. When you're ready, choose **Next**.
7. Enter a filter name, metric namespace, and metric name. Set the metric value to 1. When you're done, choose **Next** and then choose **Create metric filter**.
8. In the navigation pane, choose **Alarms**, **All alarms**.
9. Choose **Create alarm**.
10. Select the metric name that you created and then choose **Select metric**.
11. Configure the alarm as follows, and then choose **Next**:

   - For **Statistic**, choose **Sum**. This ensure that you capture the total number of data points for the specified time period.
   - For **Period**, choose **1 hour**.

- For **Whenever TimeSinceLastActive is...**, choose **Greater/Equal** and enter 10 for the threshold.
- For **Additional configuration**, **Datapoints to alarm**, leave the default of 1.

12. Choose **Next**.
13. For **Notification**, select an existing SNS topic or choose **Create new topic** to create a new one. Choose **Next**.
14. Enter a name and description for the alarm and choose **Next**.
15. When you are done previewing the alarm, choose **Create alarm**.

Publish flow logs to Amazon S3

Flow logs can publish flow log data to Amazon S3. Amazon S3 (Simple Storage Service) is a highly scalable and durable object storage service. It is designed to store and retrieve any amount of data, from anywhere on the web. S3 offers industry-leading durability and availability, with built-in features for data versioning, encryption, and access control.

When publishing to Amazon S3, flow log data is published to an existing Amazon S3 bucket that you specify. Flow log records for all of the monitored network interfaces are published to a series of log file objects that are stored in the bucket. If the flow log captures data for a VPC, the flow log publishes flow log records for all of the network interfaces in the selected VPC.

To create an Amazon S3 bucket for use with flow logs, see Create a bucket in the *Amazon S3 User Guide*.

For more information about how to streamline VPC flow log ingestion, flow log processing, and flow log visualization, see Centralized Logging with OpenSearch in the AWS Solutions Library.

For more information about CloudWatch Logs, see Logs sent to Amazon S3 in the *Amazon CloudWatch Logs User Guide*.

Pricing

Data ingestion and archival charges for vended logs apply when you publish flow logs to Amazon S3. For more information, open Amazon CloudWatch Pricing, select **Logs** and find **Vended Logs**.

Contents

- Flow log files

- Amazon S3 bucket permissions for flow logs
- Required key policy for use with SSE-KMS
- Amazon S3 log file permissions
- Create a flow log that publishes to Amazon S3
- View flow log records with Amazon S3

Flow log files

VPC Flow Logs collects data about the IP traffic going to and from your VPC into log records, aggregates those records into log files, and then publishes the log files to the Amazon S3 bucket at 5-minute intervals. Multiple files may be published and each log file may contain some or all of the flow log records for the IP traffic recorded in the previous 5 minutes.

In Amazon S3, the **Last modified** field for the flow log file indicates the date and time at which the file was uploaded to the Amazon S3 bucket. This is later than the timestamp in the file name, and differs by the amount of time taken to upload the file to the Amazon S3 bucket.

Log file format

You can specify one of the following formats for the log files. Each file is compressed into a single Gzip file.

- **Text** – Plain text. This is the default format.
- **Parquet** – Apache Parquet is a columnar data format. Queries on data in Parquet format are 10 to 100 times faster compared to queries on data in plain text. Data in Parquet format with Gzip compression takes 20 percent less storage space than plain text with Gzip compression.

> **Note**
>
> If data in Parquet format with Gzip compression is less than 100 KB per aggregation period, storing data in Parquet format may take up more space than plain text with Gzip compression due to Parquet file memory requirements.

Log file options

You can optionally specify the following options.

- **Hive-compatible S3 prefixes –** Enable Hive-compatible prefixes instead of importing partitions into your Hive-compatible tools. Before you run queries, use the **MSCK REPAIR TABLE** command.
- **Hourly partitions –** If you have a large volume of logs and typically target queries to a specific hour, you can get faster results and save on query costs by partitioning logs on an hourly basis.

Log file S3 bucket structure

Log files are saved to the specified Amazon S3 bucket using a folder structure that is based on the flow log's ID, Region, creation date, and destination options.

By default, the files are delivered to the following location.

```
bucket-and-optional-prefix/AWSLogs/account_id/vpcflowlogs/region/year/month/day/
```

If you enable Hive-compatible S3 prefixes, the files are delivered to the following location.

```
bucket-and-optional-prefix/AWSLogs/aws-account-id=account_id/aws-service=vpcflowlogs/
aws-region=region/year=year/month=month/day=day/
```

If you enable hourly partitions, the files are delivered to the following location.

```
bucket-and-optional-prefix/AWSLogs/account_id/vpcflowlogs/region/year/month/day/hour/
```

If you enable Hive-compatible partitions and partition the flow log per hour, the files are delivered to the following location.

```
bucket-and-optional-prefix/AWSLogs/aws-account-id=account_id/aws-service=vpcflowlogs/
aws-region=region/year=year/month=month/day=day/hour=hour/
```

Log file names

The file name of a log file is based on the flow log ID, Region, and creation date and time. File names use the following format.

```
aws_account_id_vpcflowlogs_region_flow_log_id_YYYYMMDDTHHmmZ_hash.log.gz
```

The following is an example of a log file for a flow log created by AWS account 123456789012, for a resource in the us-east-1 Region, on June 20, 2018 at 16:20 UTC. The file contains the flow log records with an end time between 16:20:00 and 16:24:59.

```
123456789012_vpcflowlogs_us-east-1_fl-1234abcd_20180620T1620Z_fe123456.log.gz
```

Amazon S3 bucket permissions for flow logs

By default, Amazon S3 buckets and the objects they contain are private. Only the bucket owner can access the bucket and the objects stored in it. However, the bucket owner can grant access to other resources and users by writing an access policy.

If the user creating the flow log owns the bucket and has `PutBucketPolicy` and `GetBucketPolicy` permissions for the bucket, we automatically attach the following policy to the bucket. This policy overwrites any existing policy attached to the bucket.

Otherwise, the bucket owner must add this policy to the bucket, specifying the AWS account ID of the flow log creator, or flow log creation fails. For more information, see Using bucket policies in the *Amazon Simple Storage Service User Guide*.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:*"
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryAclCheck",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:*"
                }
            }
        }
    ]
}
```

The ARN that you specify for *my-s3-arn* depends on whether you use Hive-compatible S3 prefixes.

- Default prefixes

```
arn:aws:s3:::bucket_name/optional_folder/AWSLogs/account_id/*
```

- Hive-compatible S3 prefixes

```
arn:aws:s3:::bucket_name/optional_folder/AWSLogs/aws-account-id=account_id/*
```

It is a best practice to grant these permissions to the log delivery service principal instead of individual AWS account ARNs. It is also a best practice to use the `aws:SourceAccount` and `aws:SourceArn` condition keys to protect against the confused deputy problem. The source account is the owner of the flow log and the source ARN is the wildcard (*) ARN of the logs service.

Note that the log delivery service calls the `HeadBucket` Amazon S3 API action to verify the existence and location of the S3 bucket. You are not required to grant the log delivery service permission to call this action; it will still deliver VPC flow logs even if it can't confirm that the S3 bucket exists and its location. However, there will be an `AccessDenied` error for the call to `HeadBucket` in your CloudTrail logs.

Required key policy for use with SSE-KMS

You can protect the data in your Amazon S3 bucket by enabling either Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3) or Server-Side Encryption with KMS Keys (SSE-KMS) on your S3 bucket. For more information, see Protecting data using server-side encryption in the *Amazon S3 User Guide*.

If you choose SSE-S3, no additional configuration is required. Amazon S3 handles the encryption key.

If you choose SSE-KMS, you must use a customer managed key ARN. If you use a key ID, you can run into a LogDestination undeliverable error when creating a flow log. Also, you must update the key policy for your customer managed key so that the log delivery account can write to your S3 bucket. For more information about the required key policy for use with SSE-KMS, see Amazon S3 bucket server-side encryption in the *Amazon CloudWatch Logs User Guide*.

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 S3 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 network interfaces, subnets, and VPCs.

Prerequisite

The IAM principal that creates the flow log must be using an IAM role that has the following permissions, which are required to publish flow logs to the destination Amazon S3 bucket.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
      {
        "Effect": "Allow",
        "Action": [
          "logs:CreateLogDelivery",
          "logs:DeleteLogDelivery"
        ],
        "Resource": "*"
      }
    ]
  }
```

To create a flow log using the console

1. Do one of the following:
   - Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. In the navigation pane, choose **Network Interfaces**. Select the checkbox for the network interface.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Your VPCs**. Select the checkbox for the VPC.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Subnets**. Select the checkbox for the subnet.
2. Choose **Actions**, **Create flow log**.
3. For **Filter**, specify the type of IP traffic data to log.
   - **Accept** – Log only accepted traffic.
   - **Reject** – Log only rejected traffic.
   - **All** – Log accepted and rejected traffic.
4. For **Maximum aggregation interval**, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.
5. For **Destination**, choose **Send to an Amazon S3 bucket**.
6. For **S3 bucket ARN**, specify the Amazon Resource Name (ARN) of an existing Amazon S3 bucket. You can optionally include a subfolder. For example, to specify a subfolder named `my-logs` in a bucket named `my-bucket`, use the following ARN:

   ```
   arn:aws:s3:::my-bucket/my-logs/
   ```

   The bucket cannot use `AWSLogs` as a subfolder name, as this is a reserved term.

If you own the bucket, we automatically create a resource policy and attach it to the bucket. For more information, see Amazon S3 bucket permissions for flow logs.

7. For **Log record format**, specify the format for the flow log record.
   - To use the default flow log record format, choose **AWS default format**.
   - To create a custom format, choose **Custom format**. For **Log format**, choose the fields to include in the flow log record.
8. For **Additional metadata**, select if you want to include metadata from Amazon ECS in the log format.
9. For **Log file format**, specify the format for the log file.
   - **Text –** Plain text. This is the default format.
   - **Parquet –** Apache Parquet is a columnar data format. Queries on data in Parquet format are 10 to 100 times faster compared to queries on data in plain text. Data in Parquet format with Gzip compression takes 20 percent less storage space than plain text with Gzip compression.
10. (Optional) To use Hive-compatible S3 prefixes, choose **Hive-compatible S3 prefix**, **Enable**.
11. (Optional) To partition your flow logs per hour, choose **Every 1 hour (60 mins)**.
12. (Optional) To add a tag to the flow log, choose **Add new tag** and specify the tag key and value.
13. Choose **Create flow log**.

To create a flow log that publishes to Amazon S3 using the command line

Use one of the following commands:

- create-flow-logs (AWS CLI)
- New-EC2FlowLog (AWS Tools for Windows PowerShell)

The following AWS CLI example creates a flow log that captures all traffic for the specified VPC and delivers the flow logs to the specified Amazon S3 bucket. The `--log-format` parameter specifies a custom format for the flow log records.

```
aws ec2 create-flow-logs --resource-type VPC --resource-ids vpc-00112233344556677 --
traffic-type ALL --log-destination-type s3 --log-destination arn:aws:s3:::flow-log-
bucket/custom-flow-logs/ --log-format '${version} ${vpc-id} ${subnet-id} ${instance-
```

```
id} ${srcaddr} ${dstaddr} ${srcport} ${dstport} ${protocol} ${tcp-flags} ${type} ${pkt-
srcaddr} ${pkt-dstaddr}'
```

View flow log records with Amazon S3

You can view your flow log records using the Amazon S3 console. After you create your flow log, it might take a few minutes for it to be visible in the console.

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.

To view flow log records published to Amazon S3

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Select the name of the bucket to open its details page.
3. Navigate to the folder with the log files. For example, *prefix*/AWSLogs/*account_id*/vpcflowlogs/*region*/*year*/*month*/*day*/.
4. Select the checkbox next to the file name, and then choose **Download**.

You can also query the flow log records in the log files using Amazon Athena. Amazon Athena is an interactive query service that makes it easier to analyze data in Amazon S3 using standard SQL. For more information, see Querying Amazon VPC Flow Logs in the *Amazon Athena User Guide*.

Publish flow logs to Amazon Data Firehose

Flow logs can publish flow log data directly to Amazon Data Firehose. Amazon Data Firehose is a fully managed service that collects, transforms, and delivers real-time data streams into various AWS data stores and analytics services. It handles the data ingestion on your behalf.

When it comes to VPC flow logs, Firehose can be useful. VPC flow logs capture information about the IP traffic going to and from network interfaces in your VPC. This data can be crucial for security monitoring, performance analysis, and regulatory compliance. However, managing the storage and processing of this continuous flow of log data can be a complex and resource-intensive task.

By integrating Firehose with your VPC flow logs, you can deliver this data to your preferred destination, such as Amazon S3, Amazon Redshift, or Amazon OpenSearch Service. Firehose will scale to handle the ingestion, transformation, and delivery of your VPC flow logs, relieving you of the operational burden. This allows you to focus on analyzing the logs and deriving insights, rather than worrying about the underlying infrastructure.

Additionally, Firehose offers features like data transformation, compression, and encryption, which can enhance the efficiency and security of your VPC flow log processing pipeline. Using Firehose for VPC flow logs can simplify your data management and enable you to gain insights from your network traffic data.

When publishing to Amazon Data Firehose, flow log data is published to a Amazon Data Firehose delivery stream, in plain text format.

Pricing

Standard ingestion and delivery charges apply. For more information, open Amazon CloudWatch Pricing, select **Logs** and find **Vended Logs**.

Contents

- IAM roles for cross account delivery
- Create a flow log that publishes to Amazon Data Firehose

IAM roles for cross account delivery

When you publish to Amazon Data Firehose, you can choose a delivery stream that's in the same account as the resource to monitor (the source account), or in a different account (the destination account). To enable cross account delivery of flow logs to Amazon Data Firehose, you must create an IAM role in the source account and an IAM role in the destination account.

Roles

- Source account role
- Destination account role

Source account role

In the source account, create a role that grants the following permissions. In this example, the name of the role is `mySourceRole`, but you can choose a different name for this role. The last statement allows the role in the destination account to assume this role. The condition statements ensure that this role is passed only to the log delivery service, and only when monitoring the specified resource. When you create your policy, specify the VPCs, network interfaces, or subnets that you're monitoring with the condition key `iam:AssociatedResourceARN`.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::123456789012:role/mySourceRole",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "delivery.logs.amazonaws.com"
                },
                "StringLike": {
                    "iam:AssociatedResourceARN": [
                        "arn:aws:ec2:us-
east-1:123456789012:vpc/vpc-00112233344556677"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:GetLogDelivery"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::111122223333:role/
AWSLogDeliveryFirehoseCrossAccountRole"
        }
    ]
}
```

Ensure that this role has the following trust policy, which allows the log delivery service to assume the role.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

From the source account, use the following procedure to create the role.

To create the source account role

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. On the **Create policy** page, do the following:

   a. Choose **JSON**.

   b. Replace the contents of this window with the permissions policy at the start of this section.

   c. Choose **Next**.

   d. Enter a name for your policy and an optional description and tags, and then choose **Create policy**.
5. In the navigation pane, choose **Roles**.
6. Choose **Create role**.
7. For **Trusted entity type**, choose **Custom trust policy**. For **Custom trust policy**, replace `"Principal": {},` with the following, which specifies the log delivery service. Choose **Next**.

```
"Principal": {
   "Service": "delivery.logs.amazonaws.com"
},
```

8. On the **Add permissions** page, select the checkbox for the policy that you created earlier in this procedure, and then choose **Next**.
9. Enter a name for your role and optionally provide a description.
10. Choose **Create role**.

Destination account role

In the destination account, create a role with a name that starts with **AWSLogDeliveryFirehoseCrossAccountRole**. This role must grant the following permissions.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "iam:CreateServiceLinkedRole",
          "firehose:TagDeliveryStream"
      ],
      "Resource": "*"
    }
  ]
}
```

Ensure that this role has the following trust policy, which allows the role that you created in the source account to assume this role.

JSON

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/mySourceRole"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

From the destination account, use the following procedure to create the role.

To create the destination account role

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. On the **Create policy** page, do the following:
   a. Choose **JSON**.
   b. Replace the contents of this window with the permissions policy at the start of this section.
   c. Choose **Next**.
   d. Enter a name for your policy that starts with **AWSLogDeliveryFirehoseCrossAccountRole**, and then choose **Create policy**.
5. In the navigation pane, choose **Roles**.
6. Choose **Create role**.
7. For **Trusted entity type**, choose **Custom trust policy**. For **Custom trust policy**, replace `"Principal": {},` with the following, which specifies the source account role. Choose **Next**.

```
"Principal": {
   "AWS": "arn:aws:iam::source-account:role/mySourceRole"
},
```

8. On the **Add permissions** page, select the checkbox for the policy that you created earlier in this procedure, and then choose **Next**.

9. Enter a name for your role and optionally provide a description.
10. Choose **Create role**.

Create a flow log that publishes to Amazon Data Firehose

You can create flow logs for your VPCs, subnets, or network interfaces.

Prerequisites

- Create the destination Amazon Data Firehose delivery stream. Use **Direct Put** as the source. For more information, see Creating an Amazon Data Firehose delivery stream.
- If you're publishing flow logs to a different account, create the required IAM roles, as described in the section called "IAM roles for cross account delivery".

To create a flow log that publishes to Amazon Data Firehose

1. Do one of the following:
   - Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. In the navigation pane, choose **Network Interfaces**. Select the checkbox for the network interface.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Your VPCs**. Select the checkbox for the VPC.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose **Subnets**. Select the checkbox for the subnet.
2. Choose **Actions**, **Create flow log**.
3. For **Filter**, specify the type of traffic to log.
   - **Accept** – Log only accepted traffic
   - **Reject** – Log only rejected traffic
   - **All** – Log accepted and rejected traffic
4. For **Maximum aggregation interval**, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.
5. For **Destination**, choose either of the following options:
   - **Send to Amazon Data Firehose in the same account** – The delivery stream and the resource to monitor are in the same account.

    - **Send to Amazon Data Firehose in a different account –** The delivery stream and the resource to monitor are in different accounts.
6. For **Amazon Data Firehose** stream name, choose the delivery stream that you created.
7. [Cross account delivery only] For **Service access**, choose an existing [IAM service role for cross account delivery]() that has permissions to publish logs or choose **Set up permissions** to open the IAM console and create a service role.
8. For **Log record format**, specify the format for the flow log record.
    - To use the default flow log record format, choose **AWS default format**.
    - To create a custom format, choose **Custom format**. For **Log format**, choose the fields to include in the flow log record.
9. For **Additional metadata**, select if you want to include metadata from Amazon ECS in the log format.
10. (Optional) Choose **Add tag** to apply tags to the flow log.
11. Choose **Create flow log**.

To create a flow log that publishes to Amazon Data Firehose using the command line

Use one of the following commands:

- create-flow-logs (AWS CLI)
- New-EC2FlowLog (AWS Tools for Windows PowerShell)

The following AWS CLI example creates a flow log that captures all traffic for the specified VPC and delivers the flow logs to the specified Amazon Data Firehose delivery stream in the same account.

```
aws ec2 create-flow-logs --traffic-type ALL \
  --resource-type VPC \
  --resource-ids vpc-00112233344556677 \
  --log-destination-type kinesis-data-firehose \
  --log-destination arn:aws:firehose:us-
east-1:123456789012:deliverystream/flowlogs_stream
```

The following AWS CLI example creates a flow log that captures all traffic for the specified VPC and delivers the flow logs to the specified Amazon Data Firehose delivery stream in a different account.

```
aws ec2 create-flow-logs --traffic-type ALL \
```

```
  --resource-type VPC \
  --resource-ids vpc-00112233344556677 \
  --log-destination-type kinesis-data-firehose \
  --log-destination arn:aws:firehose:us-
east-1:123456789012:deliverystream/flowlogs_stream \
  --deliver-logs-permission-arn arn:aws:iam::source-account:role/mySourceRole \
  --deliver-cross-account-role arn:aws:iam::destination-account:role/
AWSLogDeliveryFirehoseCrossAccountRole
```

As a result of creating the flow log, you can get the flow log data from the destination that you configured for the delivery stream.

Query flow logs using Amazon Athena

Amazon Athena is an interactive query service that enables you to analyze data in Amazon S3, such as your flow logs, using standard SQL. You can use Athena with VPC Flow Logs to quickly get actionable insights about the traffic flowing through your VPC. For example, you can identify which resources in your virtual private clouds (VPCs) are the top talkers or identify the IP addresses with the most rejected TCP connections.

Options

- You can streamline and automate the integration of your VPC flow logs with Athena by generating a CloudFormation template that creates the required AWS resources and predefined queries that you can run to obtain insights about the traffic flowing through your VPC.
- You can create your own queries using Athena. For more information, see Query flow logs using Amazon Athena in the *Amazon Athena User Guide*.

Pricing

You incur standard Amazon Athena charges for running queries. You incur standard AWS Lambda charges for the Lambda function that loads new partitions on a recurring schedule (when you specify a partition load frequency but do not specify a start and end date.)

To use the predefined queries

- Generate the CloudFormation template using the console
- Generate the CloudFormation template using the AWS CLI
- Run a predefined query

Generate the CloudFormation template using the console

After the first flow logs are delivered to your S3 bucket, you can integrate with Athena by generating a CloudFormation template and using the template to create a stack.

Requirements

- The selected Region must support AWS Lambda and Amazon Athena.
- The Amazon S3 buckets must be in the selected Region.
- The log record format for the flow log must include the fields used by the specific predefined queries that you'd like to run.

To generate the template using the console

1. Do one of the following:
   - Open the Amazon VPC console. In the navigation pane, choose **Your VPCs** and then select your VPC.
   - Open the Amazon VPC console. In the navigation pane, choose **Subnets** and then select your subnet.
   - Open the Amazon EC2 console. In the navigation pane, choose **Network Interfaces** and then select your network interface.
2. On the **Flow logs** tab, select a flow log that publishes to Amazon S3 and then choose **Actions**, **Generate Athena integration**.
3. Specify the partition load frequency. If you choose **None**, you must specify the partition start and end date, using dates that are in the past. If you choose **Daily**, **Weekly**, or **Monthly**, the partition start and end dates are optional. If you do not specify start and end dates, the CloudFormation template creates a Lambda function that loads new partitions on a recurring schedule.
4. Select or create an S3 bucket for the generated template, and an S3 bucket for the query results.
5. Choose **Generate Athena integration**.
6. (Optional) In the success message, choose the link to navigate to the bucket that you specified for the CloudFormation template, and customize the template.
7. In the success message, choose **Create CloudFormation stack** to open the **Create Stack** wizard in the AWS CloudFormation console. The URL for the generated CloudFormation template

is specified in the **Template** section. Complete the wizard to create the resources that are specified in the template.

Resources created by the CloudFormation template

- An Athena database. The database name is vpcflowlogsathenadatabase<*flow-logs-subscription-id*>.
- An Athena workgroup. The workgroup name is <*flow-log-subscription-id*><*partition-load-frequency*><*start-date*><*end-date*>workgroup
- A partitioned Athena table that corresponds to your flow log records. The table name is <*flow-log-subscription-id*><*partition-load-frequency*><*start-date*><*end-date*>.
- A set of Athena named queries. For more information, see Predefined queries.
- A Lambda function that loads new partitions to the table on the specified schedule (daily, weekly, or monthly).
- An IAM role that grants permission to run the Lambda functions.

Generate the CloudFormation template using the AWS CLI

After the first flow logs are delivered to your S3 bucket, you can generate and use a CloudFormation template to integrate with Athena.

Use the following get-flow-logs-integration-template command to generate the CloudFormation template.

```
aws ec2 get-flow-logs-integration-template --cli-input-json file://config.json
```

The following is an example of the `config.json` file.

```
{
    "FlowLogId": "fl-12345678901234567",
    "ConfigDeliveryS3DestinationArn": "arn:aws:s3:::my-flow-logs-athena-integration/
templates/",
    "IntegrateServices": {
        "AthenaIntegrations": [
            {
                "IntegrationResultS3DestinationArn": "arn:aws:s3:::my-flow-logs-
analysis/athena-query-results/",
                "PartitionLoadFrequency": "monthly",
```

```
                "PartitionStartDate": "2021-01-01T00:00:00",
                "PartitionEndDate": "2021-12-31T00:00:00"
            }
        ]
    }
}
```

Use the following create-stack command to create a stack using the generated CloudFormation template.

```
aws cloudformation create-stack --stack-name my-vpc-flow-logs --template-body file://
my-cloudformation-template.json
```

Run a predefined query

The generated CloudFormation template provides a set of predefined queries that you can run to quickly get meaningful insights about the traffic in your AWS network. After you create the stack and verify that all resources were created correctly, you can run one of the predefined queries.

To run a predefined query using the console

1. Open the Athena console.
2. In the left nav, choose **Query editor**. Under **Workgroup**, select the workgroup created by the CloudFormation template.
3. Select **Saved queries**, select a query, modify the parameters as needed, and run the query. For a list of available predefined queries, see Predefined queries.
4. Under **Query results**, view the query results.

Predefined queries

The following is the complete list of Athena named queries. The predefined queries that are provided when you generate the template depend on the fields that are part of the log record format for the flow log. Therefore, the template might not contain all of these predefined queries.

- **VpcFlowLogsAcceptedTraffic** – The TCP connections that were allowed based on your security groups and network ACLs.
- **VpcFlowLogsAdminPortTraffic** – The top 10 IP addresses with the most traffic, as recorded by applications serving requests on administrative ports.

- **VpcFlowLogsIPv4Traffic** – The total bytes of IPv4 traffic recorded.
- **VpcFlowLogsIPv6Traffic** – The total bytes of IPv6 traffic recorded.
- **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
- Flow log is active, but no flow log records or log group
- 'LogDestinationNotFoundException' or 'Access Denied for LogDestination' error
- Exceeding the Amazon S3 bucket policy limit
- LogDestination undeliverable
- Flow logs data size mismatch with billing data

Incomplete flow log records

Problem

Your flow log records are incomplete or are no longer being published.

Cause

There might be a problem delivering the flow logs to the CloudWatch Logs log group or SkipData entries may be present.

Solution

Check the **Flow logs** tab for the VPC, subnet, or network interface. Note that you can't describe flow logs for a VPC or subnet that was shared with you, but you can describe flow logs for a network interface that you create in a VPC or subnet that was shared with you. If there are any errors, they appear in the **Status** column. Alternatively, use the describe-flow-logs command, and check the value that's returned in the `DeliverLogsErrorMessage` field.

The following are possible error values for the status:

- `Rate limited`: This error can occur if CloudWatch Logs throttling has been applied — when the number of flow log records for a network interface is higher than the maximum number of records that can be published within a specific timeframe. This error can also occur if you've reached the quota for the number of CloudWatch Logs log groups that you can create. For more information, see CloudWatch service quotas in the *Amazon CloudWatch User Guide*.
- `Access error`: This error can occur for one of the following reasons:
  - The IAM role for your flow log does not have sufficient permissions to publish flow log records to the CloudWatch log group
  - The IAM role does not have a trust relationship with the flow logs service
  - The trust relationship does not specify the flow logs service as the principal

  For more information, see IAM role for publishing flow logs to CloudWatch Logs.
- `Unknown error`: An internal error has occurred in the flow logs service.

Flow log is active, but no flow log records or log group

Problem

You created a flow log, and the Amazon VPC or Amazon EC2 console displays the flow log as `Active`. However, you cannot see any log streams in CloudWatch Logs or log files in your Amazon S3 bucket.

Possible causes

- The flow log is still being created. In some cases, it can take ten minutes or more after you create the flow log for the log group to be created, and for data to be displayed.
- There has been no traffic recorded for your network interfaces yet. The log group in CloudWatch Logs is only created when traffic is recorded.

Solution

Wait a few minutes for the log group to be created, or for traffic to be recorded.

'LogDestinationNotFoundException' or 'Access Denied for LogDestination' error

Problem

You get a `Access Denied for LogDestination` or a `LogDestinationNotFoundException` error when you create a flow log.

Possible causes

- When creating a flow log that publishes data to an Amazon S3 bucket, this error indicates that the specified S3 bucket could not be found or that the bucket policy does not allow logs to be delivered to the bucket.
- When creating a flow log that publishes data to Amazon CloudWatch Logs, this error indicates that the IAM role does not allow logs to be delivered to the log group.

Solution

- When publishing to Amazon S3, ensure that you have specified the ARN for an existing S3 bucket, and that the ARN is in the correct format. If you do not own the S3 bucket, verify that the bucket policy has the required permissions and uses the correct account ID and bucket name in the ARN.
- When publishing to CloudWatch Logs, verify that the IAM role has the required permissions.

Exceeding the Amazon S3 bucket policy limit

Problem

You get the following error when you try to create a flow log: `LogDestinationPermissionIssueException`.

Possible causes

Amazon S3 bucket policies are limited to 20 KB in size.

Each time that you create a flow log that publishes to an Amazon S3 bucket, we automatically add the specified bucket ARN, which includes the folder path, to the `Resource` element in the bucket's policy.

Creating multiple flow logs that publish to the same bucket could cause you to exceed the bucket policy limit.

Solution

- Clean up the bucket policy by removing the flow log entries that are no longer needed.
- Grant permissions to the entire bucket by replacing the individual flow log entries with the following.

  ```
  arn:aws:s3:::bucket_name/*
  ```

  If you grant permissions to the entire bucket, new flow log subscriptions do not add new permissions to the bucket policy.

LogDestination undeliverable

Problem

You get the following error when you try to create a flow log: `LogDestination <bucket name> is undeliverable`.

Possible causes

The target Amazon S3 bucket is encrypted using server-side encryption with AWS KMS (SSE-KMS) and the default encryption of the bucket is a KMS key ID.

Solution

The value must be a KMS key ARN. Change the default S3 encryption type from KMS key ID to KMS key ARN. For more information, see Configuring default encryption in the *Amazon Simple Storage Service User Guide*.

Flow logs data size mismatch with billing data

Problem

The total data size of your flow logs does not match the size reported by billing data.

Possible causes

There may be SKIPDATA entries in your flow logs. See No data and skipped records for an explanation of SKIPDATA entries.

Solution

Confirm that SKIPDATA entries are present in your log entries by querying your logs for different entries in the log-status field.

Sample queries to check for SKIPDATA:

CW Insights:

```
fields @timestamp, @message, @logStream, @log
| filter interfaceId = 'eni-123'
| stats count(*) by interfaceId, logStatus
| sort by interfaceId, logStatus
```

Athena:

```
SELECT log_status, interface_id, count(1)
FROM vpc_flow_logs
WHERE interface_id IN ('eni-1', 'eni-2', 'eni-3')
GROUP BY log_status, interface_id
```

CloudWatch metrics for your VPCs

Amazon VPC publishes data about your VPCs to Amazon CloudWatch. You can retrieve statistics about your VPCs as an ordered set of time-series data, known as *metrics*. Think of a metric as a variable to monitor and the data as the value of that variable over time. For more information, see the Amazon CloudWatch User Guide.

Contents

- NAU metrics and dimensions
- Enable or disable NAU monitoring
- NAU CloudWatch alarm example

NAU metrics and dimensions

Network Address Usage (NAU) is a metric applied to resources in your virtual network to help you plan for and monitor the size of your VPC. There is no cost to monitor NAU. Monitoring NAU is helpful because if you exhaust the NAU or peered NAU quotas for your VPC, you can't launch new EC2 instances or provision new resources, such as Network Load Balancers, VPC endpoints, Lambda functions, transit gateway attachments, and NAT gateways.

If you've enabled Network Address Usage monitoring for a VPC, Amazon VPC sends metrics related to NAU to Amazon CloudWatch. The size of a VPC is measured by the number of Network Address Usage (NAU) units that the VPC contains.

You can use these metrics to understand the rate of your VPC growth, forecast when your VPC will reach its size limit, or create alarms when size thresholds are crossed.

The `AWS/EC2`namespace includes the following metrics for monitoring NAU.

Metric	Description
`NetworkAddressUsage`	The NAU count per VPC. **Reporting criteria** • Every 24 hours. **Dimensions** • Name: `Per-VPC Metrics`, Value: The VPC ID.

Metric	Description
`NetworkAddressUsagePeered`	The NAU count for the VPC and all VPCs that it's peered with. **Reporting criteria** • Every 24 hours. **Dimensions** • Name: `Per-VPC Metrics`, Value: The VPC ID.

The `AWS/Usage`namespace includes the following metrics for monitoring NAU.

Metric	Description
`ResourceCount`	The NAU count per VPC. **Reporting criteria** • Every 24 hours. **Dimensions** • Name: `Service`, Value: `EC2` • Name: `Type`, Value: `Resource` • Name: `Resource`, Value: The VPC ID. • Name: `Class`, Value: `NetworkAddressUsage`
`ResourceCount`	The NAU count for the VPC and all VPCs that it's peered with. **Reporting criteria** • Every 24 hours.

Metric	Description
	Dimensions • Name: `Service`, Value: `EC2` • Name: `Type`, Value: `Resource` • Name: `Resource`, Value: The VPC ID. • Name: `Class`, Value: `NetworkAddressUsagePeered`
`ResourceCount`	A combined view of NAU usage across VPCs. **Reporting criteria** • Every 24 hours. **Dimensions** • Name: `Service`, Value: `EC2` • Name: `Type`, Value: `Resource` • Name: `Resource`, Value: `VPC` • Name: `Class`, Value: `NetworkAddressUsage`

Metric	Description
`ResourceCount`	A combined view of NAU usage across peered VPCs. **Reporting criteria** • Every 24 hours. **Dimensions** • Name: `Service`, Value: `EC2` • Name: `Type`, Value: `Resource` • Name: `Resource`, Value: `VPC` • Name: `Class`, Value: `NetworkAddressUsagePeered`

Enable or disable NAU monitoring

To view NAU metrics in CloudWatch, you must first enable monitoring on each VPC to monitor.

To enable or disable monitoring NAU

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Your VPCs**.
3. Select the check box for the VPC.
4. Select **Actions**, **Edit VPC settings**.
5. Do one of the following:
   - To enable monitoring, select **Network mapping units metrics settings**, **Enable network address usage metrics**.
   - To disable monitoring, clear **Network mapping units metrics settings**, **Enable network address usage metrics**.

To enable or disable monitoring using the command line

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

NAU CloudWatch alarm example

You can use the following AWS CLI command and example `.json` to create an Amazon CloudWatch alarm and SNS notification that tracks NAU utilization of the VPC with 50,000 NAUs as the threshold. This sample requires you to first create an Amazon SNS topic. For more information, see Getting started with Amazon SNS in the *Amazon Simple Notification Service Developer Guide*.

```
aws cloudwatch put-metric-alarm --cli-input-json file://nau-alarm.json
```

The following is an example of `nau-alarm.json`.

```
{
    "Namespace": "AWS/EC2",
    "MetricName": "NetworkAddressUsage",
    "Dimensions": [{
        "Name": "Per-VPC Metrics",
        "Value": "vpc-0123456798"
    }],
    "AlarmActions": ["arn:aws:sns:us-west-1:123456789012:my_sns_topic"],
    "ComparisonOperator": "GreaterThanThreshold",
    "Period": 86400,
    "EvaluationPeriods": 1,
    "Threshold": 50000,
    "AlarmDescription": "Tracks NAU utilization of the VPC with 50k NAUs as the
 threshold",
    "AlarmName": "VPC NAU Utilization",
    "Statistic": "Maximum"
}
```

Understand codes for Amazon VPC in billing and usage reports

When you use Amazon VPC, we include related codes in your AWS billing and usage reports. Reviewing these codes helps you understand your costs and usage patterns for Amazon VPC. Tracking and managing your expenses is essential for optimizing your costs.

The following tables describe the codes for Amazon VPC that appear in your billing and usage reports. For a list of the Region codes used in the billing and usage reports, see AWS Region billing codes.

Billing codes for:

- IP address management
- VPC endpoints
- Transit gateways
- Network analysis
- Traffic mirroring
- VPC Lattice
- Cross-account/Region resources

Related resources

- Amazon VPC pricing
- AWS PrivateLink pricing
- AWS Transit Gateway pricing
- Amazon VPC Lattice pricing

IP address management

Code	Description	Units	Granularity
`*region*-PublicIPv4:InUseAddress`	The time that public IPv4 addresses are in use by a resource.	Hours	Per-second
`*region*-PublicIPv4:IdleAddress`	The time that public IPv4 addresses are not in use by a resource.	Hours	Per-second

Code	Description	Units	Granularity
region-PublicIPv4:ContiguousBlock	The use of public IPv4 addresses in an Amazon-provided contiguous IPv4 block.	Hours	Hourly
region-IPAddressManager-IP-Hours	The time that IP addresses are managed by IPAM Advanced Tier.	Hours	Hourly

VPC endpoints

Code	Description	Units	Granularity
region-VpcEndpoint-Hours	The time that interface VPC endpoints are provisioned.	Hours	Hourly
region-VpcEndpoint-Bytes	The data processed by interface VPC endpoints.	GB	Hourly
region-VpcEndpoint-GWLBE-Hours	The time that Gateway Load Balancer endpoints are provisioned.	Hours	Hourly
region-VpcEndpoint-GWLBE-Bytes	The data processed by Gateway Load Balancer endpoints.	GB	Hourly

Transit gateways

Code	Description	Units	Granularity
region-TransitGateway-Hours	The use of transit gateway attachments.	Hours	Hourly
region-TransitGateway-Bytes	The data processed by transit gateways.	GB	Hourly
region-TGW-Multicast-Consumer-Bytes	The data processed by multicast receiver instances.	GB	Hourly

Network analysis

Code	Description	Units	Granularity
region-Analysis-Runs	The number of network paths analyzed by Reachability Analyzer.	Count	Per analysis
region-NetworkInterface-Assessment	The number of network interfaces analyzed by Network Access Analyzer.	Count	Per assessment

Traffic mirroring

Code	Description	Units	Granularity
region-ENI-Mirror	The time that a network interface is configured for traffic mirroring.	Hours	Hourly

VPC Lattice

Code	Description	Units	Granularity
region-VPCLattice-Service-Hourly	The running time for VPC Lattice services.	Hours	Hourly
region-VPCLattice-DataProcessing-Bytes	The data processed by VPC Lattice services.	GB	Hourly
region-VPCLattice-RequestCount-Free	The free HTTP requests and TCP connections.	Count	Hourly
region-VpcLattice-Service-Network-Resource-Hours	The running time for VPC Lattice service networks.	Hours	Hourly

Cross-account/Region resources

Code	Description	Units	Granularity
`*region*-VpcResource-Provider-Bytes`	The data transferred from provider resources across accounts or Regions.	GB	Hourly
`*region*-VpcResource-Consumer-Bytes`	The data transferred by consumer resources across accounts or Regions.	GB	Hourly

Managing security responsibilities for 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

- Ensure data protection in Amazon Virtual Private Cloud
- Identity and access management for Amazon VPC
- Infrastructure security in Amazon VPC
- Control traffic to your AWS resources using security groups
- Control subnet traffic with network access control lists
- Resilience in Amazon Virtual Private Cloud
- Compliance validation for Amazon Virtual Private Cloud
- Block public access to VPCs and subnets

- Security best practices for your VPC

Ensure 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. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

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

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 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) 140-3.

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

Ensure internetwork traffic privacy in Amazon VPC

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

- **Security groups**: Security groups allow specific inbound and outbound traffic at the resource level (such as an EC2 instance). When you launch an instance, you can associate it with one or more security groups. Each instance in your VPC could belong to a different set of security groups. If you don't specify a security group when you launch an instance, the instance is automatically associated with the default security group for its VPC. For more information, see Security groups.
- **Network access control lists (ACL)**: Network ACLs allow or deny specific inbound and outbound traffic at the subnet level. For more information, see Control subnet traffic with network access control lists.
- **Flow logs**: Flow logs capture information about the IP traffic going to and from network interfaces in your VPC. You can create a flow log for a VPC, subnet, or individual network interface. Flow log data is published to CloudWatch Logs or Amazon S3, and it can help you diagnose overly restrictive or overly permissive security group and network ACL rules. For more information, see Logging IP traffic using VPC Flow Logs.
- **Traffic mirroring**: You can copy network traffic from an elastic network interface of an Amazon EC2 instance. You can then send the traffic to out-of-band security and monitoring appliances. For more information, see the Traffic Mirroring Guide.

Identity and access management for Amazon VPC

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

Contents

- Audience
- Authenticate with identities
- Manage access using policies
- How Amazon VPC works with IAM

- Amazon VPC policy examples
- Troubleshoot Amazon VPC identity and access
- AWS managed policies for Amazon Virtual Private Cloud

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.

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.

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.

Authenticate with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see Use cases for IAM users 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. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). 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 Methods to assume a role in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM 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.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the

principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

  - **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 Create 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 AWS 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 Use an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Manage access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

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

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed 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 Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

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

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

Access control lists (ACLs)

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

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

Other policy types

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

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) 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

- Actions
- Resources
- Condition keys
- Amazon VPC resource-based policies
- Authorization based on tags
- IAM roles

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

Actions

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

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

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

Amazon VPC shares its API namespace with Amazon EC2. Policy actions in Amazon VPC use the following prefix before the action: `ec2:`. For example, to grant a user permission to create a VPC using the `CreateVpc` API operation, you grant access to the `ec2:CreateVpc` action. Policy statements must include either an `Action` or `NotAction` element.

To specify multiple actions in a single statement, separate them with commas as shown in the following example.

```
"Action": [
      "ec2:action1",
      "ec2:action2"
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action.

```
"Action": "ec2:Describe*"
```

To see a list of Amazon VPC actions, see Actions defined by Amazon EC2 in the *Service Authorization Reference*.

Resources

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

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

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

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

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

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

For example, to specify the `vpc-1234567890abcdef0` VPC in your statement, use the ARN shown in the following example.

```
"Resource": "arn:aws:ec2:us-east-1:123456789012:vpc/vpc-1234567890abcdef0"
```

To specify all VPCs in a specific Region that belong to a specific account, use the wildcard (*).

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

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

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

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

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

To see a list of Amazon VPC resource types and their ARNs, see Resource types defined by Amazon EC2 in the *Service Authorization Reference*.

Condition keys

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

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

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

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

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

All Amazon EC2 actions support the `aws:RequestedRegion` and `ec2:Region` condition keys. For more information, see Example: Restrict access to a specific Region.

Amazon VPC defines its own set of condition keys and also supports using some global condition keys. To see a list of Amazon VPC condition keys, see Condition keys for Amazon EC2 in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon EC2.

Amazon VPC resource-based policies

Resource-based policies are JSON policy documents that specify what actions a specified principal can perform on the Amazon VPC resource and under what conditions.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, you must also grant the principal entity permission to access the resource. Grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Authorization based on tags

You can attach tags to Amazon VPC resources or pass tags in a request. To control access based on tags, you provide tag information in the condition element of a policy using condition keys. For more information, see Grant permission to tag resources during creation 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.

IAM roles

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

Use temporary credentials

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

Amazon VPC supports using temporary credentials.

Service-linked roles

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

Transit gateways support service-linked roles.

Service roles

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

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

Amazon VPC policy examples

By default, IAM roles don't have permission to create or modify VPC resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM roles that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

Contents

- Policy best practices
- Use the Amazon VPC console
- Create a VPC with a public subnet
- Modify and delete VPC resources
- Manage security groups
- Manage security group rules
- Launch instances into a specific subnet
- Launch instances into a specific VPC
- Block public access to VPCs and subnets
- Additional Amazon VPC policy examples

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon VPC resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Use the Amazon VPC console

To access the Amazon VPC console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon VPC resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM roles) with that policy.

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

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeClassicLinkInstances",
                "ec2:DescribeClientVpnEndpoints",
                "ec2:DescribeCustomerGateways",
                "ec2:DescribeDhcpOptions",
```

```
                "ec2:DescribeEgressOnlyInternetGateways",
                "ec2:DescribeFlowLogs",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeManagedPrefixLists",
                "ec2:DescribeMovingAddresses",
                "ec2:DescribeNatGateways",
                "ec2:DescribeNetworkAcls",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribePrefixLists",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupReferences",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSecurityGroupRules",
                "ec2:DescribeStaleSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeTags",
                "ec2:DescribeTrafficMirrorFilters",
                "ec2:DescribeTrafficMirrorSessions",
                "ec2:DescribeTrafficMirrorTargets",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeTransitGatewayVpcAttachments",
                "ec2:DescribeTransitGatewayRouteTables",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeVpcClassicLinkDnsSupport",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcEndpointConnectionNotifications",
                "ec2:DescribeVpcEndpointConnections",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcEndpointServicePermissions",
                "ec2:DescribeVpcEndpointServices",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpnConnections",
                "ec2:DescribeVpnGateways",
                "ec2:GetManagedPrefixListAssociations",
                "ec2:GetManagedPrefixListEntries"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

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

Create a VPC with a public subnet

The following example enables roles to create VPCs, subnets, route tables, and internet gateways. Roles can also attach an internet gateway to a VPC and create routes in route tables. The `ec2:ModifyVpcAttribute` action enables roles to enable DNS hostnames for the VPC, so that each instance launched into a VPC receives a DNS hostname.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVpc",
        "ec2:CreateSubnet",
        "ec2:DescribeAvailabilityZones",
        "ec2:CreateRouteTable",
        "ec2:CreateRoute",
        "ec2:CreateInternetGateway",
        "ec2:AttachInternetGateway",
        "ec2:AssociateRouteTable",
        "ec2:ModifyVpcAttribute"
      ],
      "Resource": "*"
    }
   ]
}
```

The preceding policy also enables roles to create a VPC in the Amazon VPC console.

Modify and delete VPC resources

You might want to control the VPC resources that roles can modify or delete. For example, the following policy allows roles to work with and delete route tables that have the tag `Purpose=Test`. The policy also specifies that roles can only delete internet gateways that have the tag `Purpose=Test`. Roles cannot work with route tables or internet gateways that do not have this tag.

JSON

```
{
    "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 roles to manage security groups. The first statement allows roles to delete any security group with the tag `Stack=test` and to manage the inbound and outbound rules for any security group with the tag `Stack=test`. The second statement requires roles to tag any security groups that they create with the tag `Stack=Test`. The third statement allows roles to create tags when creating a security group. The fourth statement allows roles to view any security group and security group rule. The fifth statement allows roles to create a security group in a VPC.

> **Note**
>
> This policy cannot be used by the AWS CloudFormation service to create a security group with required tags. If you remove the condition on the `ec2:CreateSecurityGroup` action that requires the tag, the policy will work.

JSON

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

```
            "Effect": "Allow",
            "Action": "ec2:CreateSecurityGroup",
            "Resource": "arn:aws:ec2:*:*:security-group/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Stack": "test"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "Stack"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:security-group/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateSecurityGroup"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroupRules",
                "ec2:DescribeVpcs",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateSecurityGroup",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        }
    ]
}
```

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

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

Manage security group rules

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

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

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:UpdateSecurityGroupRuleDescriptionsEgress",
                "ec2:ModifySecurityGroupRules"
            ],
            "Resource": "arn:aws:ec2:us-east-1:123456789012:security-group/*",
            "Condition": {
                "ArnEquals": {
                    "ec2:Vpc": "arn:aws:ec2:us-
east-1:123456789012:vpc/vpc-1234567890abcdef0"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSecurityGroupRules",
```

```
                "ec2:DescribeTags"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifySecurityGroupRules"
            ],
            "Resource": "arn:aws:ec2:us-east-1:123456789012:security-group-rule/
*"
        }
    ]
}
```

Launch instances into a specific subnet

The following policy grants roles permission to launch instances into a specific subnet and to use a specific security group in the request. The policy does this by specifying the ARN for the subnet and the ARN for the security group. If roles attempt to launch an instance into a different subnet or using a different security group, the request will fail (unless another policy or statement grants roles permission to do so).

The policy also grants permission to use the network interface resource. When launching into a subnet, the `RunInstances` request creates a primary network interface by default, so the role needs permission to create this resource when launching the instance.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:us-east-1::image/ami-*",
                "arn:aws:ec2:us-east-1:123456789012:instance/*",
                "arn:aws:ec2:us-
east-1:123456789012:subnet/subnet-1234567890abcdef0",
                "arn:aws:ec2:us-east-1:123456789012:network-interface/*",
```

```
                "arn:aws:ec2:us-east-1:123456789012:volume/*",
                "arn:aws:ec2:us-east-1:123456789012:key-pair/*",
                "arn:aws:ec2:us-east-1:123456789012:security-
group/sg-0abcdef1234567890"
            ]
        }
    ]
}
```

Launch instances into a specific VPC

The following policy grants roles permission to launch instances into any subnet within a specific VPC. The policy does this by applying a condition key (`ec2:Vpc`) to the subnet resource.

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

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:us-east-1:123456789012:subnet/*",
            "Condition": {
                "ArnEquals": {
                    "ec2:Vpc": "arn:aws:ec2:us-
east-1:123456789012:vpc/vpc-1234567890abcdef0"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:us-east-1::image/ami-*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/department": "dev"
                }
```

```
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:us-east-1:123456789012:instance/*",
                "arn:aws:ec2:us-east-1:123456789012:volume/*",
                "arn:aws:ec2:us-east-1:123456789012:network-interface/*",
                "arn:aws:ec2:us-east-1:123456789012:key-pair/*",
                "arn:aws:ec2:us-east-1:123456789012:security-group/*"
            ]
        }
    ]
}
```

Block public access to VPCs and subnets

The following policy examples grant roles permission to work with the VPC Block Public Access (BPA) feature to block public access to resources in VPCs and subnets.

Example 1 - Allow read-only access to VPC BPA account-wide settings and VPC BPA exclusions.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VPCBPAReadOnlyAccess",
      "Action": [
        "ec2:DescribeVpcBlockPublicAccessOptions",
        "ec2:DescribeVpcBlockPublicAccessExclusions"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Example 2 - Allow full read and write access to VPC BPA account-wide settings and VPC BPA exclusions.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VPCBPAFullAccess",
      "Action": [
        "ec2:DescribeVpcBlockPublicAccessOptions",
        "ec2:DescribeVpcBlockPublicAccessExclusions",
        "ec2:ModifyVpcBlockPublicAccessOptions",
        "ec2:CreateVpcBlockPublicAccessExclusion",
        "ec2:ModifyVpcBlockPublicAccessExclusion",
        "ec2:DeleteVpcBlockPublicAccessExclusion"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Example 3 - Allow access to all EC2 APIs except modifying VPC BPA settings and creating exclusions.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2FullAccess",
            "Action": [
                "ec2:*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
```

```
        {
            "Sid": "VPCBPAPartialAccess",
            "Action": [
                "ec2:ModifyVpcBlockPublicAccessOptions",
                "ec2:CreateVpcBlockPublicAccessExclusion"
            ],
            "Effect": "Deny",
            "Resource": "*"
        }
    ]
}
```

Additional Amazon VPC policy examples

You can find additional example IAM policies related to Amazon VPC in the following documentation:

- Managed prefix lists
- Traffic mirroring
- Transit gateways
- VPC endpoints and VPC endpoint services (AWS PrivateLink)
- VPC peering

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
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Amazon VPC resources

I am not authorized to perform an action in Amazon VPC

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your sign-in credentials.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a subnet but belongs to an IAM role that does not have `ec2:DescribeSubnets` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ec2:DescribeSubnets on resource: subnet-id
```

In this case, Mateo asks his administrator to update the policy to allow him to access the subnet.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon VPC.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon VPC. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Amazon VPC resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon VPC supports these features, see How Amazon VPC works with IAM.
- 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 Cross account resource access in IAM in the *IAM User Guide*.

AWS managed policies for Amazon Virtual Private Cloud

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AmazonVPCFullAccess

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

To view the permissions for this policy, see AmazonVPCFullAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AmazonVPCReadOnlyAccess

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

To view the permissions for this policy, see AmazonVPCReadOnlyAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AmazonVPCCrossAccountNetworkInterfaceOperations

You can attach the `AmazonVPCCrossAccountNetworkInterfaceOperations` policy to your IAM identities. This policy grants permissions that allow the identity to create network interfaces and attach them to cross-account resources.

To view the permissions for this policy, see AmazonVPCCrossAccountNetworkInterfaceOperations in the *AWS Managed Policy Reference*.

Amazon VPC updates to AWS managed policies

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

Change	Description	Date
the section called "AmazonVPCFullAccess" – Update to an existing policy	Added the AssociateSecurityGroupVpc, DescribeSecurityGroupVpcAssociations, and DisassociateSecurityGroupVpc actions, which allow you to associate, disassociate, and view security group associations with VPCs.	December 9, 2024
the section called "AmazonVPCReadOnlyAccess" – Update to an existing policy	Added the DescribeSecurityGroupVpcAssociations action, which allows you to view	December 9, 2024

Change	Description	Date
	security group associations with VPCs.	
the section called "AmazonVPCFullAccess" – Update to an existing policy	Added the GetSecurityGroupsForVpc action, which allows you to get security groups that are usable in your VPC.	February 8, 2024
the section called "AmazonVPCReadOnlyAccess" – Update to an existing policy	Added the GetSecurityGroupsForVpc action, which allows you to get security groups that are usable in your VPC.	February 8, 2024
the section called "AmazonVPCCrossAccountNetworkInterfaceOperations" – Update to an existing policy	Added the AssignIpv6Addresses and UnassignIpv6Addresses actions, which allow you to manage the IPv6 addresses associated with network interfaces.	September 25, 2023
the section called "AmazonVPCReadOnlyAccess" – Update to an existing policy	Added the DescribeSecurityGroupRules action, which allows you to view security group rules.	August 2, 2021
the section called "AmazonVPCFullAccess" – Update to an existing policy	Added the DescribeSecurityGroupRules and ModifySecurityGroupRules actions, which allow you to view and modify security group rules.	August 2, 2021

Change	Description	Date
the section called "AmazonVPCFullAccess" – 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" – Update to an existing policy	Added actions for carrier gateways, IPv6 pools, local gateways, and local gateway route tables.	June 23, 2021

Infrastructure security in Amazon VPC

As a managed service, Amazon Virtual Private Cloud is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Amazon VPC through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

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

Network isolation

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

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

You can use AWS PrivateLink to enable resources in your VPC to connect to AWS services using private IP addresses, as if those services were hosted directly in your VPC. Therefore, you do not need to use an internet gateway or NAT device to access AWS services.

Control network traffic

Consider the following options for controlling network traffic to the resources in your VPC, such as EC2 instances:

- Use security groups as the primary mechanism for controlling network access to your VPCs. When necessary, use network ACLs to provide stateless, coarse-grain network control. Security groups are more versatile than network ACLs, due to their ability to perform stateful packet filtering and create rules that reference other security groups. Network ACLs can be effective as a secondary control (for example, to deny a specific subset of traffic) or as high-level subnet guard rails. Also, because network ACLs apply to an entire subnet, they can be used as defense-in-depth in case an instance is ever launched without the correct security group.
- Use private subnets for your instances if they should not be accessed directly from the internet. Use a bastion host or NAT gateway for internet access from instances in private subnets.
- Configure subnet route tables with the minimum network routes to support your connectivity requirements.
- Consider using additional security groups or network interfaces to control and audit Amazon EC2 instance management traffic separately from regular application traffic. Therefore, you can implement special IAM policies for change control, making it easier to audit changes to security group rules or automated rule-verification scripts. Multiple network interfaces also provide additional options for controlling network traffic, including the ability to create host-based routing policies or leverage different VPC subnet routing rules based on an network interfaces assigned to a subnet.
- Use AWS Virtual Private Network or AWS Direct Connect to establish private connections from your remote networks to your VPCs. For more information, see Network-to-Amazon VPC connectivity options.
- Use VPC Flow Logs to monitor the traffic that reaches your instances.

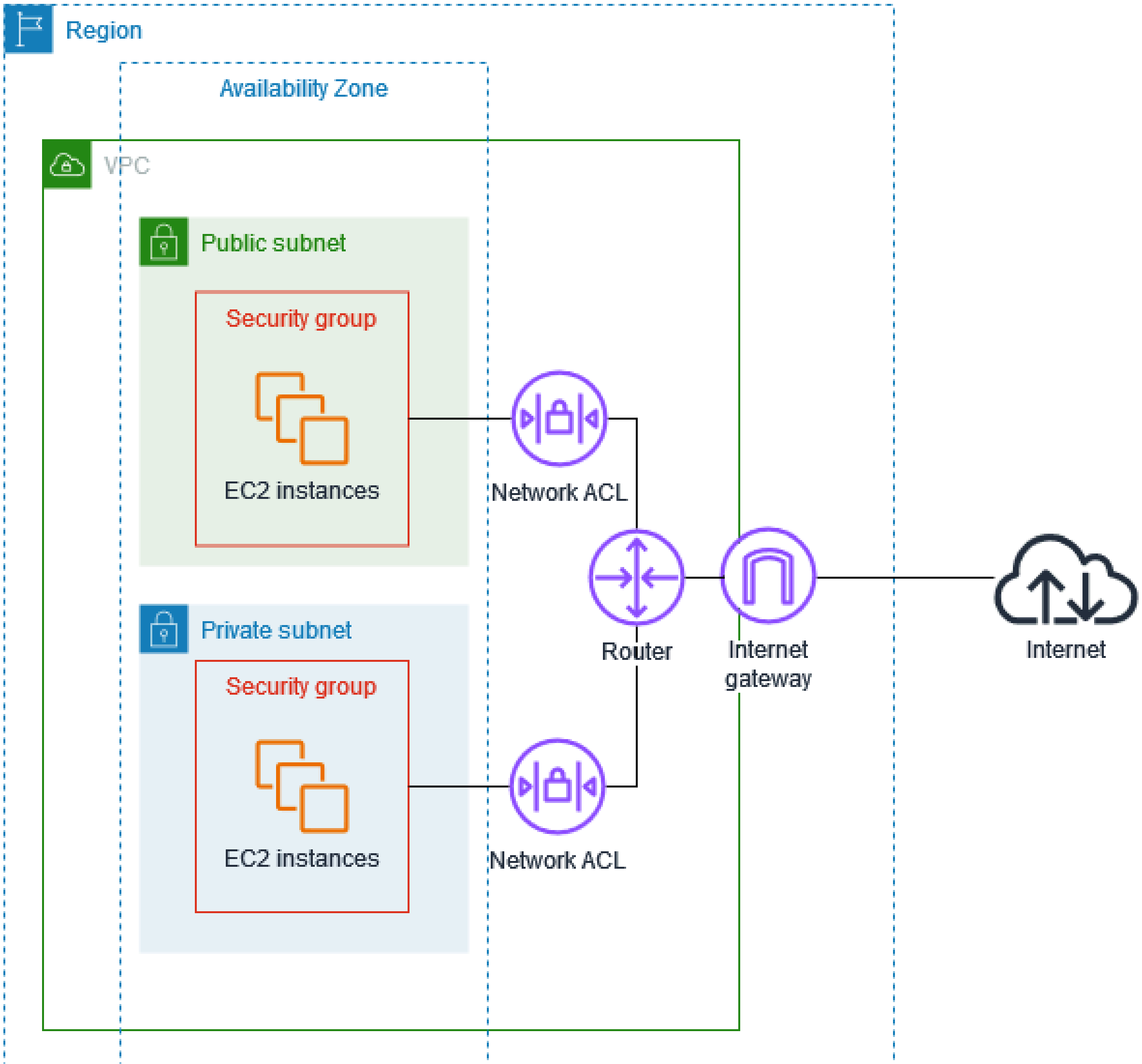
- Use AWS Security Hub to check for unintended network accessibility from your instances.
- Use AWS Network Firewall to protect the subnets in your VPC from common network threats.

Compare security groups and network ACLs

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

Characteristic	Security group	Network ACL
Level of operation	Instance level	Subnet level
Scope	Applies to all instances associated with the security group	Applies to all instances in the associated subnets
Rule type	Allow rules only	Allow and deny rules
Rule evaluation	Evaluates all rules before deciding whether to allow traffic	Evaluates rules in ascending order until a match for the traffic is found
Return traffic	Automatically allowed (stateful)	Must be explicitly allowed (stateless)

The following diagram illustrates the layers of security provided by security groups and network ACLs. For example, traffic from an internet gateway is routed to the appropriate subnet using the routes in the routing table. The rules of the network ACL that is associated with the subnet control which traffic is allowed to the subnet. The rules of the security group that is associated with an instance control which traffic is allowed to the instance.



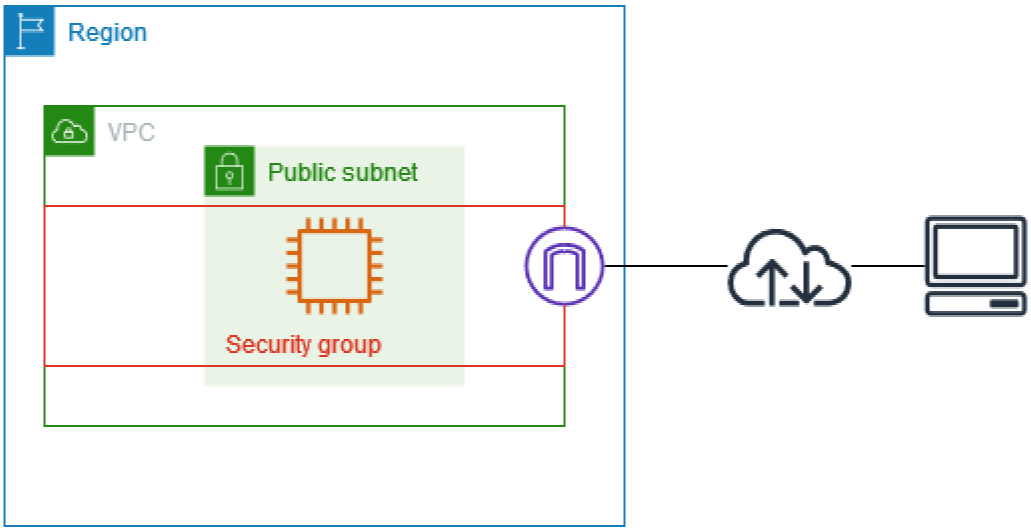
You can secure your instances using only security groups. However, you can add network ACLs as an additional layer of defense. For more information, see Example: Control access to instances in a subnet.

Control traffic to your AWS resources using security groups

A *security group* controls the traffic that is allowed to reach and leave the resources that it is associated with. For example, after you associate a security group with an EC2 instance, it controls the inbound and outbound traffic for the instance.

When you create a VPC, it comes with a default security group. You can create additional security groups for a VPC, each with their own inbound and outbound rules. You can specify the source, port range, and protocol for each inbound rule. You can specify the destination, port range, and protocol for each outbound rule.

The following diagram shows a VPC with a subnet, an internet gateway, and a security group. The subnet contains an EC2 instance. The security group is assigned to the instance. The security group acts as a virtual firewall. The only traffic that reaches the instance is the traffic allowed by the security group rules. For example, if the security group contains a rule that allows ICMP traffic to the instance from your network, then you could ping the instance from your computer. If the security group does not contain a rule that allows SSH traffic, then you could not connect to your instance using SSH.



Contents

- Security group basics
- Security group example
- Security group rules
- Default security groups for your VPCs
- Create a security group for your VPC
- Configure security group rules
- Delete a security group
- Associate security groups with multiple VPCs
- Share security groups with AWS Organizations

Pricing

There is no additional charge for using security groups.

Security group basics

- You can assign a security group to resources created in the same VPC as the security group or to resources in other VPCs if using the Security Group VPC Association feature to associate the security group to other VPCs in the same Region. You can also assign multiple security groups to a single resource.
- When you create a security group, you must provide it with a name and a description. The following rules apply:
  - A security group name must be unique within the VPC.
  - Security group names are not case-sensitive.
  - 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 can't start with `sg-`.
- Security groups are stateful. For example, if you send a request from an instance, the response traffic for that request is allowed to reach the instance regardless of the inbound security group rules. Responses to allowed inbound traffic are allowed to leave the instance, regardless of the outbound rules.
- Security groups do not filter traffic destined to and from the following:
  - Amazon Domain Name Services (DNS)
  - Amazon Dynamic Host Configuration Protocol (DHCP)
  - Amazon EC2 instance metadata
  - Amazon ECS task metadata endpoints
  - License activation for Windows instances
  - Amazon Time Sync Service
  - Reserved IP addresses used by the default VPC router
- There are quotas on the number of security groups that you can create per VPC, the number of rules that you can add to each security group, and the number of security groups that you can associate with a network interface. For more information, see Amazon VPC quotas.

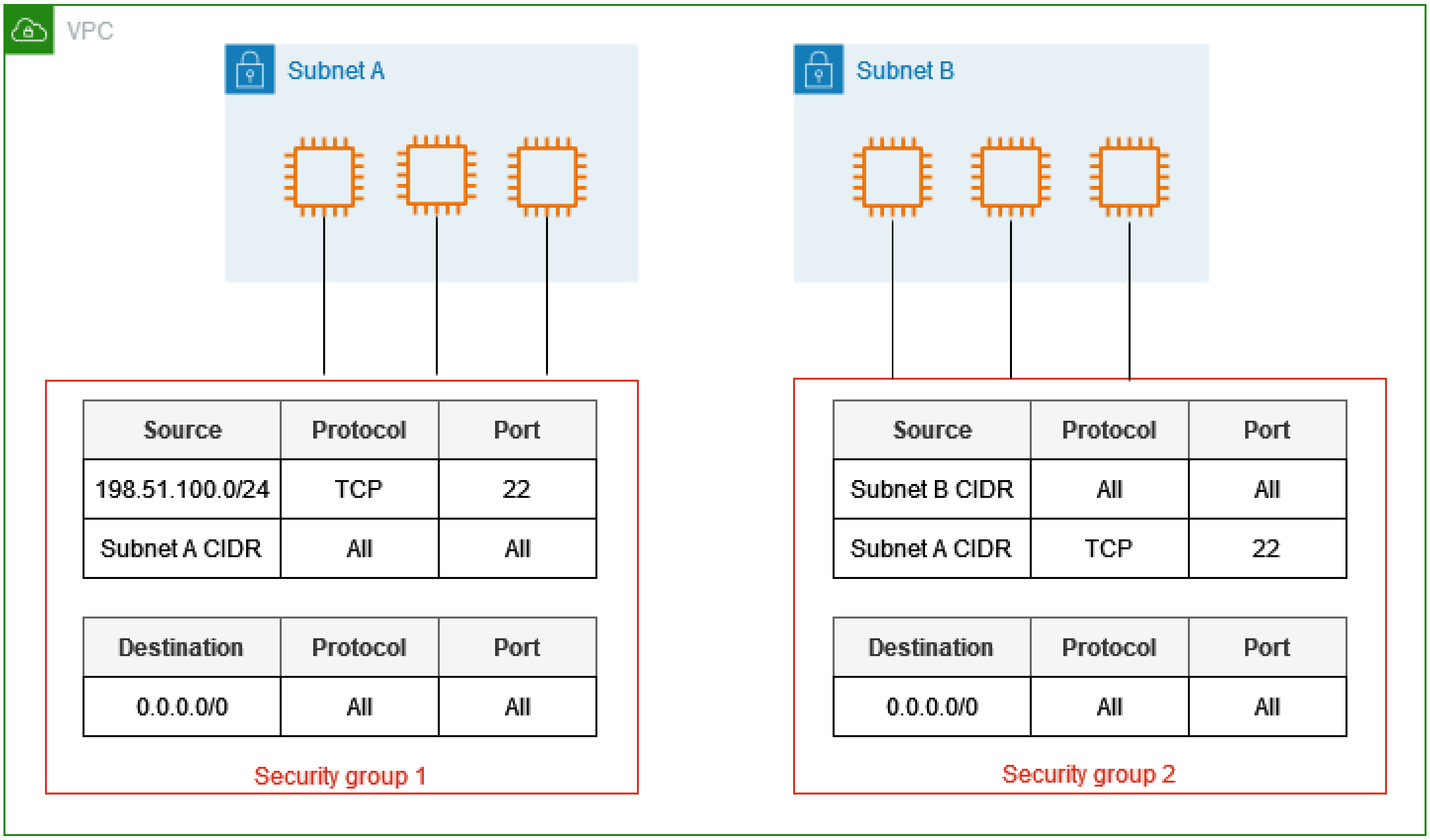
Best practices

- Authorize only specific IAM principals to create and modify security groups.
- Create the minimum number of security groups that you need, to decrease the risk of error. Use each security group to manage access to resources that have similar functions and security requirements.
- When you add inbound rules for ports 22 (SSH) or 3389 (RDP) so that you can access your EC2 instances, authorize only specific IP address ranges. If you specify 0.0.0.0/0 (IPv4) and ::/ (IPv6), this enables anyone to access your instances from any IP address using the specified protocol.
- Do not open large port ranges. Ensure that access through each port is restricted to the sources or destinations that require it.
- Consider creating network ACLs with rules similar to your security groups, to add an additional layer of security to your VPC. For more information about the differences between security groups and network ACLs, see Compare security groups and network ACLs.

Security group example

The following diagram shows a VPC with two security groups and two subnets. The instances in subnet A have the same connectivity requirements, so they are associated with security group 1. The instances in subnet B have the same connectivity requirements, so they are associated with security group 2. The security group rules allow traffic as follows:

- The first inbound rule in security group 1 allows SSH traffic to the instances in subnet A from the specified address range (for example, a range in your own network).
- The second inbound rule in security group 1 allows the instances in subnet A to communicate with each other using any protocol and port.
- The first inbound rule in security group 2 allows the instances in subnet B to communicate with each other using any protocol and port.
- The second inbound rule in security group 2 allows the instances in subnet A to communicate with the instances in subnet B using SSH.
- Both security groups use the default outbound rule, which allows all traffic.



Security group rules

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

You can add or remove rules for a security group (also referred to as *authorizing* or *revoking* inbound or outbound access). A rule applies either to inbound traffic (ingress) or outbound traffic (egress). You can grant access to a specific source or destination.

Contents

- Security group rule basics
- Components of a security group rule
- Security group referencing
- Security group size
- Stale security group rules

Security group rule basics

The following are the characteristics of security group rules:

- You can specify allow rules, but not deny rules.
- When you first create a security group, it has no inbound rules. Therefore, no inbound traffic is allowed until you add inbound rules to the security group.
- When you first create a security group, it has an outbound rule that allows all outbound traffic from the resource. You can remove the rule and add outbound rules that allow specific outbound traffic only. If your security group has no outbound rules, no outbound traffic is allowed.
- When you associate multiple security groups with a resource, the rules from each security group are aggregated to form a single set of rules that are used to determine whether to allow access.
- When you add, update, or remove rules, your changes are automatically applied to all resources associated with the security group. For instructions, see Configure security group rules.
- The effect of some rule changes can depend on how the traffic is tracked. For more information, see Connection tracking in the *Amazon EC2 User Guide*.
- When you create a security group rule, AWS assigns a unique ID to the rule. You can use the ID of a rule when you use the API or CLI to modify or delete the rule.

Limitation

Security groups cannot block DNS requests to or from the Route 53 Resolver, sometimes referred to as the 'VPC+2 IP address' (see Amazon Route 53 Resolver in the *Amazon Route 53 Developer Guide*), or as AmazonProvidedDNS. To filter DNS requests through the Route 53 Resolver, use Route 53 Resolver DNS Firewall.

Components of a security group rule

The following are the components of inbound and outbound security group rules:

- **Protocol**: The protocol to allow. The most common protocols are 6 (TCP), 17 (UDP), and 1 (ICMP).
- **Port range**: For TCP, UDP, or a custom protocol, the range of ports to allow. You can specify a single port number (for example, 22), or range of port numbers (for example, `7000-8000`).
- **ICMP type and code**: For ICMP, the ICMP type and code. For example, use type 8 for ICMP Echo Request or type 128 for ICMPv6 Echo Request. For more information, see Rules for ping/ICMP in the *Amazon EC2 User Guide*.

- **Source or destination**: The source (inbound rules) or destination (outbound rules) for the traffic to allow. Specify one of the following:
  - A single IPv4 address. You must use the `/32` prefix length. For example, `203.0.113.1/32`.
  - A single IPv6 address. You must use the `/128` prefix length. For example, `2001:db8:1234:1a00::123/128`.
  - A range of IPv4 addresses, in CIDR block notation. For example, `203.0.113.0/24`.
  - A range of IPv6 addresses, in CIDR block notation. For example, `2001:db8:1234:1a00::/64`.
  - The ID of a prefix list. For example, `pl-1234abc1234abc123`. For more information, see Managed prefix lists.
  - The ID of a security group. For example, `sg-1234567890abcdef0`. For more information, see the section called "Security group referencing".
- **(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 ._-:/()#,@[]+=;{}!$*.

For examples, see Security group rules for different use cases in the *Amazon EC2 User Guide*.

Security group referencing

When you specify a security group as the source or destination for a rule, the rule affects all instances that are associated with the security groups. The instances can communicate in the specified direction, using the private IP addresses of the instances, over the specified protocol and port.

For example, the following represents an inbound rule for a security group that references security group sg-0abcdef1234567890. This rule allows inbound SSH traffic from the instances associated with sg-0abcdef1234567890.

Source	Protocol	Port range
`sg-0abcdef1234567890`	TCP	22

When referencing a security group in a security group rule, note the following:

- You can reference a security group in the inbound rule of another security group if any of the following is true:

  - The security groups are associated with the same VPC.
  - There is a peering connection between the VPCs that the security groups are associated with.
  - There is a transit gateway between the VPCs that the security groups are associated with.
- You can reference a security group in the outbound rule if any of the following is true:
  - The security groups are associated with the same VPC.
  - There is a peering connection between the VPCs that the security groups are associated with.
- No rules from the referenced security group are added to the security group that references it.
- For inbound rules, the EC2 instances associated with a security group can receive inbound traffic from the private IP addresses of the EC2 instances associated with the referenced security group.
- For outbound rules, the EC2 instances associated with a security group can send outbound traffic to the private IP addresses of the EC2 instances associated with the referenced security group.

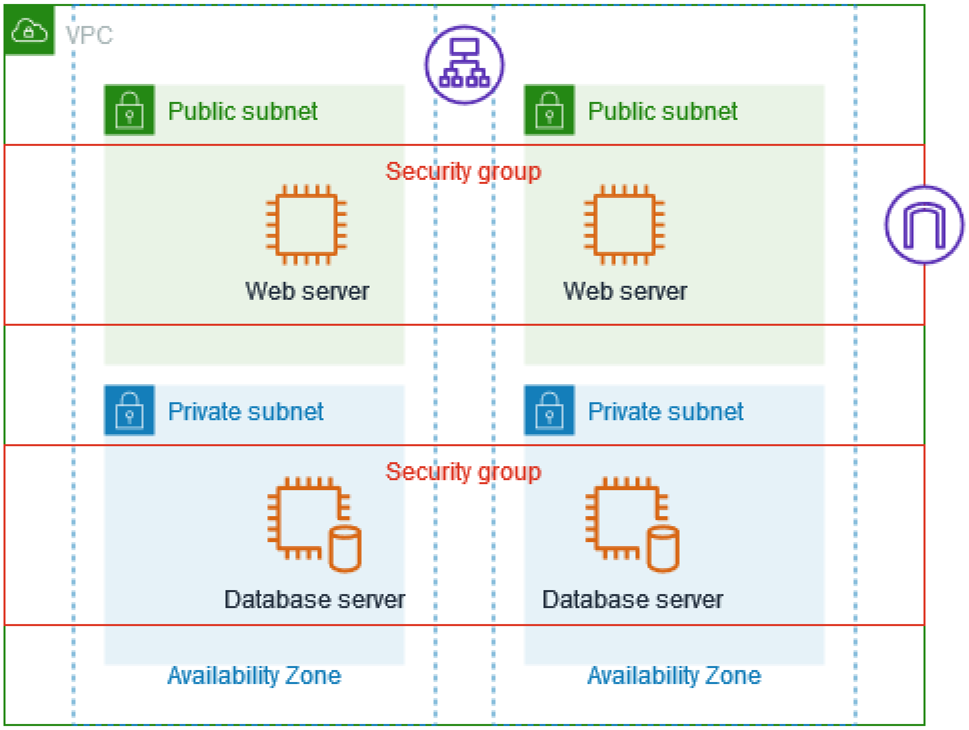
Limitation

If you configure routes to forward the traffic between two instances in different subnets through a middlebox appliance, you must ensure that the security groups for both instances allow traffic to flow between the instances. The security group for each instance must reference the private IP address of the other instance or the CIDR range of the subnet that contains the other instance as the source. If you reference the security group of the other instance as the source, this does not allow traffic to flow between the instances.

Example

The following diagram shows a VPC with subnets in two Availability Zones, an internet gateway, and an Application Load Balancer. Each Availability Zone has a public subnet for web servers and a private subnet for database servers. There are separate security groups for the load balancer, the web servers, and the database servers. Create the following security group rules to allow traffic.

- Add rules to the load balancer security group to allow HTTP and HTTPS traffic from the internet. The source is 0.0.0.0/0.
- Add rules to the security group for the web servers to allow HTTP and HTTPS traffic only from the load balancer. The source is the security group for the load balancer.
- Add rules to the security group for the database servers to allow database requests from the web servers. The source is the security group for the web servers.



Security group size

The type of source or destination determines how each rule counts toward the maximum number of rules that you can have per security group.

- A rule that references a CIDR block counts as one rule.
- A rule that references another security group counts as one rule, no matter the size of the referenced security group.
- A rule that references a customer-managed prefix list counts as the maximum size of the prefix list. For example, if the maximum size of your prefix list is 20, a rule that references this prefix list counts as 20 rules.
- A rule that references an AWS-managed prefix list counts as the weight of the prefix list. For example, if the weight of the prefix list is 10, a rule that references this prefix list counts as 10 rules. For more information, see the section called "Available AWS-managed prefix lists".

Stale security group rules

If your VPC has a VPC peering connection with another VPC, or if it uses a VPC shared by another account, a security group rule in your VPC can reference a security group in that peer VPC or shared VPC. This allows resources that are associated with the referenced security group and those that are associated with the referencing security group to communicate with each other. For more information, see Update your security groups to reference peer security groups in the *Amazon VPC Peering Guide*.

If you have a security group rule that references a security group in a peer VPC or shared VPC and the security group in the shared VPC is deleted or the VPC peering connection is deleted, the security group rule is marked as stale. You can delete stale security group rules as you would any other security group rule.

Default security groups for your VPCs

Your default VPCs and any VPCs that you create come with a default security group. The name of the default security group is "default".

We recommend that you create security groups for specific resources or groups of resources instead of using the default security group. However, if you don't associate a security group with some resources at creation time, we associate them with the default security group. For example, if you don't specify a security group when you launch an EC2 instance, we associate the instance with the default security group for its VPC.

Default security group basics

- You can change the rules for a default security group.
- You can't delete a default security group. If you try to delete a default security group, we return the following error code: `Client.CannotDelete`.

Default rules

The following table describes the default inbound rules for a default security group.

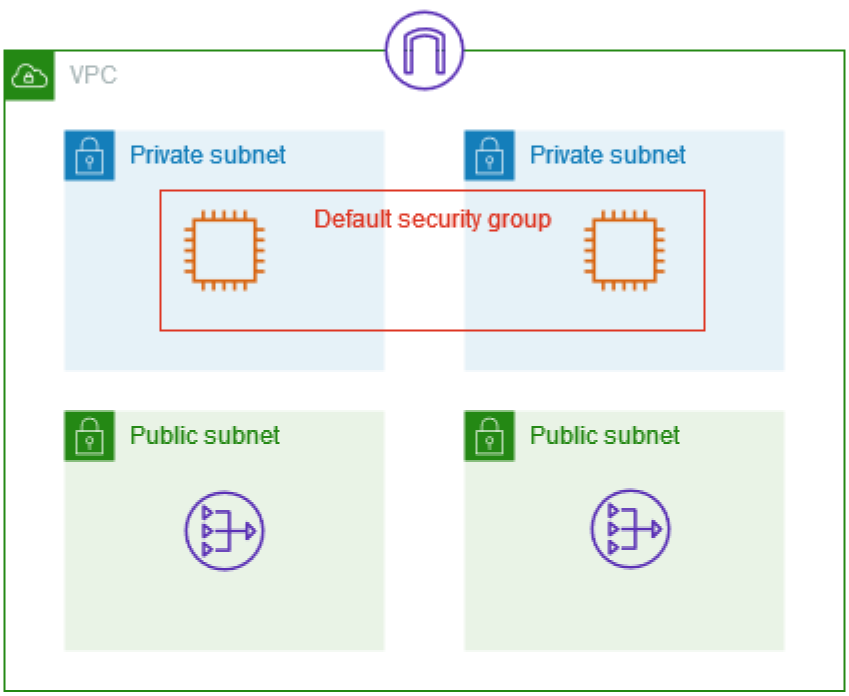
Source	Protocol	Port range	Description
sg-1234567890abcdef0	All	All	Allows inbound traffic from all resources that are assigned to this security group. The source is the ID of this security group.

The following table describes the default outbound rules for a default security group.

Destination	Protocol	Port range	Description
0.0.0.0/0	All	All	Allows all outbound IPv4 traffic.
::/0	All	All	Allows all outbound IPv6 traffic. This rule is added only if your VPC has an associated IPv6 CIDR block.

Example

The following diagram shows a VPC with a default security group, an internet gateway, and a NAT gateway. The default security contains only its default rules, and it is associated with two EC2 instances running in the VPC. In this scenario, each instance can receive inbound traffic from the other instance on all ports and protocols. The default rules do not allow the instances to receive traffic from the internet gateway or the NAT gateway. If your instances must receive additional traffic, we recommend that you create a security group with the required rules and associate the new security group with the instances instead of the default security group.



Create a security group for your VPC

Your virtual private cloud (VPC) comes with a default security group. You can create additional security groups. Security groups can be used only with resources in the VPC for which it is created.

By default, new security groups start with only an outbound rule that allows all traffic to leave the resource. You must add rules to enable any inbound traffic or to restrict the outbound traffic. You can add rules when you create a security group or later on. For more information, see Security group rules.

Required permissions

Before you begin, ensure that you have the required permissions. For more information, see the following:

- Manage security groups
- Manage security group rules

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 can't change the name and description of a security group after it is created.
5. For **VPC**, choose the VPC in which you'll create the resources to which you'll associate the security group.
6. (Optional) To add inbound rules, choose **Inbound rules**. For each rule, choose **Add rule** and specify the protocol, port, and source. For more information, see Configure security group rules.
7. (Optional) To add outbound rules, choose **Outbound rules**. For each rule, choose **Add rule** and specify the protocol, port, and destination.
8. (Optional) To add a tag, choose **Add new tag** and enter the tag key and value.
9. Choose **Create security group**.

To create a security group using the AWS CLI

Use the create-security-group command.

Alternately, you can create a new security group by copying an existing one. When you copy a security group, we automatically add the same inbound and outbound rules as the original security group and use the same VPC as the original security group. You can enter a name and description for the new security group. You can optionally choose a different VPC, and you can modify the inbound and outbound rules as needed. However, you can't copy a security group from one Region to another Region.

To create a security group based on an existing one

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security groups**.
3. Select a security group.
4. Choose **Actions**, **Copy to new security group**.
5. Enter a name and description for the security group.
6. (Optional) Choose a different VPC if needed.
7. (Optional) Add, remove, or edit the security group rules as needed.
8. Choose **Create security group**.

Configure security group rules

After you create a security group, you can add, update, and delete its security group rules. When you add, update, or delete a rule, the change is automatically applied to the resources that are associated with the security group.

Required permissions

Before you begin, ensure that you have the required permissions. For more information, see Manage security group rules.

Sources and destinations

You can specify the following as sources for inbound rules or destinations for outbound rules.

- **Custom –** A IPv4 CIDR block, and IPv6 CIDR block, another security group, or a prefix list.
- **Anywhere-IPv4 –** The 0.0.0.0/0 IPv4 CIDR block.
- **Anywhere-IPv6 –** The ::/0 IPv6 CIDR block.

- **My IP –** The public IPv4 address of your local computer.

> **Warning**
>
> If you choose **Anywhere-IPv4**, you allow traffic from all IPv4 addresses. If you choose **Anywhere-IPv6**, you allow traffic from all IPv6 addresses. It is a best practice to authorize only the specific IP address ranges that need access to your resources.

To configure security group rules 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. To edit the inbound rules, choose **Edit inbound rules** from **Actions** or the **Inbound rules** tab.

   a. To add a rule, choose **Add rule** and enter the type, protocol, port, and source for the rule.

      If the type is 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 for you.

   b. To update a rule, change its protocol, description, and source as needed. However, you can't change the source type. For example, if the source is an IPv4 CIDR block, you can't specify an IPv6 CIDR block, a prefix list, or a security group.

   c. To delete a rule, choose its **Delete** button.
5. To edit the outbound rules, choose **Edit outbound rules** from **Actions** or the **Outbound rules** tab.

   a. To add a rule, choose **Add rule** and enter the type, protocol, port, and destination for the rule. You can also enter an optional description.

      If the type is 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 for you.

   b. To update a rule, change its protocol, description, and source as needed. However, you can't change the source type. For example, if the source is an IPv4 CIDR block, you can't specify an IPv6 CIDR block, a prefix list, or a security group.

c. To delete a rule, choose its **Delete** button.

6. Choose **Save rules**.

To configure security group rules using the AWS CLI

- **Add** – Use the authorize-security-group-ingress and authorize-security-group-egress commands.
- **Remove** – Use the revoke-security-group-ingress and revoke-security-group-egress commands.
- **Modify** – Use the modify-security-group-rules, update-security-group-rule-descriptions-ingress, and update-security-group-rule-descriptions-egresscommands.

Delete a security group

When you are finished with a security group that you created, you can delete it.

Requirements

- The security group can't be associated with any resources.
- The security group can't be referenced by a rule in another security group.
- The security group can't be the default security group for a VPC.

To delete a security group using the console

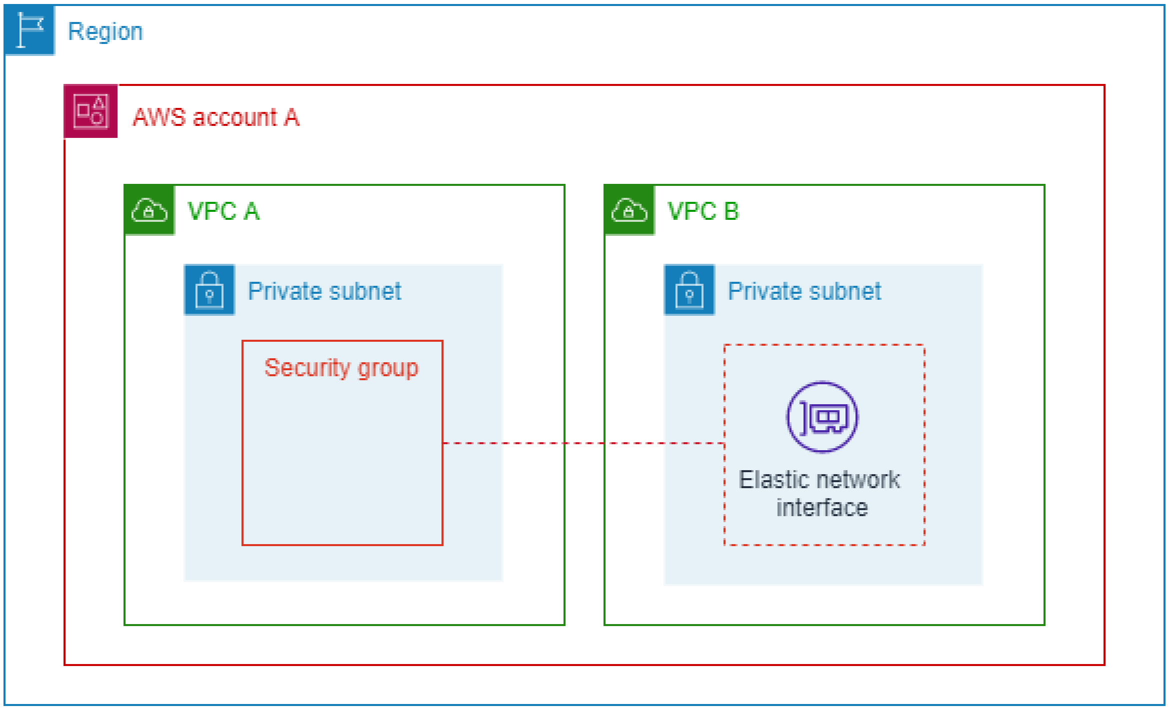
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security groups**.
3. Select the security group and choose **Actions**, **Delete security groups**.
4. If you selected more than one security group, you are prompted for confirmation. If some of the security groups can't be deleted, we display the status of each security group, which indicates whether it will be deleted. To confirm deletion, enter **Delete**.
5. Choose **Delete**.

To delete a security group using the AWS CLI

Use the delete-security-group command.

Associate security groups with multiple VPCs

If you have workloads running in multiple VPCs that share network security requirements, you can use the Security Group VPC Associations feature to associate a security group with multiple VPCs in the same Region. This enables you to manage and maintain security groups in one place for multiple VPCs in your account.



The diagram above shows AWS account A with two VPCs in it. Each of the VPCs has workloads running in a private subnet. In this case, workloads in VPC A and B subnets share the same network traffic requirements, so Account A can use the Security Group VPC associations feature to associate the security group in VPC A with VPC B. Any updates made to the associated security group are automatically applied to the traffic to workloads in the VPC B subnet.

Requirements of the Security Group VPC Associations feature

- You must own the VPC or have one of the VPC subnets shared with you to associate a security group with the VPC.
- The VPC and security group must be in the same AWS Region.
- You cannot associate a default security group with another VPC or associate a security group with a default VPC.
- Both the security group owner and the VPC owner can view the security group VPC associations.

Services that support this feature

- Amazon API Gateway (REST APIs only)
- AWS Auto Scaling
- AWS CloudFormation
- Amazon EC2
- Amazon EFS
- Amazon EKS
- Amazon FSx
- AWS PrivateLink
- Amazon Route 53
- Elastic Load Balancing
  - Application Load Balancer
  - Network Load Balancer

Associate a security group with another VPC

This section explains how to use the AWS Management Console and the AWS CLI to associate a security group with VPCs.

AWS Management Console

To associate a security group with another VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Security groups**.
3. Choose a security group to view the details.
4. Choose the **VPC associations** tab.
5. Choose **Associate VPC**.
6. Under **VPC ID**, choose a VPC to associate with the security group.
7. Choose **Associate VPC**.

Command line

To associate a security group with another VPC

1. Create a VPC association with associate-security-group-vpc.
2. Check the status of a VPC association with describe-security-group-vpc-associations and wait for the status to be `associated`.

The VPC is now associated with the security group.

Once you've associated the VPC with the security group, you can, for example, launch an instance into the VPC and choose this new security group or reference this security group in an existing security group rule.

Disassociate a security group from another VPC

This section explains how to use the AWS Management Console and the AWS CLI to disassociate a security group from VPCs. You may want to do this if your goal is to delete the security group. Security groups cannot be deleted if they are associated. You can only diassociate a security group if there are no network interfaces in the associated VPC using that security group.

AWS Management Console

To disassociate a security group from a VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Security groups**.
3. Choose a security group to view the details.
4. Choose the **VPC associations** tab.
5. Choose **Disassociate VPC**.
6. Under **VPC ID**, choose a VPC to disassociate from the security group.
7. Choose **Disassociate VPC**.
8. View the **Status** of the disassociation in the VPC associations tab and wait for the status to be `disassociated`.

Command line

To disassociate a security group from a VPC

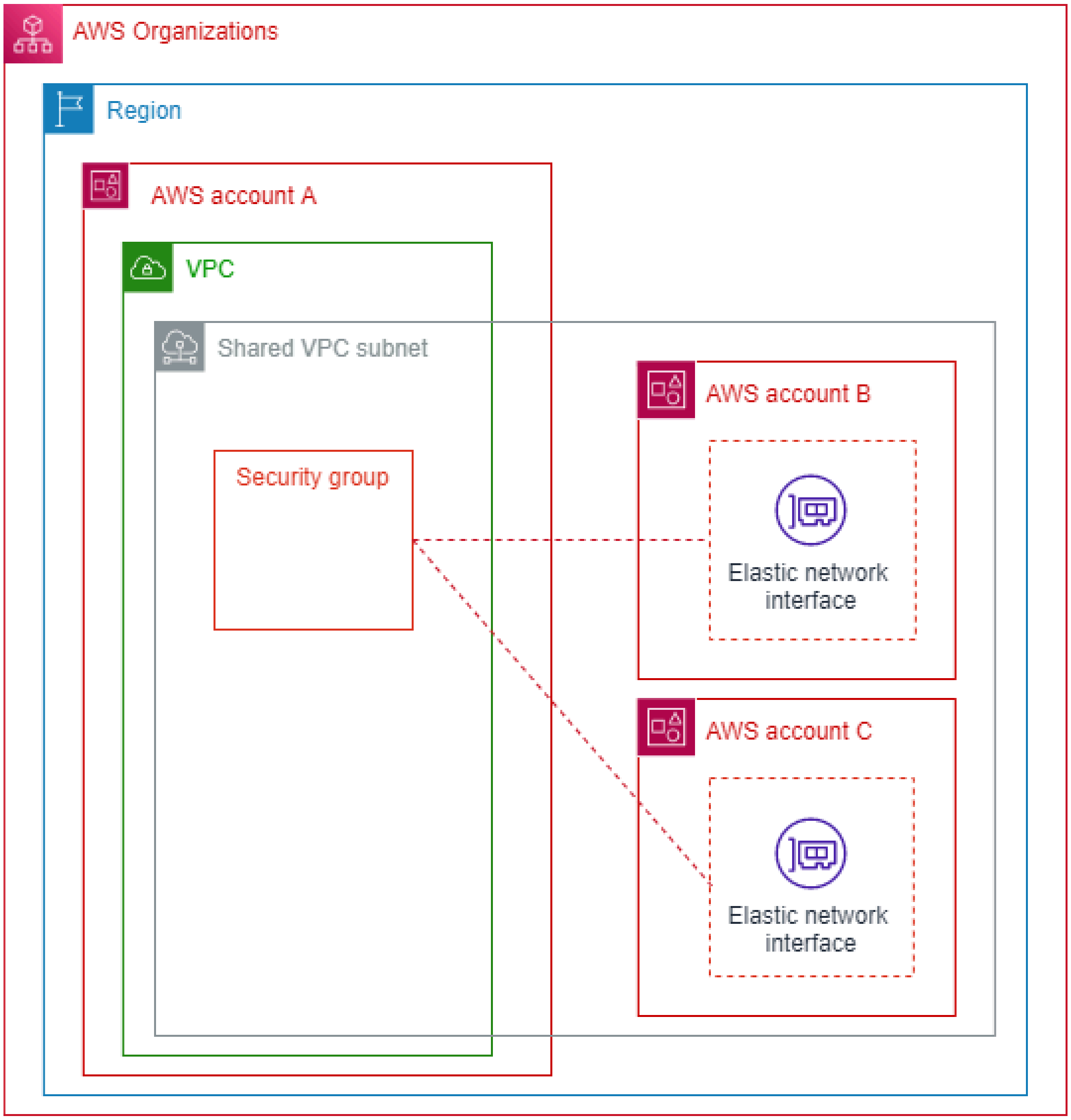
1. Disassociate a VPC association with disassociate-security-group-vpc.
2. Check the status of a VPC disassociation with describe-security-group-vpc-associations and wait for the status to be `disassociated`.

The VPC is now disassociated with the security group.

Share security groups with AWS Organizations

The Shared Security Group feature enables you to share a security group with other AWS Organizations accounts within the same AWS Region and make the security group available to be used by those accounts.

The following diagram demonstrates how you can use the Shared Security Group feature to simplify security group management across accounts in your AWS Organizations:



This diagram shows three accounts that are part of the same Organization. Account A shares a VPC subnet with Accounts B and C. Account A shares the security group with Accounts B and C using the Shared Security Group feature. Accounts B and C then use that security group when they launch instances in the shared subnet. This enables Account A to manage the security group; any updates to the security group apply to the resources that Accounts B and C have running in the shared VPC subnet.

Requirements of the Shared Security Group feature

- This feature is only available for accounts in the same Organization in AWS Organizations. Resource sharing must be enabled in AWS Organizations.
- The account that shares the security group must own both the VPC and the security group.

- You cannot share default security groups.
- You cannot share security groups that are in a default VPC.
- Participant accounts can create security groups in a shared VPC but they cannot share those security groups.
- A minimum set of permissions is required for an IAM principal to share a security group with AWS RAM. Use the `AmazonEC2FullAccess` and `AWSResourceAccessManagerFullAccess` managed IAM policies to ensure your IAM principals have the required permissions to share and use shared security groups. If you use a custom IAM policy, the `c2:PutResourcePolicy` and `ec2:DeleteResourcePolicy` actions are required. These are permission-only IAM actions. If an IAM principal doesn't have these permissions granted, an error will occur when attempting to share the security group using the AWS RAM.

Services that support this feature

- Amazon API Gateway
- Amazon EC2
- Amazon ECS
- Amazon EFS
- Amazon EKS
- Amazon EMR
- Amazon FSx
- Amazon ElastiCache
- AWS Elastic Beanstalk
- AWS Glue
- Amazon MQ
- Amazon SageMaker AI
- Elastic Load Balancing
  - Application Load Balancer
  - Network Load Balancer

How this feature affects existing quotas

Security group quotas apply. For the 'Security groups per network interface' quota, however, if a participant uses both owned and shared groups on an Elastic network interface (ENI), the minimum of owner and participant's quota applies.

Example to demonstrate how the quota is affected by this feature:

- Owner account quota: 4 security groups per interface
- Participant account quota: 5 security groups per interface.
- Owner shares groups SG-O1, SG-O2, SG-O3, SG-O4, SG-O5 with participant. Participant already has groups of their own in the VPC: SG-P1, SG-P2, SG-P3, SG-P4, SG-P5.
- If participant creates an ENI and uses only their owned groups, they can associate all 5 security groups (SG-P1, SG-P2, SG-P3, SG-P4, SG-P5) because that's their quota.
- If the participant creates an ENI and uses any shared groups on it, they can only associate up to 4 groups. In this case, the quota for such an ENI is the minimum of owner and participant's quotas. Possible valid configurations will look like this:
  - SG-O1, SG-P1, SG-P2, SG-P3
  - SG-O1, SG-O2, SG-O3, SG-O4

Share a security group

This section explains how to use the AWS Management Console and the AWS CLI to share a security group with other accounts in your Organization.

AWS Management Console

To share a security group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Security groups**.
3. Choose a security group to view the details.
4. Choose the **Sharing** tab.
5. Choose **Share security group**.
6. Choose **Create resource share**. As a result, the AWS RAM console opens where you'll create the resource share for the security group.
7. Enter a **Name** for the resource share.

8. Under **Resources - optional**, choose **Security Groups**.
9. Choose a security group. The security group cannot be a default security group and cannot be associated with the default VPC.
10. Choose **Next**.
11. Review the actions that principals will be allowed to perform and choose **Next**.
12. Under **Principals - optional**, choose **Allow sharing only within your organization**.
13. Under **Principals**, select one of the following principal types and enter the appropriate numbers:

    - **AWS account**: The account number of an account in your Organization.
    - **Organization**: The AWS Organizations ID.
    - **Organizational unit (OU)**: The ID of an OU in the Organization.
    - **IAM role**: The ARN of an IAM role. The account that created the role must be a member of the same Organization as the account creating this resource share.
    - **IAM user**: The ARN of an IAM user. The account that created the user must be a member of the same Organization as the account creating this resource share.
    - **Service principal**: You cannot share a security group with a service principal.
14. Choose **Add**.
15. Choose **Next**.
16. Choose **Create resource share**.
17. Under **Shared resources**, wait to see the **Status** of `Associated`. If there is a security group association failure, it may be due to one of the limitations listed above. View the details of the security group and the **Sharing** tab on the details page to see any messages related to why a security group may not be shareable.
18. Return to the VPC console security group list.
19. Choose the security group you shared.
20. Choose the **Sharing** tab. Your AWS RAM resource should be visible there. If it’s not, the resource share creation may have failed and you may need to recreate it.

Command line

To share a security group

1. You must first create a resource share for the security group that you want to share with AWS RAM. For steps on how to create a resource share with AWS RAM using the AWS CLI, see Creating a resource share in AWS RAM in the *AWS RAM User Guide*
2. To view created resource share associations, use get-resource-share-associations.

The security group is now shared. You can select the security group when launching an EC2 instance in a shared subnet within the same VPC.

Stop sharing a security group

This section explains how to use the AWS Management Console and the AWS CLI to stop sharing a security group with other accounts in your Organization.

AWS Management Console

To stop sharing a security group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Security groups**.
3. Choose a security group to view the details.
4. Choose the **Sharing** tab.
5. Choose a security group resource share and choose **Stop sharing**.
6. Choose **Yes, stop sharing**.

Command line

To stop sharing a security group

Delete the resource share with delete-resource-share.

The security group is no longer being shared. After the owner stops sharing a security group, the following rules apply:

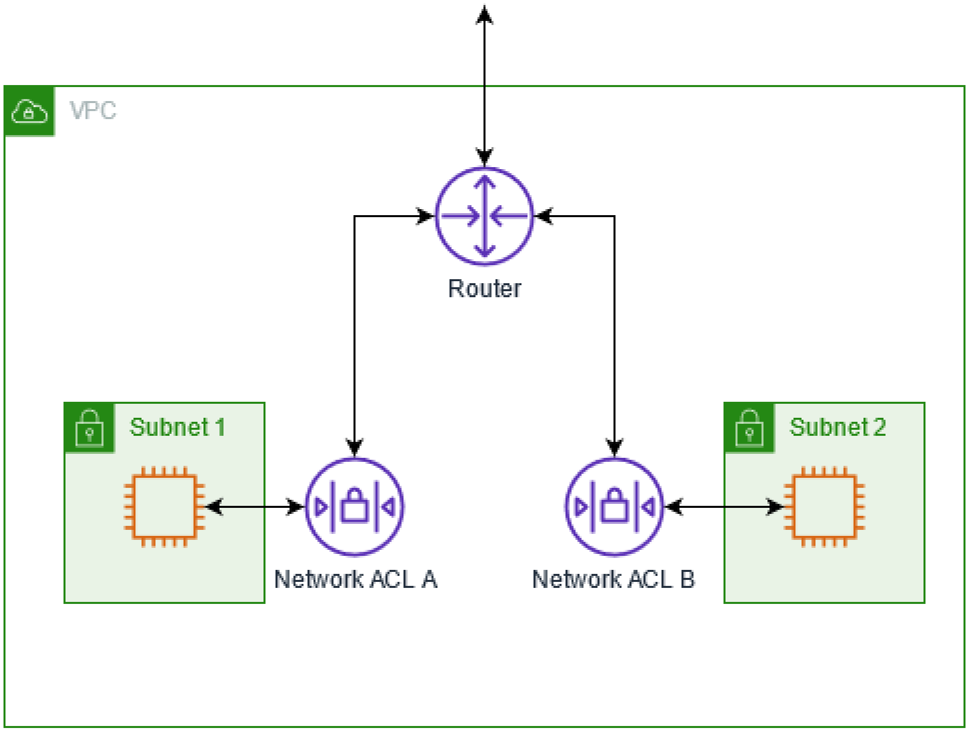
- Existing participant Elastic Network Interfaces (ENIs) continue to get any security group rule updates that are made to unshared security groups. Unsharing only prevents the participant from creating new associations with the unshared group.
- Participants can no longer associate the unshared security group with any ENIs they own.
- Participants can describe and delete the ENIs that are still associated with unshared security groups.
- If participants still have ENIs associated with the unshared security group, the owner cannot delete the unshared security group. The owner can only delete the security group after participants disassociate (remove) the security group from all their ENIs.
- Participants cannot launch new EC2 instances using an ENI associated with an unshared security group.

Control subnet traffic with network access control lists

A *network access control list (ACL)* allows or denies specific inbound or outbound traffic at the subnet level. You can use the default network ACL for your VPC, or you can create a custom network ACL for your VPC with rules that are similar to the rules for your security groups in order to add an additional layer of security to your VPC.

There is no additional charge for using network ACLs.

The following diagram shows a VPC with two subnets. Each subnet has a network ACL. When traffic enters the VPC (for example, from a peered VPC, VPN connection, or the internet), the router sends the traffic to its destination. Network ACL A determines which traffic destined for subnet 1 is allowed to enter subnet 1, and which traffic destined for a location outside subnet 1 is allowed to leave subnet 1. Similarly, network ACL B determines which traffic is allowed to enter and leave subnet 2.



For information about the differences between security groups and network ACLs, see Compare security groups and network ACLs.

Contents

- Network ACL basics
- Network ACL rules
- Default network ACL for a VPC
- Custom network ACLs for your VPC
- Path MTU Discovery and network ACLs
- Create a network ACL for your VPC
- Manage network ACL associations for your VPC
- Delete a network ACL for your VPC
- Example: Control access to instances in a subnet

Network ACL basics

The following are the basic things to know about network ACLs before you begin.

Network ACL associations

- 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 create a custom network ACL and associate it with a subnet to allow or deny specific inbound or outbound traffic at the subnet level.
- You can associate a network ACL with multiple subnets. However, a subnet can be associated with only one network ACL at a time. When you associate a network ACL with a subnet, the previous association is removed.

Network ACL rules

- A network ACL has inbound rules and outbound rules. There are quotas (or limits) to the number of rules you can have per network ACL. Each rule can either allow or deny traffic. Each rule has a number from 1 to 32766. We evaluate the rules in order, starting with the lowest numbered rule, when deciding whether allow or deny traffic. If the traffic matches a rule, the rule is applied and we do not evaluate any additional rules. We recommend that you start by creating rules in increments (for example, increments of 10 or 100) so that you can insert new rules later on, if needed.
- We evaluate the network ACL rules when traffic enters and leaves the subnet, not as it is routed within a subnet.
- NACLs are *stateless*, which means that information about previously sent or received traffic is not saved. If, for example, you create a NACL rule to allow specific inbound traffic to a subnet, responses to that traffic are not automatically allowed. This is in contrast to how security groups work. Security groups are *stateful*, which means that information about previously sent or received traffic is saved. If, for example, a security group allows inbound traffic to an EC2 instance, responses are automatically allowed regardless of outbound security group rules.

Limitations

- There are quotas (also known as limits) for network ACLs. For more information, see Network ACLs.
- Network ACLs can't block DNS requests to or from the Route 53 Resolver (also known as the VPC +2 IP address or AmazonProvidedDNS). To filter DNS requests through the Route 53 Resolver, you can enable Route 53 Resolver DNS Firewall.

- Network ACLs can't block traffic to the Instance Metadata Service (IMDS). To manage access to IMDS, see Configure the instance metadata options in the *Amazon EC2 User Guide*.
- Network ACLs do not filter traffic destined to and from the following:
  - Amazon Domain Name Services (DNS)
  - Amazon Dynamic Host Configuration Protocol (DHCP)
  - Amazon EC2 instance metadata
  - Amazon ECS task metadata endpoints
  - License activation for Windows instances
  - Amazon Time Sync Service
  - Reserved IP addresses used by the default VPC router

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.

For example rules, see the section called “Example: Control access to instances in a subnet”.

Considerations

- There are quotas (also known as limits) for the number of rules per network ACLs. For more information, see Amazon VPC quotas.
- When you add or delete a rule from an ACL, any subnets that are associated with the ACL are subject to the change. The changes take effect after a short period.
- 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.
- You might want to add a deny rule in a situation where you must open a wide range of ports, but there are certain ports within the range that you want to deny. Be sure to give the deny rule a lower number than the rule that allows the wider range of port traffic.
- If you add and delete rules from a network ACL at the same time, be careful. If you delete inbound or outbound rules and then add more new entries than are allowed (see Amazon VPC quotas, the entries selected for deletion are removed and new entries *are not added*. This can cause unexpected connectivity issues and prevent access to and from your VPC.

Default network ACL for a VPC

Your virtual private cloud (VPC) automatically comes with a default network ACL. A 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 rules where the rule number is an asterisk (*). These rules ensure that if a packet doesn't match any of the other numbered rules, it's denied.

You can modify a default network ACL by adding rules or removing the default numbered rules. You can't delete a rule where the rule number is an asterisk.

Default inbound rules

The following table shows the default inbound rules for a default network ACL. The rules for IPv6 are added only if you create the VPC with an associated IPv6 CIDR block or associate an IPv6 CIDR block with the VPC. However, if you've modified the inbound rules of a default network ACL, we do not add the rule that allows all inbound IPv6 traffic when you associate an IPv6 block with the VPC.

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

Default outbound rules

The following table shows the default outbound rules for a default network ACL. The rules for IPv6 are added only if you create the VPC with an associated IPv6 CIDR block or associate an IPv6 CIDR block with the VPC. However, if you've modified the outbound rules of a default network ACL, we do not add the rule that allows all outbound IPv6 traffic when you associate an IPv6 block with the VPC.

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 ACLs for your VPC

You can create a custom network ACL and associate it with a subnet to allow or deny specific inbound or outbound traffic at the subnet level. For more information, see the section called "Create a network ACL".

Each network ACL includes a default inbound rule and a default outbound rule whose rule number is an asterisk (*). These rules ensure that if a packet doesn't match any of the other rules, it's denied.

You can modify a network ACL by adding or removing rules. You can't delete a rule where the rule number is an asterisk.

For every rule that you add, there must be an inbound or outbound rule that allows response traffic. For more information about how to select the appropriate ephemeral port range, see Ephemeral ports.

Example inbound rules

The following table shows example inbound rules for a network ACL. The rules for IPv6 are added only if the VPC has an associated IPv6 CIDR block. IPv4 and IPv6 traffic are evaluated separately. Therefore, none of the rules for IPv4 traffic apply to IPv6 traffic. You can add IPv6 rules next to the corresponding IPv4 rules, or add the IPv6 rules after the last IPv4 rule.

As a packet comes to the subnet, we evaluate it against the inbound rules of the network ACL that is associated with the subnet, starting with the lowest numbered rule. For example, suppose there is IPv4 traffic destined for the HTTPS port (443). The packet doesn't match rule 100 or 105. It matches rule 110, which allows the traffic into the subnet. If the packet had been destined for port 139 (NetBIOS), it wouldn't match any of the numbered rules, so the * rule for IPv4 traffic ultimately denies the packet.

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.

Rule #	Type	Protocol	Port range	Source	Allow/ Deny	Comments
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).
140	Custom TCP	TCP	*32768-65535*	0.0.0.0/0	ALLOW	Allows inbound return IPv4 traffic from the internet (for requests that originate in the subnet).
145	Custom TCP	TCP	*32768-65535*	::/0	ALLOW	Allows inbound return IPv6 traffic from the internet (for requests that originate in the subnet).

Rule #	Type	Protocol	Port range	Source	Allow/ Deny	Comments
*	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).

Example outbound rules

The following table shows example outbound rules for a custom network ACL. The rules for IPv6 are added only if the VPC has an associated IPv6 CIDR block. IPv4 and IPv6 traffic are evaluated separately. Therefore, none of the rules for IPv4 traffic apply to IPv6 traffic. You can add IPv6 rules next to the corresponding IPv4 rules, or add the IPv6 rules after the last IPv4 rule.

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

Rule #	Type	Protocol	Port range	Destination	Allow/ Deny	Comments
						the subnet to the internet.
115	HTTPS	TCP	443	::/0	ALLOW	Allows outbound IPv6 HTTPS traffic from the subnet to the internet.
120	Custom TCP	TCP	*1024-6553 5*	*192.0.2.0 /24*	ALLOW	Allows outbound responses to SSH traffic from your home network.
140	Custom TCP	TCP	*32768-655 35*	0.0.0.0/0	ALLOW	Allows outbound IPv4 responses to clients on the internet (for example, serving webpages).
145	Custom TCP	TCP	*32768-655 35*	::/0	ALLOW	Allows outbound IPv6 responses to clients on the internet (for example, serving webpages).
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all outbound IPv4 traffic not already handled by a preceding rule.
*	All traffic	All	All	::/0	DENY	Denies all outbound IPv6 traffic not already handled by a preceding rule.

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 operating system of the instance.

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.

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 the following:

- Network ACLs for your Application Load Balancer
- Network ACLs for your Network Load Balancer
- Network ACLs for your Classic Load Balancer

Troubleshoot reachability issues

Reachability Analyzer is a static configuration analysis tool. Use Reachability Analyzer to analyze and debug network reachability between two resources in your VPC. Reachability Analyzer produces hop-by-hop details of the virtual path between these resources when they are reachable, and identifies the blocking component otherwise. For example, it can identify missing or misconfigured network ACL rules.

For more information, see the Reachability Analyzer Guide.

Path MTU Discovery and network ACLs

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*.

Create a network ACL for your VPC

The following tasks show you how to create a network ACL, add rules to the network ACL, and then associate the network ACL with a subnet.

Tasks

- Step 1: Create a network ACL
- Step 2: Add rules
- Step 3: Associate a subnet with a network ACL
- (Optional) Manage network ACLs using Firewall Manager

Step 1: Create a network ACL

You can create a custom network ACL for your VPC. The initial rules for a custom network ACL block all inbound and outbound traffic. Your new custom network ACL is not associated with a subnet by default and must be explicitly associated with subnets.

To create a network ACL using the console

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. (Optional) For **Name**, enter a name for your network ACL.
5. For **VPC**, select the VPC.
6. (Optional) For **Tags**, choose **Add tag** and then enter a tag key and a tag value.
7. Choose **Create network ACL**.

To create a network ACL using the command line

- create-network-acl (AWS CLI)
- New-EC2NetworkAcl (AWS Tools for Windows PowerShell)

Step 2: Add rules

You can add rules that allow or deny inbound or outbound traffic.

We process the rules in order, starting with the rule 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.

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 using the console

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.
4. To add an inbound rule, do the following:

   a. Choose the **Inbound rules** tab.
   b. Choose **Edit inbound rules**, **Add new rule**.
   c. Enter a rule number that is not already in use, a type, protocol, port range, source, and whether to allow or deny the traffic. For some types, we fill in the protocol and port for you. If you are prompted for a port range, enter a port number or a port range (for example, 49152-65535).

      To use a protocol that's not listed, choose **Custom Protocol** for the type and then select the protocol. For more information, see IANA Protocol Numbers.
   d. Choose **Save changes**.
5. To add an outbound rule, do the following:

   a. Choose the **Outbound rules** tab.
   b. Choose **Edit outbound rules**, **Add new rule**.
   c. Enter a rule number that is not already in use, a type, protocol, port range, source, and whether to allow or deny the traffic. For some types, we fill in the protocol and port

for you. If you are prompted for a port range, enter a port number or a port range (for example, 49152-65535).

To use a protocol that's not listed, choose **Custom Protocol** for the type and then select the protocol. For more information, see IANA Protocol Numbers.

d. Choose **Save changes**.

To add a rule to a network ACL using the command line

- create-network-acl-entry (AWS CLI)
- New-EC2NetworkAclEntry (AWS Tools for Windows PowerShell)

To replace a rule in a network ACL using the command line

- replace-network-acl-entry (AWS CLI)
- Set-EC2NetworkAclEntry (AWS Tools for Windows PowerShell)

To delete a rule from a network ACL using the command line

- delete-network-acl-entry (AWS CLI)
- Remove-EC2NetworkAclEntry (AWS Tools for Windows PowerShell)

Step 3: 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**.

(Optional) Manage network ACLs using Firewall Manager

AWS Firewall Manager simplifies your network ACL administration and maintenance tasks across multiple accounts and subnets. You can use Firewall Manager to monitor accounts and subnets in your organization and to automatically apply the network ACL configurations that you've defined. Firewall Manager is particularly useful when you want to protect your entire organization, or if you frequently add new subnets that you want to automatically protect from a central administrator account.

With a Firewall Manager network ACL policy, using a single administrator account, you can configure, monitor, and manage the minimum rule sets that you want to have defined in the network ACLs that you use across your organization. You specify which accounts and subnets in your organization are within scope of the Firewall Manager policy. Firewall Manager reports the compliance status of the network ACLs for in-scope subnets, and you can configure Firewall Manager to automate the remediation of noncompliant network ACLs.

For more information, see the following resources in the *AWS Firewall Manager Developer Guide*:

- AWS Firewall Manager prerequisites
- Setting up AWS Firewall Manager network ACL policies
- Using network ACL policies with Firewall Manager

Manage network ACL associations for your VPC

Each subnet is associated with one network ACL. When you first create a subnet, it is associated with the default network ACL for the VPC. You can create a custom network ACL and associate it with one or more subnets, replacing the previous network ACL association.

Tasks

- Describe your network ACL associations
- Change the subnets associated with a network ACL
- Change the network ACL associated with a subnet

Describe your network ACL associations

You can describe the network ACL that's associated with a subnet and you can also describe which subnets are associated with a network ACL.

To describe the network ACL associated 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 the subnet.
4. Select the **Network ACL** tab.

To describe the network ACL associated with a subnet using the AWS CLI

Use the following describe-network-acls command to list the network ACL associated with the specified subnet.

```
aws ec2 describe-network-acls --filters Name=association.subnet-
id,Values=subnet-0d2d1b81e0bc9c6d4 --query NetworkAcls[*].NetworkAclId
```

The following is example output.

```
[
    "acl-03701d1f82d8c3fd6"
]
```

To describe the subnets associated with a network ACL using the console

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.
4. Select the **Subnet associations** tab.

To describe the subnets associated with a network ACL using the AWS CLI

Use the following describe-network-acls command to list the subnets associated with the specified network ACL.

```
aws ec2 describe-network-acls --network-acl-ids acl-060415a18fcc9afde --query
 NetworkAcls[*].Associations[].SubnetId
```

The following is example output.

```
[
    "subnet-0d2d1b81e0bc9c6d4",
    "subnet-0e990c67809773b19",
    "subnet-0eb17d85f5dfd33b1",
    "subnet-0e01d500780bb7468"
]
```

Change the subnets associated with a network ACL

You can disassociate a custom network ACL from a subnet. After you disassociate a subnet from a custom network ACL, we automatically associate it with the default network ACL for the VPC. The changes take effect after a short period of time.

To change the subnets 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**.
3. Select the network ACL.
4. Choose **Actions**, **Edit subnet associations**.
5. Remove the subnet from **Selected subnets**.
6. Choose **Save changes**.

Change the network ACL associated with a subnet

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 for the VPC. If you create a custom network ACL, you apply the network ACL rules by associating the network ACL with one or more subnets.

After you change the network ACL for a subnet, the changes take effect after a short period of time.

To change the network ACL associated 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 the subnet.

4. Choose **Actions**, **Edit network ACL association**.
5. For **Network ACL ID**, select the network ACL to associate with the subnet, and review the inbound and outbound rules for the selected network ACL.
6. Choose **Save**.

To replace a network ACL association using the command line

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

Delete a network ACL for your VPC

When you are finished with a network ACL, you can delete it. You can't delete a network ACL if there are subnets associated with it. You can't delete the default network ACL.

To remove subnet associations from a network ACL using the console

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 subnets associated with each network ACL. This column is - if there are no associated subnets.
3. Select the network ACL.
4. Choose **Actions**, **Edit subnet associations**.
5. Remove the subnet associations.
6. Choose **Save changes**.

To describe your network ACLs, including associations, using the command line

- describe-network-acls (AWS CLI)
- Get-EC2NetworkAcl (AWS Tools for Windows PowerShell)

To replace a network ACL association using the command line

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

To delete a network ACL using the console

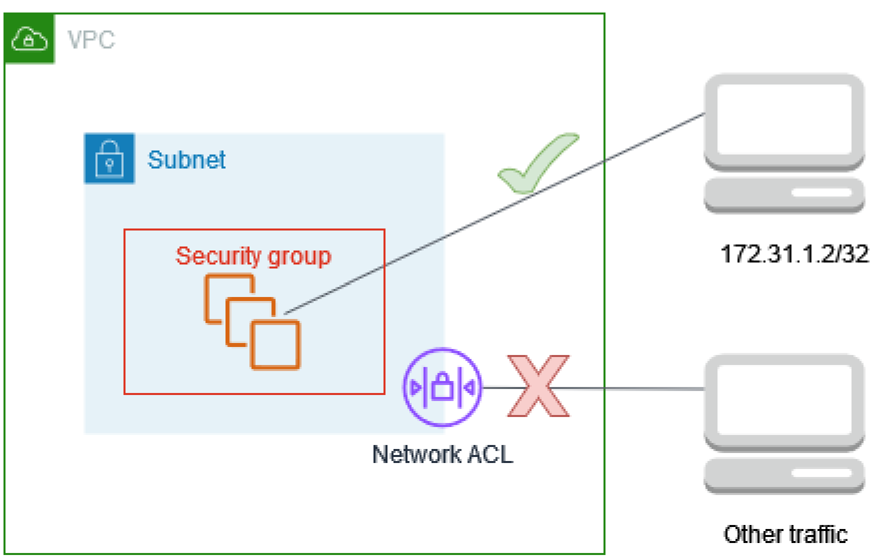
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Network ACLs**.
3. Select the network ACL.
4. Choose **Actions**, **Delete network ACLs**.
5. When prompted for confirmation, enter **`delete`** and then choose **Delete**.

To delete a network ACL using the command line

- 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 the subnet can communicate with each other, and are accessible from a trusted remote computer in order to perform administrative tasks. The remote computer might be a computer in your local network, as shown in the diagram, or it might be an instance in a different subnet or VPC. The network ACL rules for the subnet, and the security group rules for the instances, allow access from the IP address of your remote computer. All other traffic from the internet or other networks is denied.



Using a network ACL gives you the flexibility to change the security groups or security group rules for your instances while relying on the network ACL as a backup layer of defense. For example, if you accidentally update the security group to allow inbound SSH access from anywhere, but the network ACL allows access only from the IP address range of the remote computer, then the network ACL denies inbound SSH traffic from any other IP addresses.

Network ACL rules

The following are example inbound rules for the network ACL associated with the subnet. These rules apply to all instances in the subnet.

Rule #	Type	Protocol	Port range	Source	Allow/Deny	Comments
100	SSH	TCP	22	*172.31.1.2/32*	ALLOW	Allow inbound traffic from the remote computer.
*	All traffic	All	All	0.0.0.0/0	DENY	Deny all other inbound traffic.

The following are example outbound rules for the network ACL associated with the subnet. Network ACLs are stateless. Therefore, you must include a rule that allows responses to the inbound traffic.

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.

Rule #	Type	Protocol	Port range	Destination	Allow/Deny	Comments
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all other outbound traffic.

Security group rules

The following are example inbound rules for the security group associated with the instances. These rules apply to all instances associated with the security group. A user with the private key for the key pair associated with the instances can connect to the instances from the remote computer using SSH.

Protocol type	Protocol	Port range	Source	Comments
All traffic	All	All	*sg-1234567890abcdef0*	Allow communication between the instances associated with this security group.
SSH	TCP	22	*172.31.1.2/32*	Allow inbound SSH access from the remote computer.

The following are example outbound bound rules for the security group associated with the instances. Security groups are stateful. Therefore, you don't need a rule that allows responses to inbound traffic.

Protocol Type	Protocol	Port range	Destination	Comments
All traffic	All	All	*sg-1234567890abcdef0*	Allow communication between the instances associated with this security group.

Differences between network ACLs and security groups

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

Characteristic	Network ACL	Security group
Level of operation	Subnet level	Instance level
Scope	Applies to all instances in the associated subnets	Applies to all instances associated with the security group
Rule type	Allow and deny rules	Allow rules only
Rule evaluation	Evaluates rules in ascending order until a match for the traffic is found	Evaluates all rules before deciding whether to allow traffic
Return traffic	Must be explicitly allowed (stateless)	Automatically allowed (stateful)

Resilience in Amazon Virtual Private Cloud

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected using low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

AWS Regions are the primary building blocks, each representing a distinct geographical location housing multiple physically separated and isolated Availability Zones. These Availability Zones are connected through a low-latency, high-throughput, and highly redundant networking fabric, enabling seamless communication and data transfer between them.

The architecture of Availability Zones is a key differentiator, as they are designed to be far more robust and fault-tolerant than traditional single or multiple data center setups. By distributing resources across multiple Availability Zones within a Region, applications and databases can be engineered to automatically fail over between zones without any interruption to service. This level of redundancy and high availability is a critical requirement for mission-critical workloads and enables organizations to build resilient cloud-native solutions.

Furthermore, the scale and global reach of the AWS infrastructure empower customers to deploy their applications closer to end-users, reducing latency and improving the overall user experience. The availability of multiple Regions across the world also allows for effective data sovereignty and compliance, as customers can store and process data within the geographical boundaries required by their specific regulatory and business needs.

By leveraging the AWS global infrastructure, organizations can architect their cloud environments to be highly available, fault-tolerant, and scalable, with the flexibility to adapt to changing requirements and evolving business needs. This robust foundation is a key enabler for the successful implementation of modern, cloud-based applications and services.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

You can configure your VPCs to meet the resilience requirements for your workloads. For more information, see the following:

- Understand resiliency patterns and trade-offs (AWS Architecture Blog)
- Plan your network topology (AWS Well-Architected Framework)
- Amazon Virtual Private Cloud Connectivity Options (AWS Whitepapers)

Compliance validation for Amazon Virtual Private Cloud

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. 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 Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.

- 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.

Block public access to VPCs and subnets

VPC Block Public Access (BPA) is a centralized security feature that enables you to authoritatively prevent public internet access to VPC resources across an entire AWS account, ensuring compliance with security requirements while providing flexibility for specific exceptions and audit capabilities.

The VPC BPA feature has the following modes:

- **Bidirectional**: All traffic to and from internet gateways and egress-only internet gateways in this Region (except for excluded VPCs and subnets) is blocked.
- **Ingress-only**: All internet traffic to the VPCs in this Region (except for VPCs or subnets which are excluded) is blocked. Only traffic to and from NAT gateways and egress-only internet gateways is allowed because these gateways only allow outbound connections to be established.

You can also create "exclusions" for this feature for traffic you don't want to block. An exclusion is a mode that can be applied to a single VPC or subnet that exempts it from the account's VPC BPA mode and will allow bidirectional or egress-only access.

Exclusions can have either of the following modes:

- **Bidirectional**: All internet traffic to and from the excluded VPCs and subnets is allowed.
- **Egress-only**: Outbound internet traffic from the excluded VPCs and subnets is allowed. Inbound internet traffic to the excluded VPCs and subnets is blocked. This only applies when VPC BPA is set to Bidirectional.

Contents

- VPC BPA basics
- Assess impact of VPC BPA and monitor VPC BPA
- Advanced example

VPC BPA basics

This section covers important details about VPC BPA, including which services support it and how you can work with it.

Contents

- Regional availability
- AWS service impact and support
- VPC BPA limitations
- Control access to VPC BPA with an IAM policy
- Enable VPC BPA bidirectional mode for your account
- Change VPC BPA mode to ingress-only
- Create and delete exclusions
- Enable VPC BPA at the Organization level

Regional availability

VPC BPA is available in all commercial AWS Regions including GovCloud and China Regions.

In this guide, you'll also find information about using Network Access Analyzer and Reachability Analyzer with VPC BPA. Note that Network Access Analyzer and Reachability Analyzer are not available in all commercial Regions. For information about the regional availability of Network Access Analyzer and Reachability Analyzer, see Limitations in the *Network Access Analyzer Guide* and Considerations in the *Reachability Analyzer Guide*.

AWS service impact and support

The following resources and services support VPC BPA and traffic to these services and resources is impacted by VPC BPA:

- **Internet gateway**: All inbound and outbound traffic is blocked.
- **Egress-only internet gateway**: All outbound traffic is blocked. Egress-only internet gateways do not allow inbound traffic.
- **Gateway Load Balancer (GWLB)**: All inbound and outbound traffic is blocked even if the subnet containing GWLB endpoints is excluded.

- **NAT gateway**: All inbound and outbound traffic is blocked. NAT gateways require an internet gateway for internet connectivity.
- **Internet-facing Network Load Balancer**: All inbound and outbound traffic is blocked. Internet-facing Network Load Balancers require an internet gateway for internet connectivity.
- **Internet-facing Application Load Balancer**: All inbound and outbound traffic is blocked. Internet-facing Application Load Balancers require an internet gateway for internet connectivity.
- **Amazon CloudFront VPC Origins**: All inbound and outbound traffic is blocked.
- **AWS Direct Connect**: All inbound and outbound traffic that uses public virtual interfaces (public IPv4 or global unicast IPv6 addresses) is blocked. This traffic uses the internet gateway (or egress-only internet-gateway) for connectivity.
- **AWS Global Accelerator**: Inbound traffic to VPCs is blocked, whether or not the target is otherwise accessible from the internet.
- **AWS Network Firewall**: All inbound and outbound traffic is blocked even if the subnet containing firewall endpoints is excluded.
- **AWS Wavelength carrier gateway**: All inbound and outbound traffic is blocked.

Traffic related to private connectivity, such as traffic for the following services and resources, is not blocked or impacted by VPC BPA:

- AWS Client VPN
- AWS CloudWAN
- AWS Outposts local gateway
- AWS Site-to-Site VPN
- Transit gateway
- AWS Verified Access

Important

- If you are routing incoming and outgoing traffic through an appliance (such as a 3rd-party security or monitoring tool) running on an EC2 instance in a subnet, when using VPC BPA, that subnet needs to be an exclusion for traffic to flow in and out of it. Other subnets sending traffic to the appliance subnet and not to the internet gateway do not need to be added as exclusions.

- Traffic sent privately from resources in your VPC to other services running in your VPC, such as the EC2 DNS Resolver or Amazon OpenSearch Service, is allowed even when VPC BPA is turned on because it does not pass through an internet gateway in your VPC. It is possible that these services may make requests to resources outside of the VPC on your behalf, for example, in order to resolve a DNS query, and may expose information about the activity of resources within your VPC if not mitigated through other security controls.

VPC BPA limitations

VPC BPA ingress-only mode is not supported in Local Zones (LZs) where NAT gateways and egress-only internet gateways are not allowed.

Control access to VPC BPA with an IAM policy

For examples of IAM policies that allow/deny access to the VPC BPA feature, see Block public access to VPCs and subnets.

Enable VPC BPA bidirectional mode for your account

VPC BPA bidirectional mode blocks all traffic to and from internet gateways and egress-only internet gateways in this Region (except for excluded VPCs and subnets). For more information about exclusions, see Create and delete exclusions.

Important

We strongly recommend that you thoroughly review the workloads that require Internet access prior to enabling VPC BPA in your production accounts.

Note

- To enable VPC BPA on the VPCs and subnets in your account, you must own the VPCs and subnets.
- If you are currently sharing VPC subnets with other accounts, the VPC BPA mode enforced by the subnet owner applies to participant traffic as well, but participants can't control the VPC BPA settings that impact the shared subnet.

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Settings**.
3. Choose **Edit public access settings**.
4. Choose **Turn on block public access** and **Bidirectional**, then choose **Save changes**.
5. Wait for the **Status** to change to **On**. It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

VPC BPA Bidirectional mode is now on.

AWS CLI

1. Turn on VPC BPA:

```
aws ec2 --region us-east-2 modify-vpc-block-public-access-options --internet-gateway-block-mode block-bidirectional
```

   It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

2. View the status of VPC BPA:

```
aws ec2 --region us-east-2 describe-vpc-block-public-access-options
```

Change VPC BPA mode to ingress-only

VPC BPA ingress-only mode blocks all internet traffic to the VPCs in this Region (except for VPCs or subnets which are excluded). Only traffic to and from NAT gateways and egress-only internet gateways is allowed because these gateways only allow outbound connections to be established.

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Settings**.
3. Choose **Edit public access settings**.
4. Change the direction to **Ingress-only**.

5. Save the changes and wait for the status to be updated. It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

AWS CLI

1. Modify the VPC BPA block direction:

```
aws ec2 --region us-east-2 modify-vpc-block-public-access-options --internet-gateway-block-mode block-ingress
```

   It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

2. View the status of VPC BPA:

```
aws ec2 --region us-east-2 describe-vpc-block-public-access-options
```

Create and delete exclusions

A VPC BPA exclusion is a mode that can be applied to a single VPC or subnet that exempts it from the account's VPC BPA mode and will allow bidirectional or egress-only access. You can create VPC BPA exclusions for VPCs and subnets even when VPC BPA is not enabled on the account to ensure that there is no traffic disruption to the exclusions when VPC BPA is turned on. An exclusion for a VPC automatically applies to all subnets in the VPC.

You can create a maximum of 50 exclusions. For information about requesting a limit increase, see *VPC BPA exclusions per account* in [Amazon VPC quotas](#).

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Settings**.
3. In the **Block public access** tab, under **Exclusions**, do one of the following:
   - To delete an exclusion, select the exclusion and then choose **Actions** > **Delete exclusions**.
   - To create an exclusion, choose **Create exclusions** and continue with the next steps.
4. Choose a block direction:
   - **Bidirectional**: Allows all internet traffic to and from the excluded VPCs and subnets.

- **Egress-only**: Allows outbound internet traffic from the excluded VPCs and subnets. Blocks inbound internet traffic to the excluded VPCs and subnets. This setting applies when VPC BPA is set to **Bidirectional**.

5. Choose a VPC or subnet.
6. Choose **Create exclusions**.
7. Wait for the **Exclusion status** to change to **Active**. You may need to refresh the exclusion table to see the change.

The exclusion has been created.

AWS CLI

1. Modify the exclusion allow direction:

```
aws ec2 --region us-east-2 create-vpc-block-public-access-exclusion --subnet-id subnet-id --internet-gateway-exclusion-mode allow-bidirectional
```

2. It can take time for the exclusion status to update. To view the status of the exclusion:

```
aws ec2 --region us-east-2 describe-vpc-block-public-access-exclusions --exclusion-ids exclusion-id
```

Enable VPC BPA at the Organization level

If you are using AWS Organizations to manage accounts in your organization, you can use an AWS Organizations declarative policy to enforce VPC BPA on the accounts in the organization. For more information about the VPC BPA declarative policy, see Supported declarative policies in the *AWS Organizations User Guide*.

> **Note**
>
> - You can use the VPC BPA declarative policy to configure if exclusions are allowed, but you cannot create exclusions with the policy. To create exclusions, you still have to create them in the account that owns the VPC. For more information about creating VPC BPA exclusions, see Create and delete exclusions.

- If the VPC BPA declarative policy is enabled, in **Block public access** settings, you'll see **Managed by Declarative Policy** and you won't be able to modify VPC BPA settings at the account level.

Assess impact of VPC BPA and monitor VPC BPA

This section contains information on you can assess the impact of VPC BPA before you turn it on and how you monitor if traffic is being blocked after you turn it on.

Contents

- Assess the impact of VPC BPA with Network Access Analyzer
- Monitor VPC BPA impact with flow logs
- Track exclusion deletion with CloudTrail
- Verify connectivity is blocked with Reachability Analyzer

Assess the impact of VPC BPA with Network Access Analyzer

In this section, you'll use Network Access Analyzer to view the resources in your account that use an internet gateway *before* you enable VPC BPA and block access. Use this analysis to understand the impact of turning on VPC BPA in your account and blocking traffic.

Note

- Network Access Analyzer does not support IPv6; so you will not be able to use it to view the potential impact of VPC BPA on egress-only internet gateway outbound IPv6 traffic.
- You are charged for the analyses you perform with Network Access Analyzer. For more information, see Pricing in the *Network Access Analyzer Guide*.
- For information about the regional availability of Network Access Analyzer, see Limitations in the *Network Access Analyzer Guide*.

AWS Management Console

1. Open the AWS Network Insights console at https://console.aws.amazon.com/networkinsights/.

2. Choose **Network Access Analyzer**.
3. Choose **Create Network Access Scope**.
4. Choose **Assess impact of VPC Block Public Access** and choose **Next**.
5. The template is already configured to analyze traffic to and from the internet gateways in your account. You can view this under **Source** and **Destination**.
6. Choose **Next**.
7. Choose **Create Network Access Scope**.
8. Choose the scope you just created and choose **Analyze**.
9. Wait for the analysis to complete.
10. View the findings of the analysis. Each row under **Findings** shows a network path that a packet can take in a network to or from an internet gateway in your account. In this case, if you turn on VPC BPA and none of the VPCs and or subnets that appear in these findings are configured as VPC BPA exclusions, traffic to those VPCs and subnets will be restricted.
11. Analyze each finding to understand the impact of VPC BPA on resources in your VPCs.

The impact analysis is complete.

AWS CLI

1. Create a network access scope:

```
aws ec2 create-network-insights-access-scope --region us-east-2 --match-paths
 "Source={ResourceStatement={ResourceTypes=["AWS::EC2::InternetGateway"]}}"
 "Destination={ResourceStatement={ResourceTypes=["AWS::EC2::InternetGateway"]}}"
```

2. Start the scope analysis:

```
aws ec2 start-network-insights-access-scope-analysis  --region us-east-2 --
network-insights-access-scope-id nis-id
```

3. Get the results of the analysis:

```
aws ec2 get-network-insights-access-scope-analysis-findings  --region us-east-2
 --network-insights-access-scope-analysis-id nisa-0aa383a1938f94cd1 --max-items
 1
```

The results show the traffic to and from the internet gateways in all the VPCs in your account. The results are organized as "findings". "FindingId": "AnalysisFinding-1" indicates that this is the first finding in the analysis. Note that there are multiple findings and each indicates a traffic flow that will be impacted by turning on VPC BPA. The first finding will show that traffic started at an internet gateway ("SequenceNumber": 1), passed to an NACL ("SequenceNumber": 2) to a security group ("SequenceNumber": 3) and ended at an instance ("SequenceNumber": 4).

4. Analyze the findings to understand the impact of VPC BPA on resources in your VPCs.

The impact analysis is complete.

Monitor VPC BPA impact with flow logs

VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from Elastic network interfaces in your VPC. You can use this feature to monitor traffic that is blocked by VPC BPA from reaching your instance network interfaces.

Create a flow log for your VPC using the steps in Work with flow logs.

When you create the flow log, make sure you use a custom format that includes the field `reject-reason`.

When you view the flow logs, if traffic to an ENI is rejected due to VPC BPA, you'll see a `reject-reason` of BPA in the flow log entry.

In addition to the standard limitations for VPC flow logs, note the following limitations specific to VPC BPA:

- Flow logs for VPC BPA do not include skipped records.
- Flow logs for VPC BPA do not include `bytes` even if you include the `bytes` field in your flow log.

Track exclusion deletion with CloudTrail

This section explains how you can use AWS CloudTrail to monitor and track the deletion of VPC BPA exclusions.

AWS Management Console

You can view any deleted exclusions in the **CloudTrail Event history** by looking up **Resource type** > `AWS::EC2::VPCBlockPublicAccessExclusion` in the AWS CloudTrail console at https://console.aws.amazon.com/cloudtrailv2/.

AWS CLI

You can use the `lookup-events` command to view the events related to deleting exclusions:

```
aws cloudtrail lookup-events --lookup-attributes
 AttributeKey=ResourceType,AttributeValue=AWS::EC2::VPCBlockPublicAccessExclusion
```

Verify connectivity is blocked with Reachability Analyzer

VPC Reachability Analyzer can be used to evaluate whether or not certain network paths can be reached given your network configuration, including VPC BPA settings.

For information about the regional availability of Reachability Analyzer, see Considerations in the *Reachability Analyzer Guide*.

AWS Management Console

1. Open the AWS Network Insights console at https://console.aws.amazon.com/networkinsights/home#ReachabilityAnalyzer.
2. Click **Create and analyze path**.
3. For the **Source Type**, choose **Internet Gateways** and select the internet gateway you want to block traffic from the **Source dropdown**.
4. For the **Destination Type**, choose **Instances** and select the instance you want to block traffic to from the **Destination** dropdown.
5. Click **Create and analyze path**.
6. Wait for the analysis to complete. It could take a few minutes.
7. Once complete, you should see that the **Reachability Status** is **Not reachable** and that the **Path details** shows that `VPC_BLOCK_PUBLIC_ACCESS_ENABLED` is the cause of this reachability issue.

AWS CLI

1. Create a network path using the ID of the Internet Gateway you want to block traffic from (source) and the ID of the instance you want to block traffic to (destination):

```
aws ec2 --region us-east-2 create-network-insights-path --source igw-id --
destination instance-id --protocol TCP
```

2. Start an analysis on the network path:

```
aws ec2 --region us-east-2 start-network-insights-analysis --network-insights-
path-id nip-id
```

3. Retrieve the results of the analysis:

```
aws ec2 --region us-east-2 describe-network-insights-analyses --network-
insights-analysis-ids nia-id
```

4. Verify that `VPC_BLOCK_PUBLIC_ACCESS_ENABLED` is the `ExplanationCode` for the lack of reachability.

Advanced example

This section contains an advanced example that will help you understand how VPC Block Public Access feature works in different scenarios. Each scenario builds off the scenario before it, so it's important to complete the steps in order.

Important

Do not go through this example in a production account. We strongly recommend that you thoroughly review the workloads that require Internet access prior to enabling VPC BPA in your production accounts.

Note

To fully understand the VPC BPA feature, you'll need certain resources in your account. In this section, we provide an AWS CloudFormation template that you can use to provision the resources you need to fully understand how this feature works. There are costs associated with the resources you provision with the CloudFormation template and the analyses you perform with Network Access Analyzer and Reachability Analyzer. If you use the template in this section, ensure that you complete the Cleanup steps when you're done with this example.

Contents

- Deploy CloudFormation template (optional)
- View the impact of VPC BPA with Network Access Analyzer
- Scenario 1 - Connect to instances without VPC BPA turned on
- Scenario 2 - Turn on VPC BPA in Bidirectional mode
- Scenario 3 - Change VPC BPA to Ingress-only mode
- Scenario 4 - Create an exclusion
- Scenario 5 - Modify exclusion mode
- Scenario 6 - Modify VPC BPA mode
- Cleanup

Deploy CloudFormation template (optional)

To demonstrate how this feature works, you need a VPC, subnets, instances, and other resources. To make it easier to complete this demonstration, we’ve provided an AWS CloudFormation template below that you can use to quickly spin up the resources required for the scenarios in this demo. This step is optional and you may want to just view the diagrams in the Scenarios in this section.

Note

- There are costs associated with the resources you create in this section with the CloudFormation template, such as the cost of the NAT gateway and public IPv4 addresses. To avoid excess costs, ensure that you complete the Cleanup steps to remove all resources created for the purpose of this example.
- This CloudFormation template creates the underlying resources needed for VPC BPA but does not enable the VPC BPA feature itself. The resources deployed here are intended

to help you understand and test VPC BPA functionality once you choose to enable it separately.

The template creates the following resources in your account:

- Egress-only internet gateway
- Internet gateway
- NAT gateway
- Two public subnets
- One private subnet
- Two EC2 instances with public and private IPv4 addresses
- One EC2 instance with an IPv6 address and a private IPv4 address
- One EC2 instance with a private IPv4 address only
- Security group with SSH and ICMP inbound traffic allowed and ALL outbound traffic allowed
- VPC flow log
- One EC2 Instance Connect endpoint in Subnet B

Copy the template below and save it to a .yaml file.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Creates a VPC with public and private subnets, NAT gateway, and EC2
 instances for VPC BPA.

Parameters:
  InstanceAMI:
    Description: ID of the Amazone Machine Image (AMI) to use with the instances
 launched by this template
    Type: AWS::EC2::Image::Id
  InstanceType:
    Description: EC2 Instance type to use with the instances launched by this template
    Type: String
    Default: t2.micro

Resources:

  # VPC
  VPCBPA:
```

```
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true
      InstanceTenancy: default
      Tags:
        - Key: Name
          Value: VPC BPA

  # VPC IPv6 CIDR
  VPCBPAIpv6CidrBlock:
    Type: AWS::EC2::VPCCidrBlock
    Properties:
      VpcId: !Ref VPCBPA
      AmazonProvidedIpv6CidrBlock: true

  # EC2 Key Pair
  VPCBPAKeyPair:
    Type: AWS::EC2::KeyPair
    Properties:
      KeyName: vpc-bpa-key

  # Internet Gateway
  VPCBPAInternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: VPC BPA Internet Gateway

  VPCBPAInternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      VpcId: !Ref VPCBPA
      InternetGatewayId: !Ref VPCBPAInternetGateway

  # Egress-Only Internet Gateway
  VPCBPAEgressOnlyInternetGateway:
    Type: AWS::EC2::EgressOnlyInternetGateway
    Properties:
      VpcId: !Ref VPCBPA

  # Subnets
```

```
  VPCBPAPublicSubnetA:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPCBPA
      CidrBlock: 10.0.1.0/24
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: VPC BPA Public Subnet A

  VPCBPAPublicSubnetB:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPCBPA
      CidrBlock: 10.0.2.0/24
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: VPC BPA Public Subnet B

  VPCBPAPrivateSubnetC:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPCBPA
      CidrBlock: 10.0.3.0/24
      MapPublicIpOnLaunch: false
      Ipv6CidrBlock: !Select [0, !GetAtt VPCBPA.Ipv6CidrBlocks]
      AssignIpv6AddressOnCreation: true
      Tags:
        - Key: Name
          Value: VPC BPA Private Subnet C

  # NAT Gateway
  VPCBPANATGateway:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt VPCBPANATGatewayEIP.AllocationId
      SubnetId: !Ref VPCBPAPublicSubnetB
      Tags:
        - Key: Name
          Value: VPC BPA NAT Gateway

  VPCBPANATGatewayEIP:
    Type: AWS::EC2::EIP
```

```
    Properties:
      Domain: vpc
      Tags:
        - Key: Name
          Value: VPC BPA NAT Gateway EIP

  # Route Tables
  VPCBPAPublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPCBPA
      Tags:
        - Key: Name
          Value: VPC BPA Public Route Table

  VPCBPAPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: VPCBPAInternetGatewayAttachment
    Properties:
      RouteTableId: !Ref VPCBPAPublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref VPCBPAInternetGateway

  VPCBPAPublicSubnetARouteTableAssoc:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref VPCBPAPublicSubnetA
      RouteTableId: !Ref VPCBPAPublicRouteTable

  VPCBPAPublicSubnetBRouteTableAssoc:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref VPCBPAPublicSubnetB
      RouteTableId: !Ref VPCBPAPublicRouteTable

  VPCBPAPrivateRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPCBPA
      Tags:
        - Key: Name
          Value: VPC BPA Private Route Table

  VPCBPAPrivateRoute:
```

```
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref VPCBPAPrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref VPCBPANATGateway

  VPCBPAPrivateSubnetCRoute:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref VPCBPAPrivateRouteTable
      DestinationIpv6CidrBlock: ::/0
      EgressOnlyInternetGatewayId: !Ref VPCBPAEgressOnlyInternetGateway

  VPCBPAPrivateSubnetCRouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref VPCBPAPrivateSubnetC
      RouteTableId: !Ref VPCBPAPrivateRouteTable

  # EC2 Instances Security Group
  VPCBPAInstancesSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: VPC BPA Instances Security Group
      GroupDescription: Allow SSH and ICMP access
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
        - IpProtocol: icmp
          FromPort: -1
          ToPort: -1
          CidrIp: 0.0.0.0/0
      VpcId: !Ref VPCBPA
      Tags:
        - Key: Name
          Value: VPC BPA Instances Security Group

  # EC2 Instances
  VPCBPAInstanceA:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref InstanceAMI
```

```
      InstanceType: t2.micro
      KeyName: !Ref VPCBPAKeyPair
      SubnetId: !Ref VPCBPAPublicSubnetA
      SecurityGroupIds:
        - !Ref VPCBPAInstancesSecurityGroup
      Tags:
        - Key: Name
          Value: VPC BPA Instance A

  VPCBPAInstanceB:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref InstanceAMI
      InstanceType: !Ref InstanceType
      KeyName: !Ref VPCBPAKeyPair
      SubnetId: !Ref VPCBPAPublicSubnetB
      SecurityGroupIds:
        - !Ref VPCBPAInstancesSecurityGroup
      Tags:
        - Key: Name
          Value: VPC BPA Instance B

  VPCBPAInstanceC:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref InstanceAMI
      InstanceType: !Ref InstanceType
      KeyName: !Ref VPCBPAKeyPair
      SubnetId: !Ref VPCBPAPrivateSubnetC
      SecurityGroupIds:
        - !Ref VPCBPAInstancesSecurityGroup
      Tags:
        - Key: Name
          Value: VPC BPA Instance C

  VPCBPAInstanceD:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref InstanceAMI
      InstanceType: !Ref InstanceType
      KeyName: !Ref VPCBPAKeyPair
      NetworkInterfaces:
        - DeviceIndex: '0'
          GroupSet:
```

```
            - !Ref VPCBPAInstancesSecurityGroup
          SubnetId: !Ref VPCBPAPrivateSubnetC
          Ipv6AddressCount: 1
      Tags:
        - Key: Name
          Value: VPC BPA Instance D

  # Flow Logs IAM Role
  VPCBPAFlowLogRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: vpc-flow-logs.amazonaws.com
            Action: 'sts:AssumeRole'
      Tags:
        - Key: Name
          Value: VPC BPA Flow Logs Role

  VPCBPAFlowLogPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: VPC-BPA-FlowLogsPolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action:
              - 'logs:CreateLogGroup'
              - 'logs:CreateLogStream'
              - 'logs:PutLogEvents'
              - 'logs:DescribeLogGroups'
              - 'logs:DescribeLogStreams'
            Resource: '*'
      Roles:
        - !Ref VPCBPAFlowLogRole

  # Flow Logs
  VPCBPAFlowLog:
    Type: AWS::EC2::FlowLog
    Properties:
```

```
      ResourceId: !Ref VPCBPA
      ResourceType: VPC
      TrafficType: ALL
      LogDestinationType: cloud-watch-logs
      LogGroupName: /aws/vpc-flow-logs/VPC-BPA
      DeliverLogsPermissionArn: !GetAtt VPCBPAFlowLogRole.Arn
      LogFormat: '${version} ${account-id} ${interface-id} ${srcaddr} ${dstaddr}
 ${srcport} ${dstport} ${protocol} ${packets} ${bytes} ${start} ${end} ${action} ${log-
status} ${vpc-id} ${subnet-id} ${instance-id} ${tcp-flags} ${type} ${pkt-srcaddr}
 ${pkt-dstaddr} ${region} ${az-id} ${sublocation-type} ${sublocation-id} ${pkt-src-aws-
service} ${pkt-dst-aws-service} ${flow-direction} ${traffic-path} ${reject-reason}'
      Tags:
        - Key: Name
          Value: VPC BPA Flow Logs

  # EC2 Instance Connect Endpoint
  VPCBPAEC2InstanceConnectEndpoint:
    Type: AWS::EC2::InstanceConnectEndpoint
    Properties:
      SecurityGroupIds:
        - !Ref VPCBPAInstancesSecurityGroup
      SubnetId: !Ref VPCBPAPublicSubnetB

Outputs:
  VPCBPAVPCId:
    Description: A reference to the created VPC
    Value: !Ref VPCBPA
    Export:
      Name: vpc-id

  VPCBPAPublicSubnetAId:
    Description: The ID of the public subnet A
    Value: !Ref VPCBPAPublicSubnetA

  VPCBPAPublicSubnetAName:
    Description: The name of the public subnet A
    Value: VPC BPA Public Subnet A

  VPCBPAPublicSubnetBId:
    Description: The ID of the public subnet B
    Value: !Ref VPCBPAPublicSubnetB

  VPCBPAPublicSubnetBName:
    Description: The name of the public subnet B
```

```
    Value: VPC BPA Public Subnet B

  VPCBPAPrivateSubnetCId:
    Description: The ID of the private subnet C
    Value: !Ref VPCBPAPrivateSubnetC

  VPCBPAPrivateSubnetCName:
    Description: The name of the private subnet C
    Value: VPC BPA Private Subnet C

  VPCBPAInstanceAId:
    Description: The ID of instance A
    Value: !Ref VPCBPAInstanceA

  VPCBPAInstanceBId:
    Description: The ID of instance B
    Value: !Ref VPCBPAInstanceB

  VPCBPAInstanceCId:
    Description: The ID of instance C
    Value: !Ref VPCBPAInstanceC

  VPCBPAInstanceDId:
    Description: The ID of instance D
    Value: !Ref VPCBPAInstanceD
```

AWS Management Console

1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Choose **Create stack** and upload the .yaml template file.
3. Go through the steps to launch the template. You'll need to enter an image ID and an instance type (like t2.micro). You'll also need to allow CloudFormation to create an IAM role for you for the flow log creation and permission to log to Amazon CloudWatch.
4. Once you launch the stack, view the **Events** tab to view progress and ensure that the stack completes before you continue.

AWS CLI

1. Run the following command to create the CloudFormation stack:

```
aws cloudformation create-stack --stack-name VPC-BPA-stack --template-body
 file://sampletemplate.yaml --capabilities CAPABILITY_IAM --region us-east-2
```

Output:

```
{
    "StackId": "arn:aws:cloudformation:us-east-2:470889052923:stack/VPC-BPA-
stack/8a7a2cc0-8001-11ef-b196-06386a84b72f"
}
```

2. View the progress and ensure that the stack completes before you continue:

```
aws cloudformation describe-stack-events --stack-name VPC-BPA-stack --region us-
east-2
```

View the impact of VPC BPA with Network Access Analyzer

In this section, you'll use Network Access Analyzer to view the resources in your account that use the internet gateway. Use this analysis to understand the impact of turning on VPC BPA in your account and blocking traffic.

For information about the regional availability of Network Access Analyzer, see Limitations in the *Network Access Analyzer Guide*.

AWS Management Console

1. Open the AWS Network Insights console at https://console.aws.amazon.com/networkinsights/.
2. Choose **Network Access Analyzer**.
3. Choose **Create Network Access Scope**.
4. Choose **Assess impact of VPC Block Public Access** and choose **Next**.
5. The template is already configured to analyze traffic to and from the internet gateways in your account. You can view this under **Source** and **Destination**.
6. Choose **Next**.
7. Choose **Create Network Access Scope**.
8. Choose the scope you just created and choose **Analyze**.

9. Wait for the analysis to complete.
10. View the findings of the analysis. Each row under **Findings** shows a network path that a packet can take in a network to or from an internet gateway in your account. In this case, if you turn on VPC BPA and none of the VPCs and or subnets that appear in these findings are configured as VPC BPA exclusions, traffic to those VPCs and subnets will be restricted.
11. Analyze each finding to understand the impact of VPC BPA on resources in your VPCs.

The impact analysis is complete.

AWS CLI

1. Create a network access scope:

```
aws ec2 create-network-insights-access-scope --match-paths
 "Source={ResourceStatement={ResourceTypes=["AWS::EC2::InternetGateway"]}}"
 "Destination={ResourceStatement={ResourceTypes=["AWS::EC2::InternetGateway"]}}"
 --region us-east-2
```

Output:

```
{
  "NetworkInsightsAccessScope": {
    "NetworkInsightsAccessScopeId": "nis-04cad3c4b3a1d5e3e",
    "NetworkInsightsAccessScopeArn": "arn:aws:ec2:us-
east-2:470889052923:network-insights-access-scope/nis-04cad3c4b3a1d5e3e",
    "CreatedDate": "2024-09-30T15:55:53.171000+00:00",
    "UpdatedDate": "2024-09-30T15:55:53.171000+00:00"
  },
  "NetworkInsightsAccessScopeContent": {
    "NetworkInsightsAccessScopeId": "nis-04cad3c4b3a1d5e3e",
    "MatchPaths": [
      {
        "Source": {
          "ResourceStatement": {
            "ResourceTypes": [
              "AWS::EC2::InternetGateway"
            ]
          }
        }
      },
      {
```

```
          "Destination": {
            "ResourceStatement": {
              "ResourceTypes": [
                "AWS::EC2::InternetGateway"
              ]
            }
          }
        }
      ]
    }
}
```

2. Start the scope analysis:

```
aws ec2 start-network-insights-access-scope-analysis --network-insights-access-
scope-id nis-04cad3c4b3a1d5e3e --region us-east-2
```

Output:


```
{
  "NetworkInsightsAccessScopeAnalysis": {
    "NetworkInsightsAccessScopeAnalysisId": "nisa-0aa383a1938f94cd1",
    "NetworkInsightsAccessScopeAnalysisArn": "arn:aws:ec2:us-
east-2:470889052923:network-insights-access-scope-analysis/
nisa-0aa383a1938f94cd",
    "NetworkInsightsAccessScopeId": "nis-04cad3c4b3a1d5e3e",
    "Status": "running",
    "StartDate": "2024-09-30T15:56:59.109000+00:00",
    "AnalyzedEniCount": 0
  }
}
```


3. Get the results of the analysis:

```
aws ec2 get-network-insights-access-scope-analysis-findings --network-insights-
access-scope-analysis-id nisa-0aa383a1938f94cd1 --region us-east-2 --max-items 1
```

Output:

```
{
  "AnalysisFindings": [
    {
```

```
      "NetworkInsightsAccessScopeAnalysisId": "nisa-0aa383a1938f94cd1",
      "NetworkInsightsAccessScopeId": "nis-04cad3c4b3a1d5e3e",
      "FindingId": "AnalysisFinding-1",
      "FindingComponents": [
        {
          "SequenceNumber": 1,
          "Component": {
            "Id": "igw-04a5344b4e30486f1",
            "Arn": "arn:aws:ec2:us-east-2:470889052923:internet-gateway/
igw-04a5344b4e30486f1",
            "Name": "VPC BPA Internet Gateway"
          },
          "OutboundHeader": {
            "DestinationAddresses": [
              "10.0.1.85/32"
            ]
          },
          "InboundHeader": {
            "DestinationAddresses": [
              "10.0.1.85/32"
            ],
            "DestinationPortRanges": [
              {
                "From": 22,
                "To": 22
              }
            ],
            "Protocol": "6",
            "SourceAddresses": [
              "0.0.0.0/5",
              "100.0.0.0/10",
              "96.0.0.0/6"
            ],
            "SourcePortRanges": [
              {
                "From": 0,
                "To": 65535
              }
            ]
          },
          "Vpc": {
            "Id": "vpc-0762547ec48b6888d",
            "Arn": "arn:aws:ec2:us-east-2:470889052923:vpc/
vpc-0762547ec48b6888d",
```

```
            "Name": "VPC BPA"
          }
        },
        {
          "SequenceNumber": 2,
          "AclRule": {
            "Cidr": "0.0.0.0/0",
            "Egress": false,
            "Protocol": "all",
            "RuleAction": "allow",
            "RuleNumber": 100
          },
          "Component": {
            "Id": "acl-06194fc3a4a03040b",
            "Arn": "arn:aws:ec2:us-east-2:470889052923:network-acl/
acl-06194fc3a4a03040b"
          }
        },
        {
          "SequenceNumber": 3,
          "Component": {
            "Id": "sg-093dde06415d03924",
            "Arn": "arn:aws:ec2:us-east-2:470889052923:security-group/
sg-093dde06415d03924",
            "Name": "VPC BPA Instances Security Group"
          },
          "SecurityGroupRule": {
            "Cidr": "0.0.0.0/0",
            "Direction": "ingress",
            "PortRange": {
              "From": 22,
              "To": 22
            },
            "Protocol": "tcp"
          }
        },
        {
          "SequenceNumber": 4,
          "AttachedTo": {
            "Id": "i-058db34f9a0997895",
            "Arn": "arn:aws:ec2:us-east-2:470889052923:instance/
i-058db34f9a0997895",
            "Name": "VPC BPA Instance A"
          },
```

```
          "Component": {
            "Id": "eni-0fa23f2766f03b286",
            "Arn": "arn:aws:ec2:us-east-2:470889052923:network-interface/
eni-0fa23f2766f03b286"
          },
          "InboundHeader": {
            "DestinationAddresses": [
              "10.0.1.85/32"
            ],
            "DestinationPortRanges": [
              {
                "From": 22,
                "To": 22
              }
            ],
            "Protocol": "6",
            "SourceAddresses": [
              "0.0.0.0/5",
              "100.0.0.0/10",
              "96.0.0.0/6"
            ],
            "SourcePortRanges": [
              {
                "From": 0,
                "To": 65535
              }
            ]
          },
          "Subnet": {
            "Id": "subnet-035d235a762eeed04",
            "Arn": "arn:aws:ec2:us-east-2:470889052923:subnet/
subnet-035d235a762eeed04",
            "Name": "VPC BPA Public Subnet A"
          },
          "Vpc": {
            "Id": "vpc-0762547ec48b6888d",
            "Arn": "arn:aws:ec2:us-east-2:470889052923:vpc/
vpc-0762547ec48b6888d",
            "Name": "VPC BPA"
          }
        }
      ]
    }
  ],
```

```
  "AnalysisStatus": "succeeded",
  "NetworkInsightsAccessScopeAnalysisId": "nisa-0aa383a1938f94cd1",
  "NextToken":
 "eyJOZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ=="
}
```

The results show the traffic to and from the internet gateways in all the VPCs in your account. The results are organized as "findings". "FindingId": "AnalysisFinding-1" indicates that this is the first finding in the analysis. Note that there are multiple findings and each indicates a traffic flow that will be impacted by turning on VPC BPA. The first finding will show that traffic started at an internet gateway ("SequenceNumber": 1), passed to an NACL ("SequenceNumber": 2) to a security group ("SequenceNumber": 3) and ended at an instance ("SequenceNumber": 4).

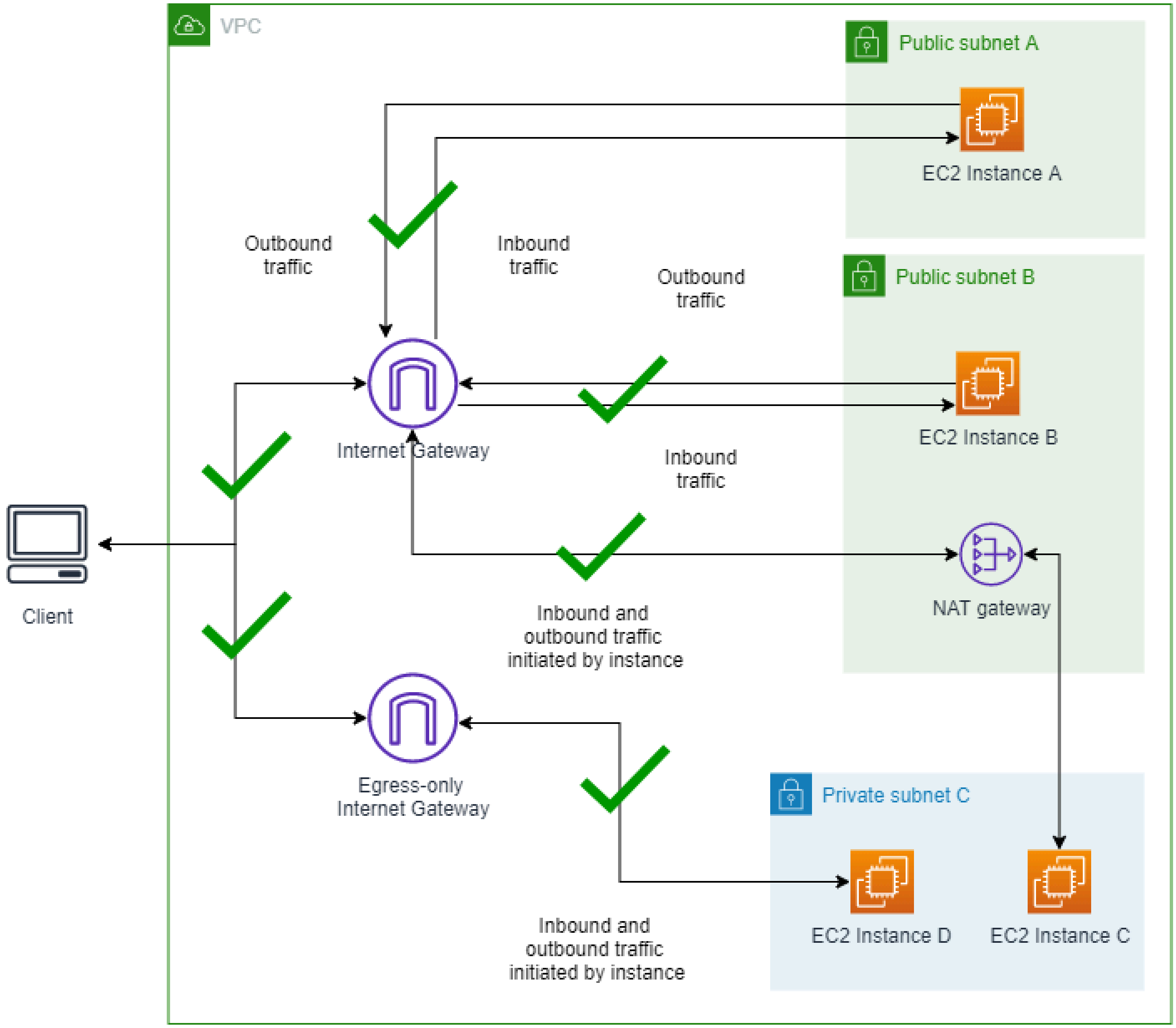
4. Analyze the findings to understand the impact of VPC BPA on resources in your VPCs.

The impact analysis is complete.

Scenario 1 - Connect to instances without VPC BPA turned on

In this section, EC2 instances in public subnets A and B are reachable from the internet via the Internet Gateway, which allows both inbound and outbound traffic. Instances C and D in the private subnet can initiate outbound traffic through the NAT Gateway or Egress-Only Internet Gateway, but are not directly reachable from the internet. This setup provides internet access to some resources while protecting others. The purpose of this setup is to set a baseline and ensure that, before you enable VPC BPA, all instances can be reached, you'll connect to all instances and ping a public IP address.

Diagram of a VPC without VPC BPA turned on:



1.1 Connect to instances

Complete this section to connect to your instances with VPC BPA turned off to ensure you can connect without issue. All of the instances created with the CloudFormation for this example have names like, "VPC BPA Instance A".

AWS Management Console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Open the Instance A details.

3. Connect to instance A using the **EC2 Instance Connect** > **Connect using EC2 Instance Connect Endpoint** option.

4. Choose **Connect**. Once you successfully connect to the instance, ping www.amazon.com to verify you can send outbound requests to the internet.

5. Use the same method you used to connect to instance A to connect to instances B, C, and D. From each instance, ping www.amazon.com to verify you can send outbound requests to the internet.

AWS CLI

1. Ping Instance A using the public IPv4 address to check inbound traffic:

```
ping 18.225.8.244
```

Output:

```
Pinging 18.225.8.244 with 32 bytes of data:

Reply from 18.225.8.244: bytes=32 time=51ms TTL=110
Reply from 18.225.8.244: bytes=32 time=61ms TTL=110
```

Note that the ping is successful and traffic is not blocked.

2. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-058db34f9a0997895 --region us-east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available. Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #_   ~_  ####_        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~._.   _/
/ /
```

```
/m/'
Last login: Fri Sep 27 18:27:57 2024 from 3.16.146.5
[ec2-user@ip-10-0-1-85 ~]$ ping www.amazon.com
PING www-amazon-com.customer.fastly.net (18.65.233.187) 56(84) bytes of data.
64 bytes from 18.65.233.187 (18.65.233.187): icmp_seq=15 ttl=58 time=2.06 ms
64 bytes from 18.65.233.187 (18.65.233.187): icmp_seq=16 ttl=58 time=2.26 ms
```

Note that the ping is successful and traffic is not blocked.

3. Ping Instance B using the public IPv4 address to check inbound traffic:

```
ping 3.18.106.198
```

Output:

```
Pinging 3.18.106.198 with 32 bytes of data:
Reply from 3.18.106.198: bytes=32 time=83ms TTL=110
Reply from 3.18.106.198: bytes=32 time=54ms TTL=110
```

Note that the ping is successful and traffic is not blocked.

4. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id  i-08552a0774b5c8f72 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.
Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #   ~_  ####        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~..   _/
/ /
/m/'
Last login: Fri Sep 27 18:12:27 2024 from 3.16.146.5
[ec2-user@ip-10-0-2-98 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

```
64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=1 ttl=249 time=1.55 ms
64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=2 ttl=249 time=1.67 ms
```

Note that the ping is successful and traffic is not blocked.

5. Connect to Instance C. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-04eca55f2a482b2c4 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.
Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #   ~_  ####        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~..   _/
/ /
/m/'
Last login: Thu Sep 19 20:31:26 2024 from 10.0.2.86
[ec2-user@ip-10-0-3-180 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=1 ttl=248 time=1.75 ms
64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=2 ttl=248 time=1.97 ms
64 bytes from server-3-160-24-26.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=3 ttl=248 time=1.08 ms
```

Note that the ping is successful and traffic is not blocked.

6. Connect to Instance D. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-05f9e6a9cfac1dba0 --region us-east-2 --connection-type eice
```

Output:

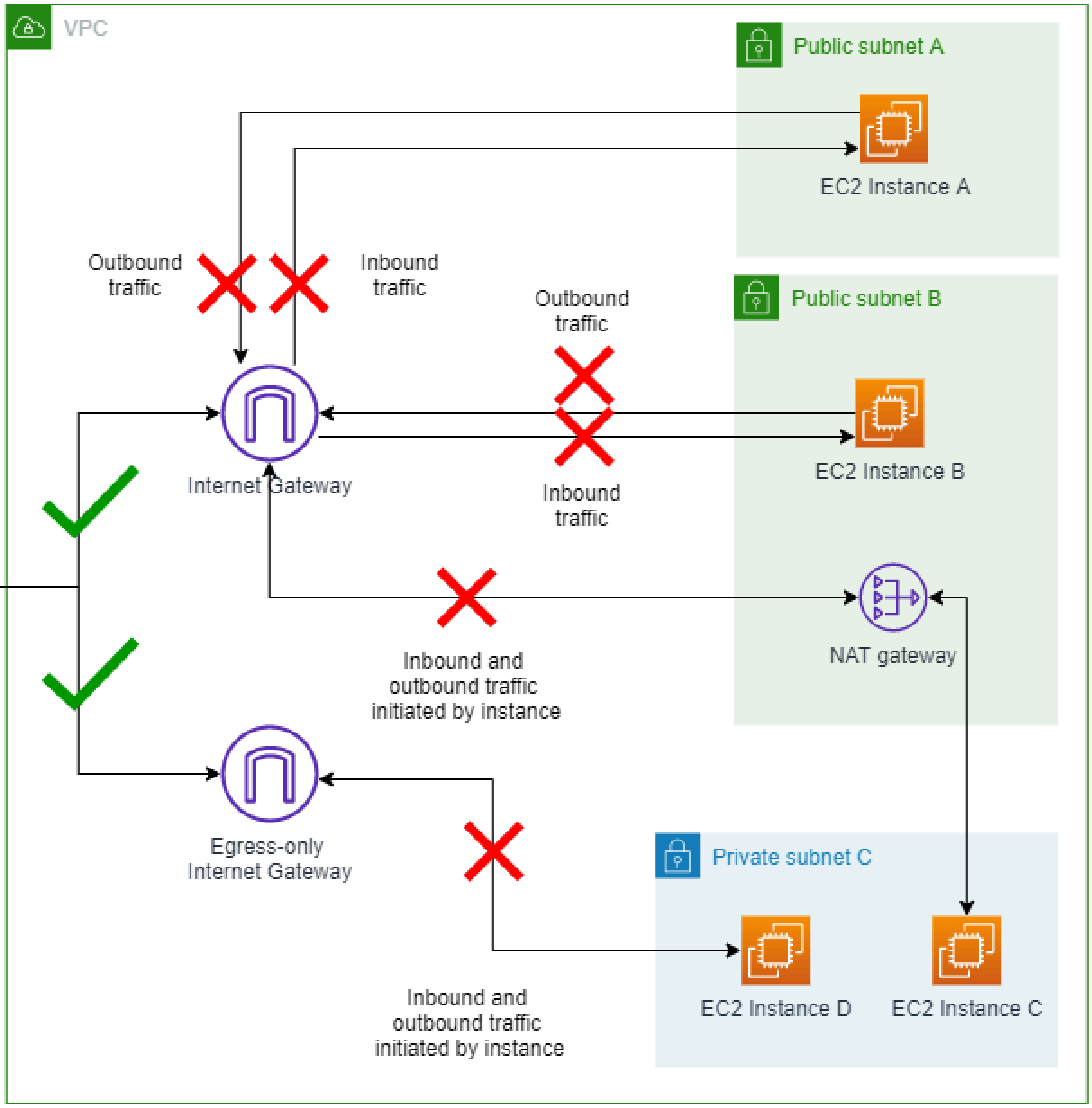
```
The authenticity of host '10.0.3.59 can't be established.
ECDSA key fingerprint is SHA256:c4naBCqbC61/cExDyccEproNU+1HHSpMSzl2J6cOtIZA8g.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.3.59' (ECDSA) to the list of known hosts.
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #   ~_  ####        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~..   _/
_/ _/
_/m/'
[ec2-user@ip-10-0-3-59 ~]$ ping www.amazon.com
PING www.amazon.com(2600:9000:25f3:ee00:7:49a5:5fd4:b121
 (2600:9000:25f3:ee00:7:49a5:5fd4:b121)) 56 data bytes
64 bytes from 2600:9000:25f3:ee00:7:49a5:5fd4:b121
 (2600:9000:25f3:ee00:7:49a5:5fd4:b121): icmp_seq=1 ttl=58 time=1.19 ms
64 bytes from 2600:9000:25f3:ee00:7:49a5:5fd4:b121
 (2600:9000:25f3:ee00:7:49a5:5fd4:b121): icmp_seq=2 ttl=58 time=1.38 ms
```

Note that the ping is successful and traffic is not blocked.

Scenario 2 - Turn on VPC BPA in Bidirectional mode

In this section you'll turn on VPC BPA and block traffic to and from the internet gateways in your account.

Diagram showing VPC BPA Bidirectional mode turned on:



2.1 Enable VPC BPA bidirectional mode

Complete this section to enable VPC BPA. VPC BPA bidirectional mode blocks all traffic to and from internet gateways and egress-only internet gateways in this Region (except for excluded VPCs and subnets).

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Settings**.
3. Choose **Edit public access settings**.

4. Choose **Turn on block public access** and **Bidirectional**, then choose **Save changes**.
5. Wait for the **Status** to change to **On**. It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

VPC BPA is now on.

AWS CLI

1. Use the modify-vpc-block-public-access-options command to turn on VPC BPA:

   ```
   aws ec2 --region us-east-2 modify-vpc-block-public-access-options --internet-gateway-block-mode block-bidirectional
   ```

   It may take a few minutes for VPC BPA settings to take effect and the status to be updated.
2. View the status of VPC BPA:

   ```
   aws ec2 --region us-east-2 describe-vpc-block-public-access-options
   ```

2.2 Connect to instances

Complete this section to connect to your instances.

AWS Management Console

1. Ping the public IPv4 address of Instance A and Instance B as you did in Scenario 1. Note that traffic is blocked.
2. Connect to instance A using the **EC2 Instance Connect** > **Connect using EC2 Instance Connect Endpoint** option as you did in Scenario 1. Make sure you use the endpoint option.
3. Choose **Connect**. Once you successfully connect to the instance, ping www.amazon.com. Note that all outbound traffic is blocked.
4. Use the same method you used to connect to instance A to connect to instances B, C, and D, test outbound requests to the internet. Note that all outbound traffic is blocked.

AWS CLI

1. Ping Instance A using the public IPv4 address to check inbound traffic:

```
ping 18.225.8.244
```

Output:

```
Pinging 18.225.8.244 with 32 bytes of data:

Request timed out.
```

Note that the ping fails and traffic is blocked.

2. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-058db34f9a0997895 --region us-east-2 --connection-type eice
```

Output:

```
The authenticity of host '10.0.1.85' can't be established.
ECDSA key fingerprint is SHA256:3zo/gSss+HAZ+7eTyWlOB/Ke04IM+hadjsoLJeRTWBk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.1.85' (ECDSA) to the list of known hosts.
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #_   ~_  ####_        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~._.   _/
/ /
/m/'
Last login: Fri Sep 27 14:16:53 2024 from 3.16.146.5
[ec2-user@ip-10-0-1-85 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

3. Ping Instance B using the public IPv4 address to check inbound traffic:

```
ping 3.18.106.198
```

Output:

```
Pinging 3.18.106.198 with 32 bytes of data:
Request timed out.
```

Note that the ping fails and traffic is blocked.

4. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id  i-08552a0774b5c8f72 --region us-
east-2 --connection-type eice
```

Output:

```
The authenticity of host '10.0.2.98' can't be established.
ECDSA key fingerprint is SHA256:0IjXKKyVlDthcCfI0IPIJMUiItAOLYKRNLGTYURnFXo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.98' (ECDSA) to the list of known hosts.
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #   ~_  ####        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~..   _/
/ /
/m/'
Last login: Fri Sep 27 14:18:16 2024 from 3.16.146.5
[ec2-user@ip-10-0-2-98 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

5. Connect to Instance C. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-04eca55f2a482b2c4 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #   ~_  ####        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~..   _/
/ /
/m/'
Last login: Tue Sep 24 15:17:56 2024 from 10.0.2.86
[ec2-user@ip-10-0-3-180 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

6. Connect to Instance D. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-05f9e6a9cfac1dba0 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #   ~_  ####        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~..   _/
_/ _/
_/m/'
Last login: Fri Sep 27 16:42:01 2024 from 3.16.146.5
[ec2-user@ip-10-0-3-59 ~]$ ping www.amazon.com
PING www.amazon.com(2600:9000:25f3:8200:7:49a5:5fd4:b121
 (2600:9000:25f3:8200:7:49a5:5fd4:b121)) 56 data bytes
```

Note that the ping fails and traffic is blocked.

2.3 Optional: Verify connectivity is blocked with Reachability Analyzer

VPC Reachability Analyzer can be used to understand whether or not certain network paths can be reached given your network configuration, including VPC BPA settings. In this example you will analyze the same network path that was attempted earlier to confirm that VPC BPA is the reason why connectivity is failing.

AWS Management Console

1. Go to the Network Insights console at https://console.aws.amazon.com/networkinsights/home#ReachabilityAnalyzer.
2. Click **Create and analyze path**.
3. For the **Source Type**, choose **Internet Gateways** and select the internet gateway tagged **VPC BPA Internet Gateway** from the **Source** dropdown.
4. For the **Destination Type**, choose **Instances** and select the instance tagged with **VPC BPA Instance A** from the **Destination** dropdown.
5. Click **Create and analyze path**.
6. Wait for the analysis to complete. It could take a few minutes.
7. Once complete, you should see that the **Reachability Status**is **Not reachable** and that the **Path details** shows that `VPC_BLOCK_PUBLIC_ACCESS_ENABLED` is the cause.

AWS CLI

1. Create a network path using the ID of the internet gateway tagged VPC BPA Internet Gateway and the ID of the instance tagged VPC BPA Instance A:

   ```
   aws ec2 --region us-east-2 create-network-insights-path --source igw-id --
   destination instance-id --protocol TCP
   ```

2. Start an analysis on the network path:

   ```
   aws ec2 --region us-east-2 start-network-insights-analysis --network-insights-
   path-id nip-id
   ```

3. Retrieve the results of the analysis:

   ```
   aws ec2 --region us-east-2 describe-network-insights-analyses --network-
   insights-analysis-ids nia-id
   ```

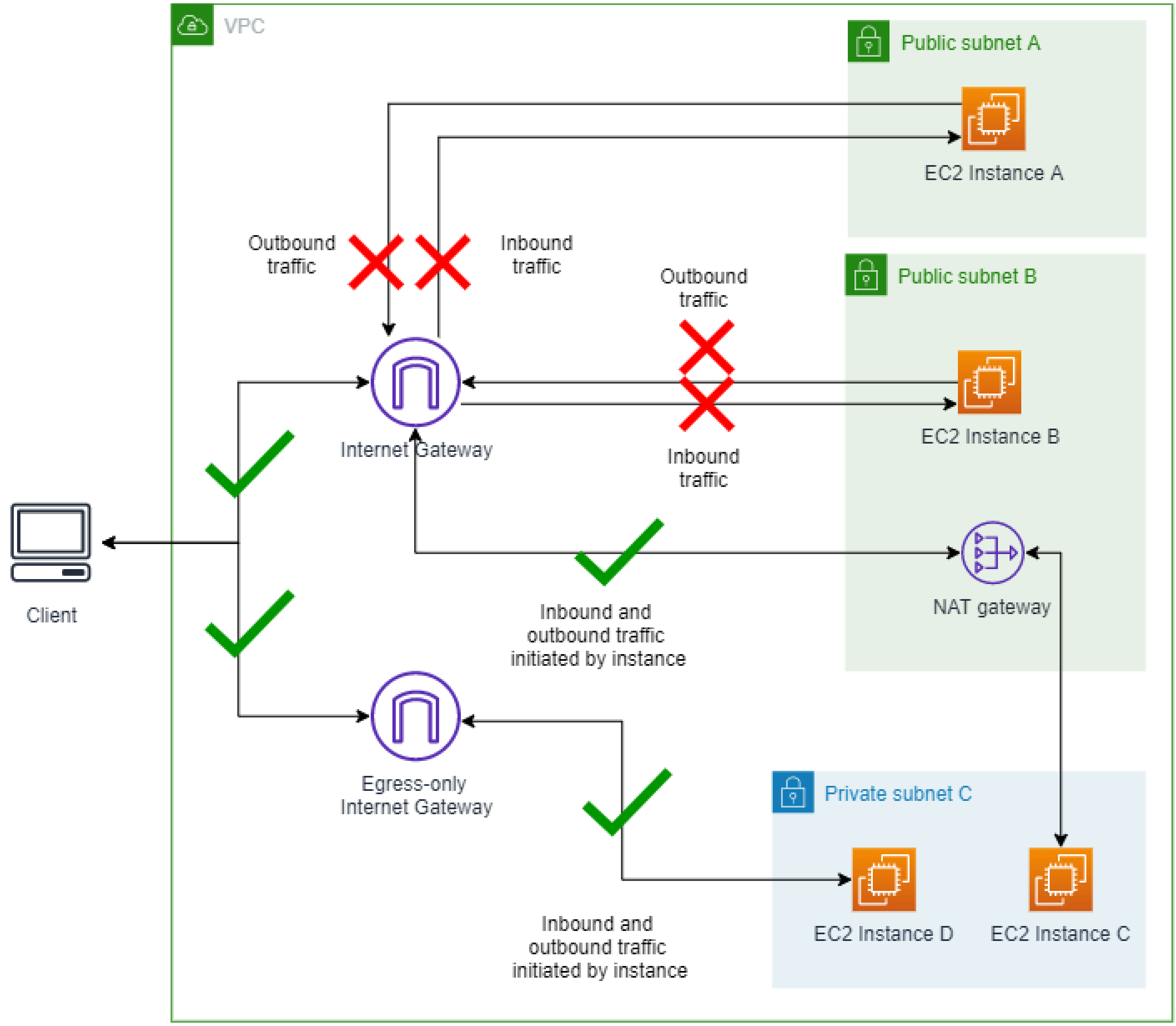
4. Verify that `VPC_BLOCK_PUBLIC_ACCESS_ENABLED` is the `ExplanationCode` for the lack of reachability.

Note that you can also Monitor VPC BPA impact with flow logs.

Scenario 3 - Change VPC BPA to Ingress-only mode

In this section you'll change the VPC BPA traffic direction and allow only traffic that uses a NAT gateway or egress-only internet gateway. EC2 instances A and B in the public subnets will be unreachable from the internet because BPA blocks inbound traffic through the Internet Gateway. Instances C and D in the private subnet will remain able to initiate outbound traffic via the NAT gateway and Egress-Only Internet Gateway, and therefore can still reach the internet.

Diagram of VPC BPA Ingress-only mode turned on:



3.1 Change mode to ingress-only

Complete this section to change the mode.

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Settings**.
3. In the **Block public access** tab, choose **Edit public access settings**.

4. Modify the public access settings in the VPC console and change the direction to **Ingress-only**.
5. Save the changes and wait for the status to be updated. It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

AWS CLI

1. Modify the VPC BPA mode:

```
aws ec2 --region us-east-2 modify-vpc-block-public-access-options --internet-
gateway-block-mode block-ingress
```

   It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

2. View the status of VPC BPA:

```
aws ec2 --region us-east-2 describe-vpc-block-public-access-options
```

3.2 Connect to instances

Complete this section to connect to the instances.

AWS Management Console

1. Ping the public IPv4 address of Instance A and Instance B as you did in Scenario 1. Note that traffic is blocked.
2. Connect to Instance A and B using EC2 instance connect as you did in Scenario 1 and ping www.amazon.com from them. Note that you cannot ping a public site on the internet from Instance A or B and traffic is blocked.
3. Connect to Instance C and D using EC2 instance connect as you did in Scenario 1 and ping www.amazon.com from them. Note that you can ping a public site on the internet from Instance C or D and traffic is allowed.

AWS CLI

1. Ping Instance A using the public IPv4 address to check inbound traffic:

```
ping 18.225.8.244
```

Output:

```
Pinging 18.225.8.244 with 32 bytes of data:

Request timed out.
```

Note that the ping fails and traffic is blocked.

2. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-058db34f9a0997895 --region us-east-2 --connection-type eice
```

Output:

```
The authenticity of host '10.0.1.85' can't be established.
ECDSA key fingerprint is SHA256:3zo/gSss+HAZ+7eTyWlOB/Ke04IM+hadjsoLJeRTWBk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.1.85' (ECDSA) to the list of known hosts.
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #_   ~_  ####_        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~._.   _/
/ /
/m/'
Last login: Fri Sep 27 14:16:53 2024 from 3.16.146.5
[ec2-user@ip-10-0-1-85 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

3. Ping Instance B using the public IPv4 address to check inbound traffic:

```
ping 3.18.106.198
```

Output:

```
Pinging 3.18.106.198 with 32 bytes of data:
Request timed out.
```

Note that the ping fails and traffic is blocked.

4. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-08552a0774b5c8f72 --region us-
east-2 --connection-type eice
```

Output:

```
The authenticity of host '10.0.2.98 ' can't be established.
ECDSA key fingerprint is SHA256:0IjXKKyVlDthcCfI0IPIJMUiItAOLYKRNLGTYURnFXo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.98' (ECDSA) to the list of known hosts.
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #   ~_  ####        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~..   _/
_/ /
/m/'
Last login: Fri Sep 27 14:18:16 2024 from 3.16.146.5
[ec2-user@ip-10-0-2-98 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

5. Connect to Instance C. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-04eca55f2a482b2c4 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'

Last login: Tue Sep 24 15:28:09 2024 from 10.0.2.86

[ec2-user@ip-10-0-3-180 ~]$ ping www.amazon.com

PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.

64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=1 ttl=248 time=1.84 ms
64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=2 ttl=248 time=1.40 ms
```

Note that the ping is successful and traffic is not blocked.

6. Connect to Instance D. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-05f9e6a9cfac1dba0 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
```

```
        _/m/'
Last login: Fri Sep 27 16:48:38 2024 from 3.16.146.5
[ec2-user@ip-10-0-3-59 ~]$ ping www.amazon.com
PING www.amazon.com(2600:9000:25f3:5800:7:49a5:5fd4:b121
 (2600:9000:25f3:5800:7:49a5:5fd4:b121)) 56 data bytes
64 bytes from 2600:9000:25f3:5800:7:49a5:5fd4:b121
 (2600:9000:25f3:5800:7:49a5:5fd4:b121): icmp_seq=14 ttl=58 time=1.47 ms
64 bytes from 2600:9000:25f3:5800:7:49a5:5fd4:b121
 (2600:9000:25f3:5800:7:49a5:5fd4:b121): icmp_seq=16 ttl=58 time=1.59 ms
```

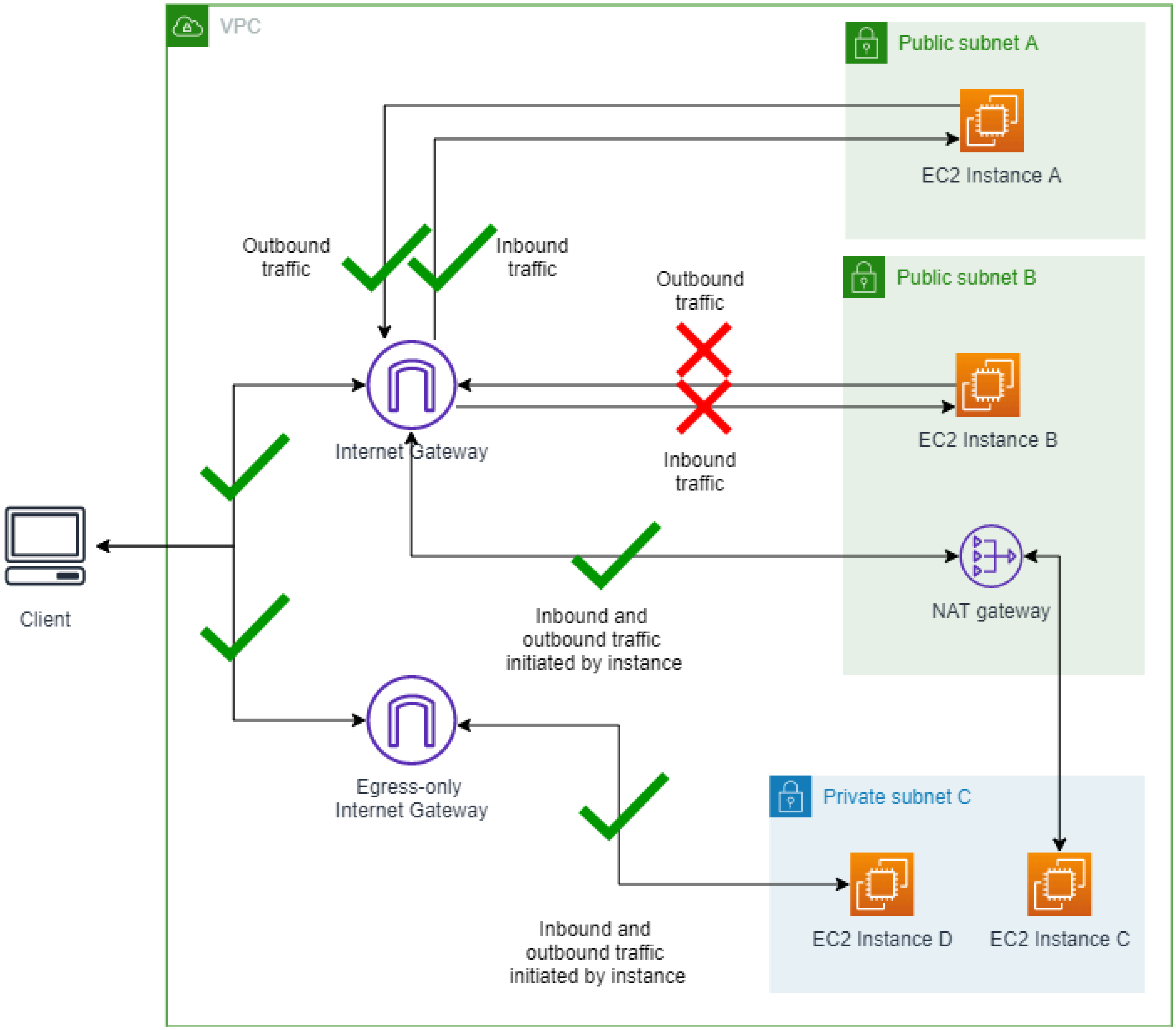
Note that the ping is successful and traffic is not blocked.

Scenario 4 - Create an exclusion

In this section, you'll create an exclusion. VPC BPA will then only block traffic on the subnets *without* an exclusion. A VPC BPA exclusion is a mode that can be applied to a single VPC or subnet that exempts it from the account's VPC BPA mode and will allow bidirectional or egress-only access. You can create VPC BPA exclusions for VPCs and subnets even when VPC BPA is not enabled on the account to ensure that there is no traffic disruption to the exclusions when VPC BPA is turned on.

In this example, we'll create an exclusion for Subnet A to show how traffic to exclusions is impacted by VPC BPA.

Diagram of VPC BPA Ingress-only mode turned on and Subnet A exclusion with Bidirectional mode turned on:



4.1 Create an exclusion for Subnet A

Complete this section to create an exclusion. A VPC BPA exclusion is a mode that can be applied to a single VPC or subnet that exempts it from the account's VPC BPA mode and will allow bidirectional or egress-only access. You can create VPC BPA exclusions for VPCs and subnets even when VPC BPA is not enabled on the account to ensure that there is no traffic disruption to the exclusions when VPC BPA is turned on.

AWS Management Console

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

2. On the left navigation pane, choose **Settings**.
3. In the **Block public access** tab, under **Exclusions**, choose **Create exclusions**.
4. Choose **VPC BPA Public Subnet A**, ensure that allow direction **Bidirectional** is selected, and choose **Create exclusions**.
5. Wait for the **Exclusion status** to change to **Active**. You may need to refresh the exclusion table to see the change.

The exclusion has been created.

AWS CLI

1. Modify the exclusion allow direction:

```
aws ec2 --region us-east-2 create-vpc-block-public-access-exclusion --subnet-
id subnet-id --internet-gateway-exclusion-mode allow-bidirectional
```

2. It can take time for the exclusion status to update. To view the status of the exclusion:

```
aws ec2 --region us-east-2 describe-vpc-block-public-access-exclusions --
exclusion-ids exclusion-id
```

4.2 Connect to instances

Complete this section to connect to the instances.

AWS Management Console

1. Ping the public IPv4 address of Instance A. Note that traffic is allowed.
2. Ping the public IPv4 address of Instance B. Note that traffic is blocked.
3. Connect to Instance A using EC2 instance connect as you did in Scenario 1 and ping www.amazon.com. Note that you can ping a public site on the internet from Instance A. Traffic is allowed.
4. Connect to Instance B using EC2 instance connect as you did in Scenario 1 and ping www.amazon.com from it. Note that you cannot ping a public site on the internet from Instance B. Traffic is blocked.

5. Connect to Instance C and D using EC2 instance connect as you did in Scenario 1 and ping www.amazon.com from them. Note that you can ping a public site on the internet from Instance C or D. Traffic is allowed.

AWS CLI

1. Ping Instance A using the public IPv4 address to check inbound traffic:

```
ping 18.225.8.244
```

Output:

```
Pinging 18.225.8.244 with 32 bytes of data:

Reply from 18.225.8.244: bytes=32 time=51ms TTL=110
Reply from 18.225.8.244: bytes=32 time=61ms TTL=110
```

Note that the ping is successful and traffic is not blocked.

2. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-058db34f9a0997895 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #_   ~_  ####_        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~._.   _/
/ /
/m/'
Last login: Fri Sep 27 17:58:12 2024 from 3.16.146.5
[ec2-user@ip-10-0-1-85 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=1 ttl=249 time=1.03 ms
```

```
64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=2 ttl=249 time=1.72 ms
```

Note that the ping is successful and traffic is not blocked.

3. Ping Instance B using the public IPv4 address to check inbound traffic:

```
ping 3.18.106.198
```

Output:

```
Pinging 3.18.106.198 with 32 bytes of data:
Request timed out.
```

Note that the ping fails and traffic is blocked.

4. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-08552a0774b5c8f72 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #   ~_  ####        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~..   _/
_/ /
/m/'
Last login: Fri Sep 27 18:12:03 2024 from 3.16.146.5
[ec2-user@ip-10-0-2-98 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

5. Connect to Instance C. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-04eca55f2a482b2c4 --region us-
east-2 --connection-type eice
```

Output

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
,     #   ~_  ####        Amazon Linux 2023
~~  _#####\  ~~     ###|
~~       #/ ___   https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '->
~~~         /
~~..   _/
_/ /
/m/'

Last login: Tue Sep 24 15:28:09 2024 from 10.0.2.86

[ec2-user@ip-10-0-3-180 ~]$ ping www.amazon.com

PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.

64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=1 ttl=248 time=1.84 ms
64 bytes from server-3-160-24-126.cmh68.r.cloudfront.net (18.65.233.187):
 icmp_seq=2 ttl=248 time=1.40 ms
```

Note that the ping is successful and traffic is not blocked.

6. Connect to Instance D. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-05f9e6a9cfac1dba0 --region us-
east-2 --connection-type eice
```

Output

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
```

```
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 27 18:00:52 2024 from 3.16.146.5
[ec2-user@ip-10-0-3-59 ~]$ ping www.amazon.com
PING
 www.amazon.com(g2600-141f-4000-059a-0000-0000-0000-3bd4.deploy.static.akamaitechnologie
 (2600:141f:4000:59a::3bd4)) 56 data bytes
64 bytes from
 g2600-141f-4000-059a-0000-0000-0000-3bd4.deploy.static.akamaitechnologies.com
 (2600:141f:4000:59a::3bd4): icmp_seq=1 ttl=48 time=15.9 ms
64 bytes from
 g2600-141f-4000-059a-0000-0000-0000-3bd4.deploy.static.akamaitechnologies.com
 (2600:141f:4000:59a::3bd4): icmp_seq=2 ttl=48 time=15.8 ms
```

Note that the ping is successful and traffic is not blocked.

4.3 Optional: Verify connectivity with Reachability Analyzer

Using the same network path created in Reachability Analyzer in Scenario 2, you can now run a new analysis and confirm that the path is reachable now that an exclusion has been created for Public Subnet A.

For information about the regional availability of Reachability Analyzer, see Considerations in the *Reachability Analyzer Guide*.

AWS Management Console

1. From the Network Path you created earlier in the Network Insights console, click **Re-run analysis**.
2. Wait for the analysis to complete. It may take several minutes.
3. Confirm that the path is now **Reachable**.

AWS CLI

1. Using the network path ID created earlier, start a new analysis:

```
aws ec2 --region us-east-2 start-network-insights-analysis --network-insights-path-id nip-id
```

2. Retrieve the results of the analysis:

```
aws ec2 --region us-east-2 describe-network-insights-analyses --network-insights-analysis-ids nia-id
```

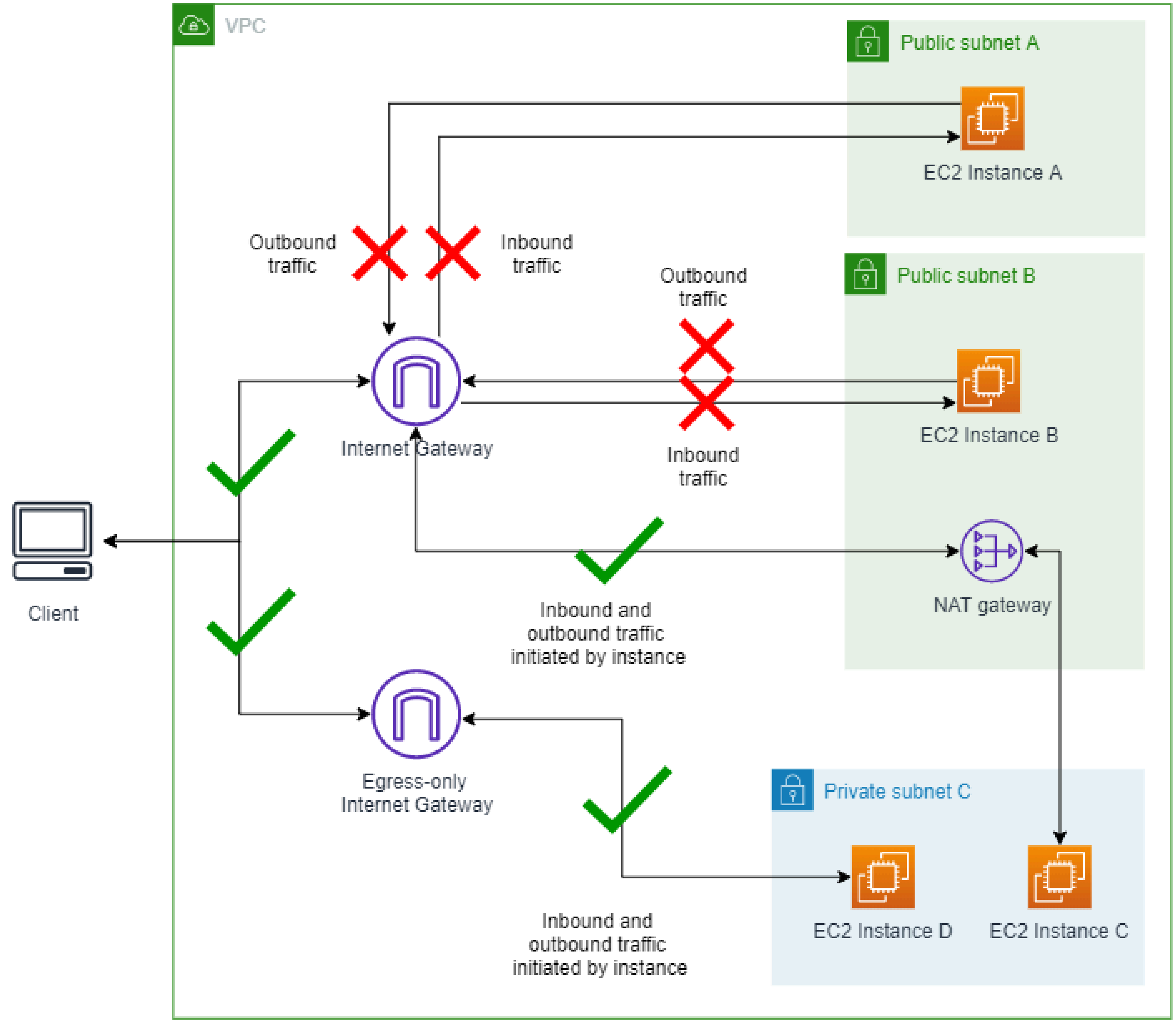
3. Confirm that the `VPC_BLOCK_PUBLIC_ACCESS_ENABLED` explanation code is no longer present.

Scenario 5 - Modify exclusion mode

In this section you'll change the allow traffic direction on the exclusion to see how it impacts VPC BPA.

> **Note**
>
> In this scenario, you'll change the exclusion mode to Egress-only. Note that when you do this, the Egress-only exclusion on Subnet A doesn't allow outbound traffic, which is counterintuitive because you'd expect it to permit outbound traffic. However, since the account-level BPA is Ingress-only, Egress-only exclusions are ignored, and Subnet A's routing to an internet gateway is restricted by VPC BPA, blocking outbound traffic. To enable outbound traffic on Subnet A, you'd have to switch VPC BPA to Bidirectional mode.

Diagram of VPC BPA Ingress-only mode turned on and Subnet A exclusion with egress-only mode turned on:



5.1 Change exclusion allow direction to egress-only

Complete this section to change the exclusion allow direction.

AWS Management Console

1. Edit the exclusion you created in Scenario 4 and change the allow direction to **Egress-only**.
2. Choose **Save changes**.
3. Wait for the **Exclusion** status to change to **Active**. It may take a few minutes for VPC BPA settings to take effect and the status to be updated. You may need to refresh the exclusion table to see the change.

AWS CLI

1. Modify the exclusion allow direction:

```
aws ec2 --region us-east-2 modify-vpc-block-public-access-exclusion --exclusion-
id exclusion-id --internet-gateway-exclusion-mode allow-egress
```

   It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

2. It can take time for the exclusion status to update. To view the status of the exclusion:

```
aws ec2 --region us-east-2 describe-vpc-block-public-access-exclusion
```

5.2 Connect to instances

Complete this section to connect to the instances.

AWS Management Console

1. Ping the public IPv4 address of Instance A and B. Note that traffic is blocked.
2. Connect to Instance A and B using EC2 instance connect as you did in Scenario 1 and ping www.amazon.com. Note that you cannot ping a public site on the internet from Instance A or B. Traffic is blocked.
3. Connect to Instance C and D using EC2 instance connect as you did in Scenario 1 and ping www.amazon.com from them. Note that you can ping a public site on the internet from Instance C or D. Traffic is allowed.

AWS CLI

1. Ping Instance A using the public IPv4 address to check inbound traffic:

```
ping 18.225.8.244
```

   Output:

```
Pinging 18.225.8.244 with 32 bytes of data:
Request timed out.
```

Note that the ping fails and traffic is blocked.

2. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-058db34f9a0997895 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 27 18:09:55 2024 from 3.16.146.5
[ec2-user@ip-10-0-1-85 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

3. Ping Instance B using the public IPv4 address to check inbound traffic:

```
ping 3.18.106.198
```

Output:

```
Pinging 3.18.106.198 with 32 bytes of data:
Request timed out.
```

Note that the ping fails and traffic is blocked.

4. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-058db34f9a0997895 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 27 18:09:55 2024 from 3.16.146.5
[ec2-user@ip-10-0-1-85 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

5. Connect to Instance C. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-04eca55f2a482b2c4 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 27 18:00:31 2024 from 3.16.146.5
[ec2-user@ip-10-0-3-180 ~]$ ping www.amazon.com
PING www.amazon.com(2600:9000:25f3:a600:7:49a5:5fd4:b121
 (2600:9000:25f3:a600:7:49a5:5fd4:b121)) 56 data bytes
```

```
64 bytes from 2600:9000:25f3:a600:7:49a5:5fd4:b121
 (2600:9000:25f3:a600:7:49a5:5fd4:b121): icmp_seq=1 ttl=58 time=1.51 ms
64 bytes from 2600:9000:25f3:a600:7:49a5:5fd4:b121
 (2600:9000:25f3:a600:7:49a5:5fd4:b121): icmp_seq=2 ttl=58 time=1.49 ms
```

Note that the ping is successful and traffic is not blocked.

6. Connect to Instance D. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-05f9e6a9cfac1dba0 --region us-
east-2 --connection-type eice
```

Output:

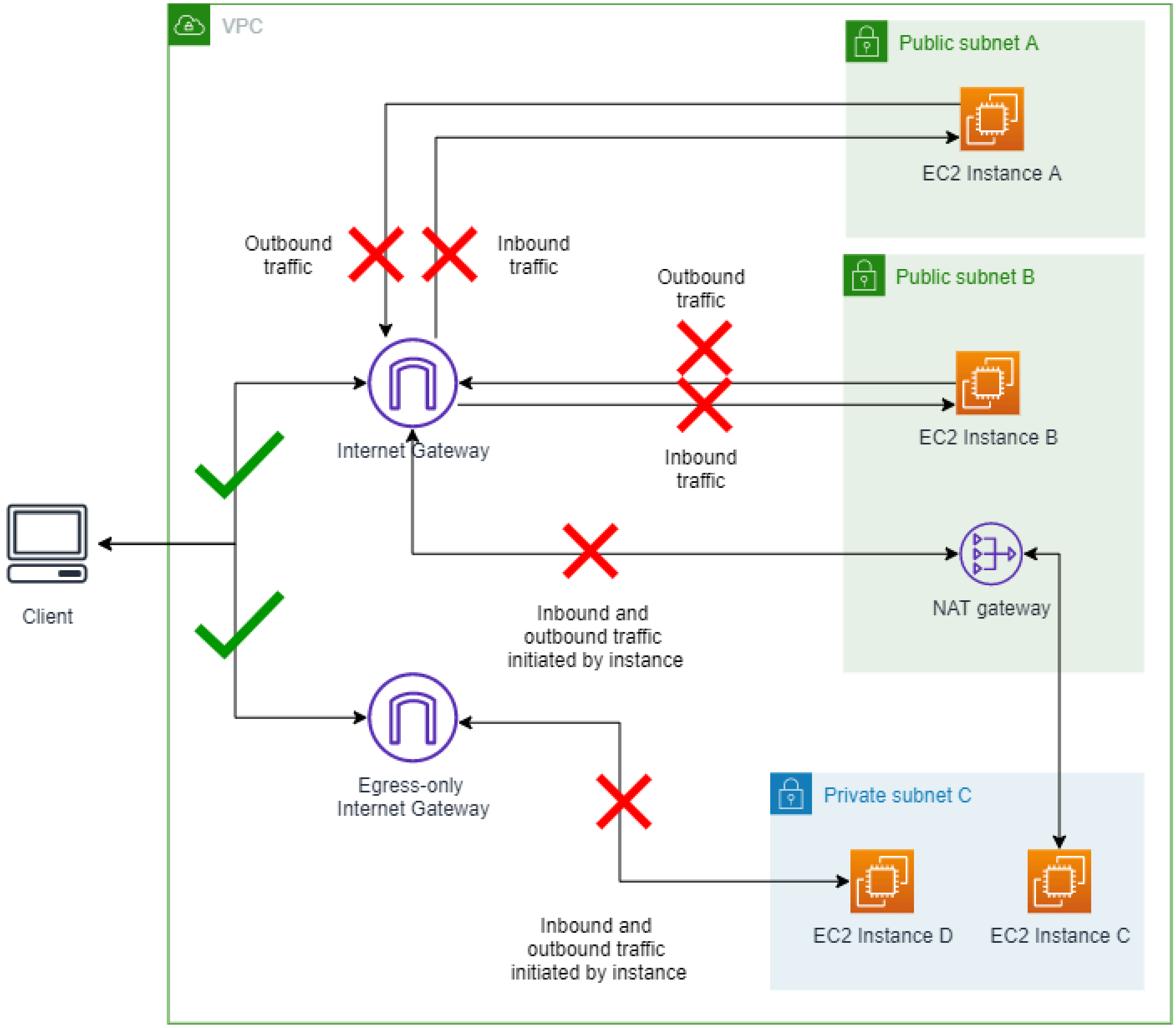
```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 27 18:13:55 2024 from 3.16.146.5
[ec2-user@ip-10-0-3-59 ~]$ ping www.amazon.com
PING www.amazon.com(2606:2cc0::374 (2606:2cc0::374)) 56 data bytes
64 bytes from 2606:2cc0::374 (2606:2cc0::374): icmp_seq=1 ttl=58 time=1.21 ms
64 bytes from 2606:2cc0::374 (2606:2cc0::374): icmp_seq=2 ttl=58 time=1.51 ms
```

Note that the ping is successful and traffic is not blocked.

Scenario 6 - Modify VPC BPA mode

In this section you'll change the VPC BPA block direction to see how it impacts traffic. In this scenario, VPC BPA enabled in bidirectional mode blocks all traffic just like in Scenario 1. Unless an exclusion has access to a NAT gateway or egress-only internet gateway, traffic is blocked.

Diagram of VPC BPA Bidirectional mode turned on and Subnet A exclusion with egress-only mode turned on:



6.1 Change VPC BPA to bidirectional mode

Complete this section to change the VPC BPA mode.

AWS Management Console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the left navigation pane, choose **Settings**.
3. Choose **Edit public access settings**.
4. Change the block direction to **Bidirectional** then choose **Save changes**.

5. Wait for the **Status** to change to **On**. It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

AWS CLI

1. Modify the VPC BPA block direction:

```
aws ec2 --region us-east-2 modify-vpc-block-public-access-options --internet-gateway-block-mode block-bidirectional
```

   It may take a few minutes for VPC BPA settings to take effect and the status to be updated.

2. View the status of VPC BPA:

```
aws ec2 --region us-east-2 describe-vpc-block-public-access-options
```

6.2 Connect to instances

Complete this section to connect to the instances.

AWS Management Console

1. Ping the public IPv4 address of Instance A and B. Note that traffic is blocked.
2. Connect to Instance A and B using EC2 instance connect as you did in Scenario 1 and ping www.amazon.com. Note that you cannot ping a public site on the internet from Instance A or B. Traffic is blocked.
3. Connect to Instance C and D using EC2 instance connect as you did in Scenario 1 and ping www.amazon.com from them. Note that you cannot ping a public site on the internet from Instance C or D. Traffic is blocked.

AWS CLI

1. Ping Instance A using the public IPv4 address to check inbound traffic:

```
ping 18.225.8.244
```

   Output:

```
Pinging 18.225.8.244 with 32 bytes of data:
Request timed out.
```

Note that the ping fails and traffic is blocked.

2. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-058db34f9a0997895 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 27 18:17:44 2024 from 3.16.146.5
[ec2-user@ip-10-0-1-85 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

3. Ping Instance A using the public IPv4 address to check inbound traffic:

```
ping 3.18.106.198
```

Output:

```
Pinging 3.18.106.198 with 32 bytes of data:
Request timed out.
```

Note that the ping fails and traffic is blocked.

4. Use the private IPv4 address to connect and check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-058db34f9a0997895 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 27 18:09:55 2024 from 3.16.146.5
[ec2-user@ip-10-0-1-85 ~]$ ping www.amazon.com
PING d3ag4hukkh62yn.cloudfront.net (18.65.233.187) 56(84) bytes of data.
```

Note that the ping fails and traffic is blocked.

5. Connect to Instance C. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-04eca55f2a482b2c4 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 27 18:19:45 2024 from 3.16.146.5
[ec2-user@ip-10-0-3-180 ~]$ ping www.amazon.com
```

```
PING www.amazon.com(2600:9000:25f3:6200:7:49a5:5fd4:b121
 (2600:9000:25f3:6200:7:49a5:5fd4:b121)) 56 data bytes
```

Note that the ping fails and traffic is blocked.

6. Connect to Instance D. Since there is no public IP address to ping, use EC2 Instance Connect to connect and then ping a public IP from the instance to check outbound traffic:

```
aws ec2-instance-connect ssh --instance-id i-05f9e6a9cfac1dba0 --region us-
east-2 --connection-type eice
```

Output:

```
A newer release of "Amazon Linux" is available.  Version 2023.5.20240916:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 27 18:20:58 2024 from 3.16.146.5
[ec2-user@ip-10-0-3-59 ~]$ ping www.amazon.com
PING www.amazon.com(2600:9000:25f3:b400:7:49a5:5fd4:b121
 (2600:9000:25f3:b400:7:49a5:5fd4:b121)) 56 data bytes
```

Note that the ping fails and traffic is blocked.

Cleanup

In this section you'll delete all of the resources you've created for this advanced example. It's important to cleanup the resources to avoid excess additional charges for resources created in your account.

Delete the CloudFormation resources

Complete this section to delete the resources you created with the AWS CloudFormation template.

AWS Management Console

1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Choose the VPC BPA stack.
3. Choose **Delete**.
4. Once you start deleting the stack, view the **Events** tab to view progress and ensure that the stack is deleted. You may have to force delete the stack for it to be fully deleted.

AWS CLI

1. Delete the CloudFormation stack. You may have to force delete the stack for it to be fully deleted.

```
aws cloudformation delete-stack --stack-name VPC-BPA-stack --region us-east-2
```

2. View the progress and ensure that the stack is deleted.

```
aws cloudformation describe-stack-events --stack-name VPC-BPA-stack --region us-
east-2
```

Track exclusion deletion with AWS CloudTrail

Complete this section to track exclusion deletion with AWS CloudTrail. CloudTrail entries appear when you delete an exclusion.

AWS Management Console

You can view any deleted exclusions in the CloudTrail Event history by looking up **Resource type** > **AWS::EC2::VPCBlockPublicAccessExclusion** in the AWSCloudTrail console at https://console.aws.amazon.com/cloudtrailv2/.

AWS CLI

You can use the lookup-events command to view the events related to deleting exclusions:

```
aws cloudtrail lookup-events --lookup-attributes
 AttributeKey=ResourceType,AttributeValue=AWS::EC2::VPCBlockPublicAccessExclusion
```

The advanced example is complete.

Security best practices for your VPC

The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

- When you add subnets to your VPC to host your application, create them in multiple Availability Zones. An Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. Using multiple Availability Zones makes your production applications highly available, fault tolerant, and scalable.
- Use security groups to control traffic to EC2 instances in your subnets. For more information, see Security groups.
- Use network ACLs to control inbound and outbound traffic at the subnet level. For more information, see Control subnet traffic with network access control lists.
- Manage access to AWS resources in your VPC using AWS Identity and Access Management (IAM) identity federation, users, and roles. For more information, see Identity and access management for Amazon VPC.
- Use VPC Flow Logs to monitor the IP traffic going to and from a VPC, subnet, or network interface. For more information, see VPC Flow Logs.
- Use Network Access Analyzer to identify unintended network access to resources in our VPCs. For more information, see the Network Access Analyzer Guide.
- Use AWS Network Firewall to monitor and protect your VPC by filtering inbound and outbound traffic. For more information, see the AWS Network Firewall Guide.
- Use Amazon GuardDuty to detect potential threats to your accounts, containers, workloads, and data within your AWS environment. The foundational threat detection includes monitoring the VPC flow logs associated with your Amazon EC2 instances. For more information, see VPC Flow Logs in the *Amazon GuardDuty User Guide*.

For answers to frequently asked questions related to VPC security, see *Security and Filtering* in the Amazon VPC FAQs.

Use Amazon VPC with other AWS services

Amazon Virtual Private Cloud (VPC) is a foundational AWS service that provides a secure, customizable networking environment for your cloud infrastructure. Beyond creating and managing your own VPC, you can leverage the integration between VPC and other AWS services to build comprehensive solutions tailored to your specific needs.

You can connect your VPC to various AWS services using AWS PrivateLink. This enables private connectivity between your VPC and supported AWS services or on-premises applications, keeping network traffic within the AWS network and avoiding exposure to the public internet. This is particularly useful for maintaining strict security boundaries and compliance requirements.

To further strengthen the security of your VPC, you can use AWS Network Firewall. This managed firewall service allows you to define and enforce network-level security policies, filtering both north-south and east-west traffic within your VPC. By pairing Network Firewall with your VPC, you can enhance your defense strategy and protect your cloud resources from unauthorized access or malicious activity.

Additionally, you can filter DNS traffic within your VPC using the Route 53 Resolver DNS Firewall. This capability enables you to create custom DNS filtering rules to control which domains your VPC resources can resolve, providing an additional layer of security and compliance enforcement.

If you encounter reachability issues between resources within your VPC or connected to your VPC, you can leverage Reachability Analyzer. Reachability Analyzer performs virtual connectivity tests, providing detailed hop-by-hop path information and identifying any blocking components. This troubleshooting tool can quickly help you identify and resolve network connectivity problems.

By integrating these complementary AWS services with your VPC, you can build powerful, secure, and resilient cloud solutions that address your unique business and architectural requirements.

Contents

- Connect your VPC to services using AWS PrivateLink
- Filter network traffic using AWS Network Firewall
- Filter DNS traffic using Route 53 Resolver DNS Firewall
- Troubleshoot reachability issues using Reachability Analyzer

Connect your VPC to services using AWS PrivateLink

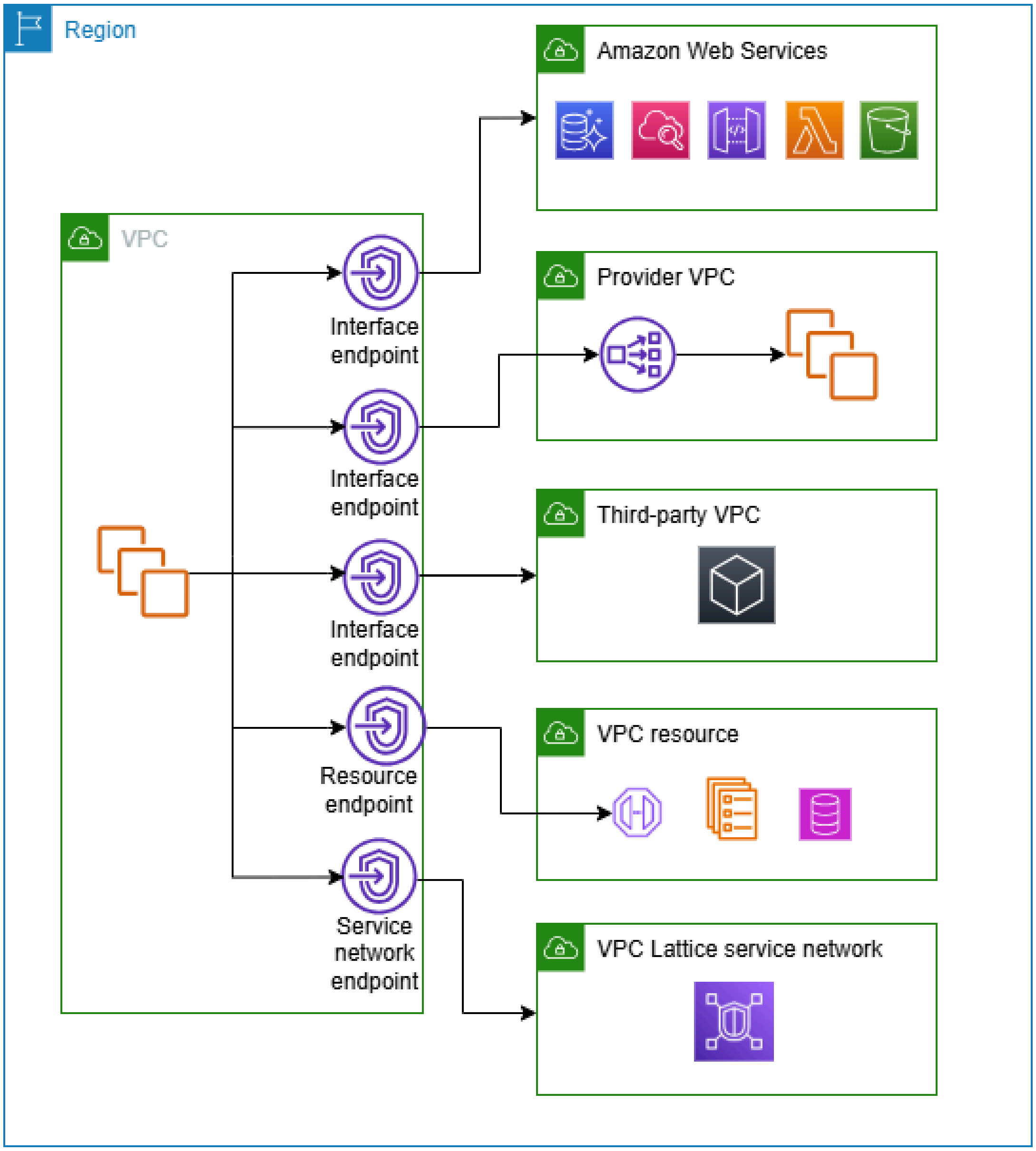
AWS PrivateLink establishes private connectivity between virtual private clouds (VPC) and supported AWS services, services hosted by other AWS accounts, supported AWS Marketplace services, and supported resources. You do not need to use an internet gateway, NAT device, AWS Direct Connect connection, or AWS Site-to-Site VPN connection to communicate with the service or resource.

To use AWS PrivateLink, create a VPC endpoint in any subnets from which you need to access the service or resource. This creates elastic network interfaces in the specified subnets that serve as entry points for traffic destined to the service or resource.

You can also create your own VPC endpoint service, powered by AWS PrivateLink and enable other AWS customers to access your service. PrivateLink enables the creation of private API endpoints, allowing organizations to expose their own services securely to other AWS customers. This empowers businesses to monetize their internal capabilities, foster collaborative ecosystems, and maintain control over how their services are accessed and consumed.

One of the key benefits of using AWS PrivateLink is the ability to establish secure, private connectivity without the need for traditional networking constructs like internet gateways, NAT devices, or VPN connections. This helps simplify the network architecture, reduce the attack surface, and improve overall security by keeping the data traffic confined within the AWS network.

The following diagram shows common use cases for AWS PrivateLink. The VPC has several EC2 instances in a private subnet that have access to resources through five VPC endpoints. There are three interface VPC endpoints, one resource VPC endpoint, and one service-network VPC endpoint.



For more information, see [AWS PrivateLink](https://docs.aws.amazon.com/vpc/latest/privatelink/).

Filter network traffic using AWS Network Firewall

You can filter network traffic at the perimeter of your VPC using AWS Network Firewall. Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service. For more information, see the AWS Network Firewall Developer Guide.

You implement Network Firewall with the following AWS resources.

Network Firewall resource	Description
Firewall	A firewall connects a firewall policy's network traffic filtering be havior to the VPC that you want to protect. The firewall configura tion 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 f irewall. 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 f irewall 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*.

Filter DNS traffic using Route 53 Resolver DNS Firewall

With DNS Firewall, you define domain name filtering rules in rule groups that you associate with your VPCs. You can specify lists of domain names to allow or block, and you can customize the responses for the DNS queries that you block. For more information, see the Route 53 Resolver DNS Firewall Documentation.

You implement DNS Firewall with the following AWS resources.

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 specifica tions 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*.

Troubleshoot reachability issues using Reachability Analyzer

Reachability Analyzer is a static configuration analysis tool. Use Reachability Analyzer to analyze and debug network reachability between two resources in your VPC. Reachability Analyzer produces hop-by-hop details of the virtual path between these resources when they are reachable, and identifies the blocking component otherwise.

You can use Reachability Analyzer to analyze reachability between the following resources:

- Instances
- Internet gateways
- Network interfaces
- Transit gateways
- Transit gateway attachments
- VPC endpoint services
- VPC endpoints
- VPC peering connections
- VPN gateways

For more information, see the Reachability Analyzer Guide.

VPC examples

Amazon Virtual Private Cloud (VPC) is a fundamental building block within the AWS ecosystem, allowing you to provision isolated virtual networks tailored to your specific needs. By creating and managing your own VPCs, you gain full control over the networking environment, including the ability to define IP address ranges, subnets, routing tables, and connectivity options.

This section contains three example configurations for your virtual private clouds (VPCs), each designed to address a different set of requirements:

- VPC for a test environment: This configuration shows how to create a VPC that you can use as a development or test environment.
- VPC for Web and database servers: This configuration shows how to create a VPC that you can use for a resilent architecture in a production environment.
- VPC with servers in private subnets and NAT: In this more advanced configuration, all EC2 instances are provisioned within private subnets, with a NAT gateway facilitating secure outbound internet access. This is an example where you need to limit direct internet connectivity to your resources while still enabling necessary outbound communication.

By providing these example VPC configurations, we hope to illustrate the flexibility and customization options available when designing your cloud networking environment. The specific VPC setup you choose should be based on your application's architecture, security requirements, and overall business objectives. Carefully planning your VPC infrastructure can help you create a robust, scalable, and secure virtual network that supports the growth and evolution of your cloud-based workloads.

Examples

- Example: VPC for a test environment
- Example: VPC for web and database servers
- Example: VPC with servers in private subnets and NAT

Related examples

- To connect your VPCs to each other, see VPC peering configurations in the *Amazon VPC Peering Guide*.

- To connect your VPCs to your own network, see Site-to-Site VPN scenarios in the *AWS Site-to-Site VPN User Guide*.
- To connect your VPCs to each other and to your own network, see Example transit gateway scenarios in the *Amazon VPC Transit Gateways*.

Additional resources

- Understand resiliency patterns and trade-offs (AWS Architecture Blog)
- Plan your network topology (AWS Well-Architected Framework)
- Amazon Virtual Private Cloud Connectivity Options (AWS Whitepapers)

Example: VPC for a test environment

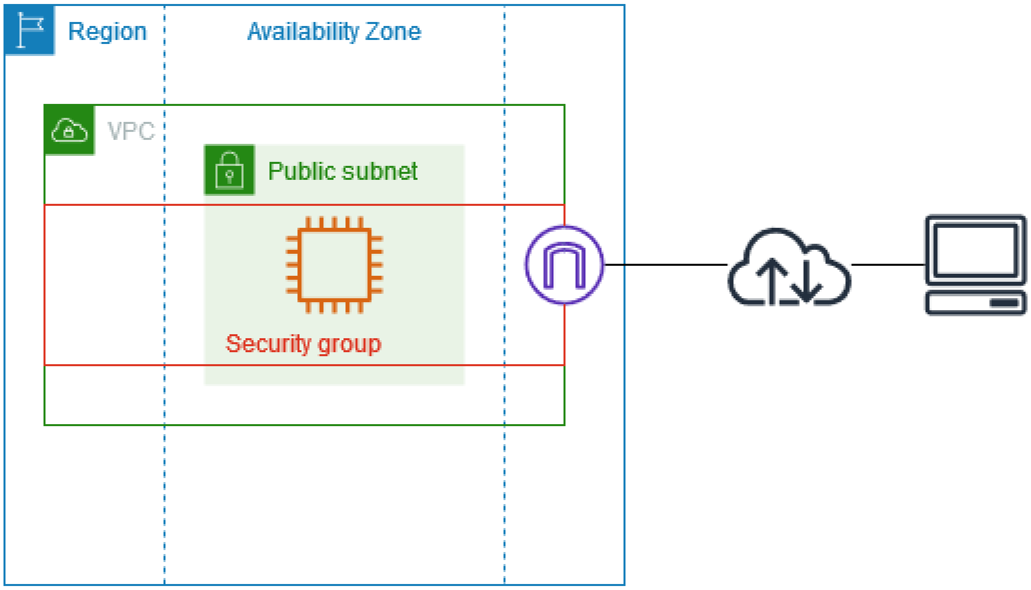
This example demonstrates how to create a VPC that you can use as a development or test environment. Because this VPC is not intended to be used in production, it is not necessary to deploy your servers in multiple Availability Zones. To keep the cost and complexity low, you can deploy your servers in a single Availability Zone.

Contents

- Overview
- 1. Create the VPC
- 2. Deploy your application
- 3. Test your configuration
- 4. Clean up

Overview

The following diagram provides an overview of the resources included in this example. The VPC has a public subnet in a single Availability Zone and an internet gateway. The server is an EC2 instance that runs in the public subnet. The security group for the instance allows SSH traffic from your own computer, plus any other traffic specifically required for your development or testing activities.



Routing

When you create this VPC by using the Amazon VPC console, we create a route table for the public subnet with local routes and routes to the internet gateway. The following is an example of the route table with routes for both IPv4 and IPv6. If you create an IPv4-only subnet instead of a dual stack subnet, your route table has only the IPv4 routes.

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

For this example configuration, you must create a security group for your instance that allows the traffic that your application needs. For example, you might need to add a rule that allows SSH traffic from your computer or HTTP traffic from your network.

The following are example inbound rules for a security group, with rules for both IPv4 and IPv6. If you create IPv4-only subnets instead of dual stack subnets, you need only the rules for IPv4.

Source	Protocol	Port range	Description
0.0.0.0/0	TCP	80	Allows inbound HTTP access from all IPv4 addresses
::/0	TCP	80	Allows inbound HTTP access from all IPv6 addresses
0.0.0.0/0	TCP	443	Allows inbound HTTPS access from all IPv4 addresses
::/0	TCP	443	Allows inbound HTTPS access from all IPv6 addresses
Public IPv4 address range of your network	TCP	22	(Optional) Allows inbound SSH access from IPv4 IP addresses in your network
IPv6 address range of your network	TCP	22	(Optional) Allows inbound SSH access from IPv6 IP addresses in your network
Public IPv4 address range of your network	TCP	3389	(Optional) Allows inbound RDP access from IPv4 IP addresses in your network
IPv6 address range of your network	TCP	3389	(Optional) Allows inbound RDP access from IPv6 IP addresses in your network

1. Create the VPC

Use the following procedure to create a VPC with a public subnet in one Availability Zone. This configuration is suitable for a development or testing environment.

To create the VPC

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

2. On the dashboard, choose **Create VPC**.
3. For **Resources to create**, choose **VPC and more**.
4. **Configure the VPC**

    a. For **Name tag auto-generation**, enter a name for the VPC.

    b. For **IPv4 CIDR block**, you can keep the default suggestion, or alternatively you can enter the CIDR block required by your application or network. For more information, see the section called "VPC CIDR blocks".

    c. (Optional) If your application communicates by using IPv6 addresses, choose **IPv6 CIDR block**, **Amazon-provided IPv6 CIDR block**.

5. **Configure the subnets**

    a. For **Number of Availability Zones**, choose **1**. You can keep the default Availability Zone, or alternatively you can expand **Customize AZs** and select an Availability Zone.

    b. For **Number of public subnets**, choose **1**.

    c. For **Number of private subnets**, choose **0**.

    d. You can keep the default CIDR block for the public subnet, or alternatively you can expand **Customize subnet CIDR blocks** and enter a CIDR block. For more information, see the section called "Subnet CIDR blocks".

6. For **NAT gateways**, keep the default value, **None**.
7. For **VPC endpoints**, choose **None**. A gateway VPC endpoint for S3 is used only to access Amazon S3 from private subnets.
8. For **DNS options**, keep both options selected. As a result, your instance will receive a public DNS hostname that corresponds to its public IP address.
9. Choose **Create VPC**.

2. Deploy your application

There are a variety of ways to deploy EC2 instances. For example:

- Amazon EC2 launch instance wizard
- Amazon EC2 Auto Scaling
- AWS CloudFormation
- Amazon Elastic Container Service (Amazon ECS)

After you deploy an EC2 instance, you can connect to the instance, install the software that you need for your application, and then create an image for future use. For more information, see Create an AMI in the *Amazon EC2 User Guide*. Alternatively, you can use EC2 Image Builder to create and manage your Amazon Machine Image (AMI).

3. Test your configuration

After you've finished deploying your application, you can test it. If you can't connect to your EC2 instance, or if your application can't send or receive the traffic that you expect, you can use Reachability Analyzer to help you troubleshoot. For example, Reachability Analyzer can identify configuration issues with your route tables or security groups. For more information, see the Reachability Analyzer Guide.

4. Clean up

When you are finished with this configuration, you can delete it. Before you can delete the VPC, you must terminate your instance. For more information, see the section called “Delete your VPC”.

Example: VPC for web and database servers

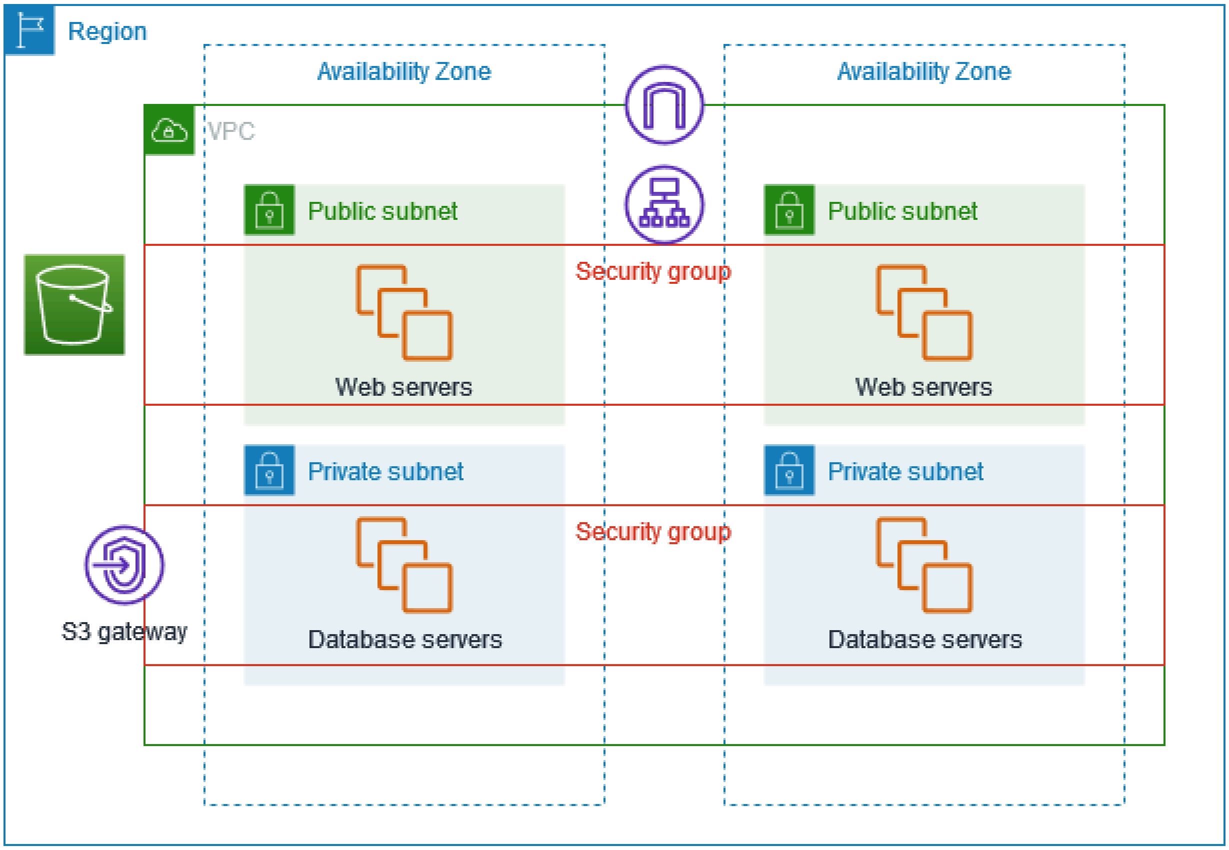
This example demonstrates how to create a VPC that you can use for a two-tier architecture in a production environment. To improve resiliency, you deploy the servers in two Availability Zones.

Contents

- Overview
- 1. Create the VPC
- 2. Deploy your application
- 3. Test your configuration
- 4. Clean up

Overview

The following diagram provides an overview of the resources included in this example. The VPC has public subnets and private subnets in two Availability Zones. The web servers run in the public subnets and receive traffic from clients through a load balancer. The security group for the web servers allows traffic from the load balancer. The database servers run in the private subnets and receive traffic from the web servers. The security group for the database servers allows traffic from the web servers. The database servers can connect to Amazon S3 by using a gateway VPC endpoint.



Routing

When you create this VPC by using the Amazon VPC console, we create a route table for the public subnets with local routes and routes to the internet gateway, and a route table for each private subnet with local routes and a route to the gateway VPC endpoint.

The following is an example of a route table for the public subnets, with routes for both IPv4 and IPv6. If you create IPv4-only subnets instead of dual stack subnets, your route table has only the IPv4 routes.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local

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

The following is an example of a route table for the private subnets, with local routes for both IPv4 and IPv6. If you created IPv4-only subnets, your route table has only the IPv4 route. The last route sends traffic destined for Amazon S3 to the gateway VPC endpoint.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
s3-prefix-list-id	*s3-gateway-id*

Security

For this example configuration, you create a security group for the load balancer, a security group for the web servers, and a security group for the database servers.

Load balancer

The security group for your Application Load Balancer or Network Load Balancer must allow inbound traffic from clients on the load balancer listener port. To accept traffic from anywhere on the internet, specify 0.0.0.0/0 as the source. The load balancer security group must also allow outbound traffic from the load balancer to the target instances on the instance listener port and the health check port.

Web servers

The following security group rules allow the web servers to receive HTTP and HTTPS traffic from the load balancer. You can optionally allow the web servers to receive SSH or RDP traffic from your network. The web servers can send SQL or MySQL traffic to your database servers.

Source	Protocol	Port range	Description
ID of the security group for the load balancer	TCP	80	Allows inbound HTTP access from the load balancer
ID of the security group for the load balancer	TCP	443	Allows inbound HTTPS access from the load balancer
Public IPv4 address range of your network	TCP	22	(Optional) Allows inbound SSH access from IPv4 IP addresses in your network
IPv6 address range of your network	TCP	22	(Optional) Allows inbound SSH access from IPv6 IP addresses in your network
Public IPv4 address range of your network	TCP	3389	(Optional) Allows inbound RDP access from IPv4 IP addresses in your network
IPv6 address range of your network	TCP	3389	(Optional) Allows inbound RDP access from IPv6 IP addresses in your network

Destination	Protocol	Port range	Description
ID of the security group for instances running Microsoft SQL Server	TCP	1433	Allows outbound Microsoft SQL Server access to the database servers
ID of the security group for	TCP	3306	Allows outbound MySQL access to the database servers

Destination	Protocol	Port range	Description
instances running MySQL			

Database servers

The following security group rules allow the database servers to receive read and write requests from the web servers.

Source	Protocol	Port range	Comments
ID of the web server security group	TCP	1433	Allows inbound Microsoft SQL Server access from the web servers
ID of the web server security group	TCP	3306	Allows inbound MySQL Server access from the web servers

Destination	Protocol	Port range	Comments
0.0.0.0/0	TCP	80	Allows outbound HTTP access to the internet over IPv4
0.0.0.0/0	TCP	443	Allows outbound HTTPS access to the internet over IPv4

For more information about security groups for Amazon RDS DB instances, see Controlling access with security groups in the *Amazon RDS User Guide*.

1. Create the VPC

Use the following procedure to create a VPC with a public subnet and a private subnet in two Availability Zones.

To create the VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the dashboard, choose **Create VPC**.
3. For **Resources to create**, choose **VPC and more**.
4. Configure the VPC:
   a. Keep **Name tag auto-generation** selected to create Name tags for the VPC resources or clear it to provide your own Name tags for the VPC resources.
   b. For **IPv4 CIDR block**, you can keep the default suggestion, or alternatively you can enter the CIDR block required by your application or network. For more information, see the section called "VPC CIDR blocks".
   c. (Optional) If your application communicates by using IPv6 addresses, choose **IPv6 CIDR block**, **Amazon-provided IPv6 CIDR block**.
   d. Choose a **Tenancy** option. This option defines if EC2 instances that you launch into the VPC will run on hardware that's shared with other AWS accounts or on hardware that's dedicated for your use only. If you choose the tenancy of the VPC to be `Default`, EC2 instances launched into this VPC will use the tenancy attribute specified when you launch the instance. For more information, see Launch an instance using defined parameters in the *Amazon EC2 User Guide*. If you choose the tenancy of the VPC to be `Dedicated`, the instances will always run as Dedicated Instances on hardware that's dedicated for your use.
5. Configure the subnets:
   a. For **Number of Availability Zones**, choose **2**, so that you can launch instances in two Availability Zones to improve resiliency.
   b. For **Number of public subnets**, choose **2**.
   c. For **Number of private subnets**, choose **2**.
   d. You can keep the default CIDR blocks for the subnets, or alternatively you can expand **Customize subnets CIDR blocks** and enter a CIDR block. For more information, see the section called "Subnet CIDR blocks".
6. For **NAT gateways**, keep the default value, **None**.
7. For **VPC endpoints**, keep the default value, **S3 Gateway**. While there is no effect unless you access an S3 bucket, there is no cost to enable this VPC endpoint.
8. For **DNS options**, keep both options selected. As a result, your web servers will receive public DNS hostnames that correspond to their public IP addresses.

9. Choose **Create VPC**.

2. Deploy your application

Ideally, you've already tested your web servers and database servers in a development or test environment, and created the scripts or images that you'll use to deploy your application in production.

You can use EC2 instances for your web servers. There are a variety of ways to deploy EC2 instances. For example:

- Amazon EC2 launch instance wizard
- AWS CloudFormation
- Amazon Elastic Container Service (Amazon ECS)

To improve availability, you can use Amazon EC2 Auto Scaling to deploy servers in multiple Availability Zones and maintain the minimum server capacity that is required by your application.

You can use Elastic Load Balancing to distribute traffic evenly across your servers. You can attach your load balancer to an Auto Scaling group.

You can use EC2 instances for your database servers, or one of our purpose-built database types. For more information, see Databases on AWS: How to choose.

3. Test your configuration

After you've finished deploying your application, you can test it. If your application can't send or receive the traffic that you expect, you can use Reachability Analyzer to help you troubleshoot. For example, Reachability Analyzer can identify configuration issues with your route tables or security groups. For more information, see the Reachability Analyzer Guide.

4. Clean up

When you are finished with this configuration, you can delete it. Before you can delete the VPC, you must terminate your instances and delete the load balancer. For more information, see the section called “Delete your VPC”.

Example: VPC with servers in private subnets and NAT

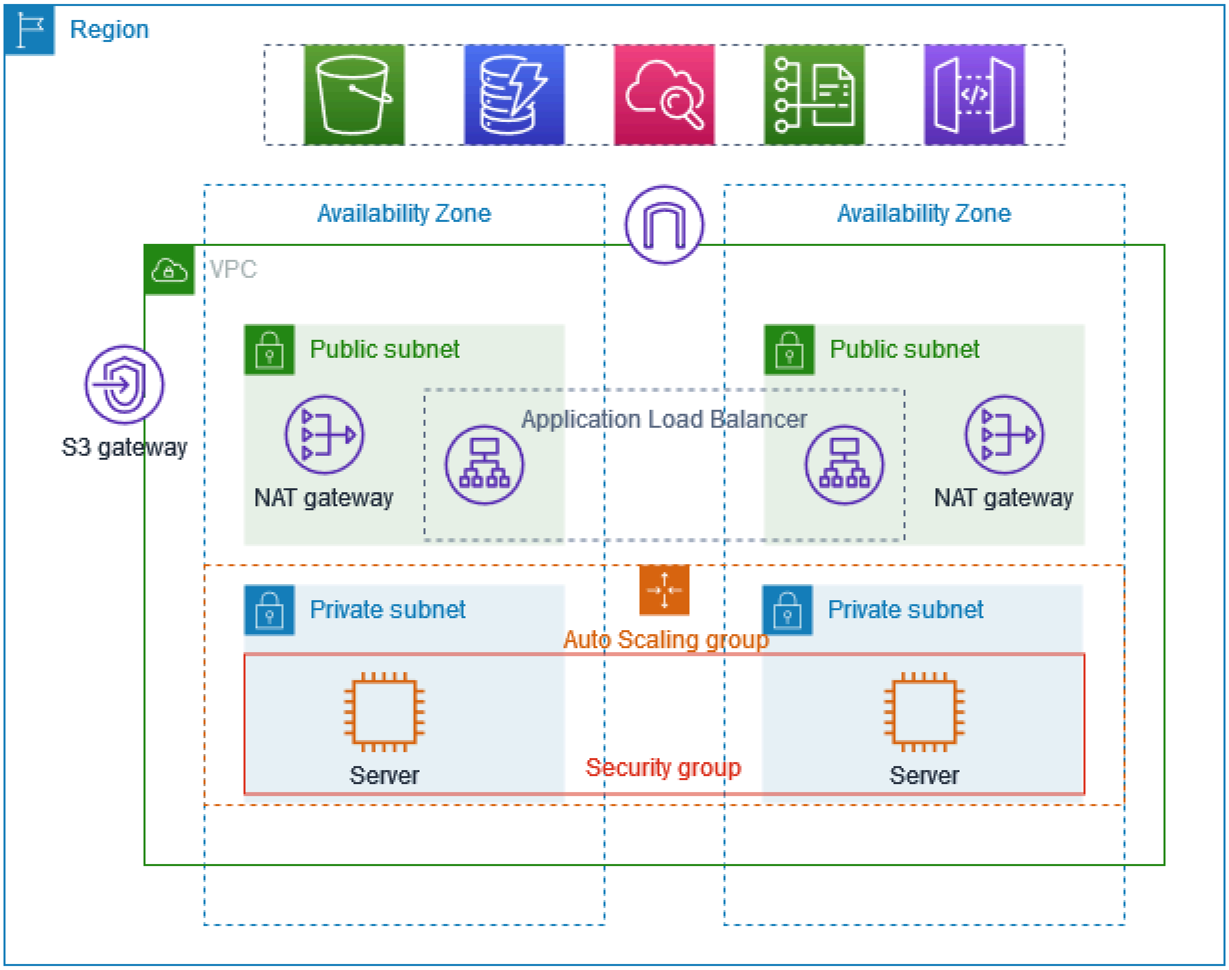
This example demonstrates how to create a VPC that you can use for servers in a production environment. To improve resiliency, you deploy the servers in two Availability Zones, by using an Auto Scaling group and an Application Load Balancer. For additional security, you deploy the servers in private subnets. The servers receive requests through the load balancer. The servers can connect to the internet by using a NAT gateway. To improve resiliency, you deploy the NAT gateway in both Availability Zones.

Contents

- Overview
- 1. Create the VPC
- 2. Deploy your application
- 3. Test your configuration
- 4. Clean up

Overview

The following diagram provides an overview of the resources included in this example. The VPC has public subnets and private subnets in two Availability Zones. Each public subnet contains a NAT gateway and a load balancer node. The servers run in the private subnets, are launched and terminated by using an Auto Scaling group, and receive traffic from the load balancer. The servers can connect to the internet by using the NAT gateway. The servers can connect to Amazon S3 by using a gateway VPC endpoint.



Routing

When you create this VPC by using the Amazon VPC console, we create a route table for the public subnets with local routes and routes to the internet gateway. We also create a route table for the private subnets with local routes, and routes to the NAT gateway, egress-only internet gateway, and gateway VPC endpoint.

The following is an example of the route table for the public subnets, with routes for both IPv4 and IPv6. If you create IPv4-only subnets instead of dual stack subnets, your route table includes only the IPv4 routes.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local

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

The following is an example of a route table for one of the private subnets, with routes for both IPv4 and IPv6. If you created IPv4-only subnets, the route table includes only the IPv4 routes. The last route sends traffic destined for Amazon S3 to the gateway VPC endpoint.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	*nat-gateway-id*
::/0	*eigw-id*
s3-prefix-list-id	*s3-gateway-id*

Security

The following is an example of the rules that you might create for the security group that you associate with your servers. The security group must allow traffic from the load balancer over the listener port and protocol. It must also allow health check traffic.

Source	Protocol	Port range	Comments
ID of the load balancer security group	*listener protocol*	*listener port*	Allows inbound traffic from the load balancer on the listener port
ID of the load balancer security group	*health check protocol*	*health check port*	Allows inbound health check traffic from the load balancer

1. Create the VPC

Use the following procedure to create a VPC with a public subnet and a private subnet in two Availability Zones, and a NAT gateway in each Availability Zone.

To create the VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the dashboard, choose **Create VPC**.
3. For **Resources to create**, choose **VPC and more**.
4. **Configure the VPC**
    a. For **Name tag auto-generation**, enter a name for the VPC.
    b. For **IPv4 CIDR block**, you can keep the default suggestion, or alternatively you can enter the CIDR block required by your application or network.
    c. If your application communicates by using IPv6 addresses, choose **IPv6 CIDR block**, **Amazon-provided IPv6 CIDR block**.
5. **Configure the subnets**
    a. For **Number of Availability Zones**, choose **2**, so that you can launch instances in multiple Availability Zones to improve resiliency.
    b. For **Number of public subnets**, choose **2**.
    c. For **Number of private subnets**, choose **2**.
    d. You can keep the default CIDR block for the public subnet, or alternatively you can expand **Customize subnet CIDR blocks** and enter a CIDR block. For more information, see the section called “Subnet CIDR blocks”.
6. For **NAT gateways**, choose **1 per AZ** to improve resiliency.
7. If your application communicates by using IPv6 addresses, for **Egress only internet gateway**, choose **Yes**.
8. For **VPC endpoints**, if your instances must access an S3 bucket, keep the **S3 Gateway** default. Otherwise, instances in your private subnet can't access Amazon S3. There is no cost for this option, so you can keep the default if you might use an S3 bucket in the future. If you choose **None**, you can always add a gateway VPC endpoint later on.
9. For **DNS options**, clear **Enable DNS hostnames**.
10. Choose **Create VPC**.

2. Deploy your application

Ideally, you've finished testing your servers in a development or test environment, and created the scripts or images that you'll use to deploy your application in production.

You can use Amazon EC2 Auto Scaling to deploy servers in multiple Availability Zones and maintain the minimum server capacity required by your application.

To launch instances by using an Auto Scaling group

1. Create a launch template to specify the configuration information needed to launch your EC2 instances by using Amazon EC2 Auto Scaling. For step-by-step directions, see Create a launch template for your Auto Scaling group in the *Amazon EC2 Auto Scaling User Guide*.
2. Create an Auto Scaling group, which is a collection of EC2 instances with a minimum, maximum, and desired size. For step-by-step directions, see Create an Auto Scaling group using a launch template in the *Amazon EC2 Auto Scaling User Guide*.
3. Create a load balancer, which distributes traffic evenly across the instances in your Auto Scaling group, and attach the load balancer to your Auto Scaling group. For more information, see the Elastic Load Balancing User Guide and Use Elastic Load Balancing in the *Amazon EC2 Auto Scaling User Guide*.

3. Test your configuration

After you've finished deploying your application, you can test it. If your application can't send or receive the traffic that you expect, you can use Reachability Analyzer to help you troubleshoot. For example, Reachability Analyzer can identify configuration issues with your route tables or security groups. For more information, see the Reachability Analyzer Guide.

4. Clean up

When you are finished with this configuration, you can delete it. Before you can delete the VPC, you must delete the Auto Scaling group, terminate your instances, delete the NAT gateways, and delete the load balancer. For more information, see the section called “Delete your VPC”.

VPC Tutorials

Amazon Virtual Private Cloud (VPC) is the foundation of your network infrastructure in AWS. While the AWS Management Console provides a user-friendly interface, the AWS Command Line Interface (CLI) offers greater flexibility and automation capabilities for creating and managing your VPC resources.

This guide presents two essential VPC deployment scenarios:

- A basic VPC setup with a public subnet, ideal for simple web applications
- An advanced VPC configuration with both private and public subnets using NAT gateways, suitable for multi-tier applications

By following these tutorials, you'll learn how to:

- Create VPCs with different subnet configurations
- Set up internet and NAT gateways
- Configure route tables and security groups
- Use AWS CLI commands to manage your network infrastructure
- Implement AWS networking best practices

Let's start building your AWS network infrastructure using the command line.

Tutorials

- Getting started with Amazon VPC using the AWS CLI
- Create a VPC with private subnets and NAT gateways using AWS CLI

Getting started with Amazon VPC using the AWS CLI

This tutorial guides you through creating a Virtual Private Cloud (VPC) using the AWS Command Line Interface (AWS CLI). You'll learn how to set up a VPC with public and private subnets, configure internet connectivity, and deploy EC2 instances to demonstrate a common web application architecture.

Prerequisites

Before you begin this tutorial, make sure you have the following:

1. The AWS CLI. If you need to install it, follow the AWS CLI installation guide.
2. Configured your AWS CLI with appropriate credentials. Run `aws configure` if you haven't set up your credentials yet.
3. Basic understanding of networking concepts.
4. Identity and access management for Amazon VPC to create and manage VPC resources in your AWS account.

Cost considerations

This tutorial creates AWS resources that may incur costs in your account. The primary cost comes from the NAT Gateway ($0.045 per hour plus data processing charges) and EC2 instances (t2.micro, approximately $0.0116 per hour each). If you complete this tutorial in one hour and then clean up all resources, the total cost will be approximately $0.07. For cost optimization in development environments, consider using a NAT Instance instead of a NAT Gateway, which can reduce costs significantly.

Let's verify that your AWS CLI is properly configured before proceeding.

```
aws configure list
```

You should see your AWS access key, secret key, and default region. Also, verify that you have the necessary permissions to create VPC resources.

```
aws sts get-caller-identity
```

This command displays your AWS account ID, user ID, and ARN, confirming that your credentials are valid.

Create a VPC

A Virtual Private Cloud (VPC) is a virtual network dedicated to your AWS account. In this section, you'll create a VPC with a CIDR block of 10.0.0.0/16, which provides up to 65,536 IP addresses.

Create the VPC

The following command creates a new VPC and assigns it a name tag.

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 'ResourceType=vpc,Tags=[{Key=Name,Value=MyVPC}]'
```

Take note of the VPC ID in the output. You'll need it for subsequent commands. For the purpose of this tutorial, we'll use "vpc-0123456789abcdef0" as an example VPC ID. Replace this with your actual VPC ID in all commands.

Enable DNS support and hostnames

By default, DNS resolution and DNS hostnames are disabled in a new VPC. Enable these features to allow instances in your VPC to resolve domain names.

```
aws ec2 modify-vpc-attribute --vpc-id vpc-0123456789abcdef0 --enable-dns-support
aws ec2 modify-vpc-attribute --vpc-id vpc-0123456789abcdef0 --enable-dns-hostnames
```

These commands don't produce output if successful. Your VPC now has DNS support and hostname resolution enabled.

Create subnets

Subnets are segments of a VPC's IP address range where you can place groups of isolated resources. In this section, you'll create public and private subnets in two Availability Zones for high availability.

Get available Availability Zones

First, retrieve the Availability Zones available in your region.

```
aws ec2 describe-availability-zones
```

For this tutorial, we'll use the first two Availability Zones. Note their names from the output (e.g., "us-east-1a" and "us-east-1b").

Create public subnets

Public subnets are used for resources that need to be accessible from the internet, such as web servers.

```
aws ec2 create-subnet \
  --vpc-id vpc-0123456789abcdef0 \
  --cidr-block 10.0.0.0/24 \
  --availability-zone us-east-1a \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-AZ1}]'
```

Note the subnet ID from the output. For this tutorial, we'll use "subnet-0123456789abcdef0" as an example for the first public subnet.

```
aws ec2 create-subnet \
  --vpc-id vpc-0123456789abcdef0 \
  --cidr-block 10.0.1.0/24 \
  --availability-zone us-east-1b \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-AZ2}]'
```

Note the subnet ID from the output. For this tutorial, we'll use "subnet-0123456789abcdef1" as an example for the second public subnet.

Create private subnets

Private subnets are used for resources that should not be directly accessible from the internet, such as databases.

```
aws ec2 create-subnet \
  --vpc-id vpc-0123456789abcdef0 \
  --cidr-block 10.0.2.0/24 \
  --availability-zone us-east-1a \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-AZ1}]'
```

Note the subnet ID from the output. For this tutorial, we'll use "subnet-0123456789abcdef2" as an example for the first private subnet.

```
aws ec2 create-subnet \
  --vpc-id vpc-0123456789abcdef0 \
  --cidr-block 10.0.3.0/24 \
  --availability-zone us-east-1b \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-AZ2}]'
```

Note the subnet ID from the output. For this tutorial, we'll use "subnet-0123456789abcdef3" as an example for the second private subnet.

You now have four subnets: two public subnets and two private subnets, distributed across two Availability Zones.

Tip: When planning your CIDR blocks, ensure they don't overlap with your existing networks. For production environments, allocate enough IP addresses for future growth while keeping subnets reasonably sized for security and management.

Configure internet connectivity

To allow resources in your VPC to communicate with the internet, you need to create and attach an Internet Gateway. In this section, you'll set up internet connectivity for your VPC.

Create an Internet Gateway

An Internet Gateway enables communication between your VPC and the internet.

```
aws ec2 create-internet-gateway \
  --tag-specifications 'ResourceType=internet-gateway,Tags=[{Key=Name,Value=MyIGW}]'
```

Note the Internet Gateway ID from the output. For this tutorial, we'll use "igw-0123456789abcdef0" as an example.

Attach the Internet Gateway to your VPC

After creating the Internet Gateway, attach it to your VPC.

```
aws ec2 attach-internet-gateway --internet-gateway-id igw-0123456789abcdef0 --vpc-id
 vpc-0123456789abcdef0
```

Create and configure route tables

Route tables contain rules (routes) that determine where network traffic is directed. First, create a route table for your public subnets.

```
aws ec2 create-route-table \
  --vpc-id vpc-0123456789abcdef0 \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Public-RT}]'
```

Note the route table ID from the output. For this tutorial, we'll use "rtb-0123456789abcdef0" as an example for the public route table.

Add a route to the Internet Gateway in the public route table.

```
aws ec2 create-route --route-table-id rtb-0123456789abcdef0 --destination-cidr-block
 0.0.0.0/0 --gateway-id igw-0123456789abcdef0
```

Associate the public subnets with the public route table.

```
aws ec2 associate-route-table --route-table-id rtb-0123456789abcdef0 --subnet-id
 subnet-0123456789abcdef0
aws ec2 associate-route-table --route-table-id rtb-0123456789abcdef0 --subnet-id
 subnet-0123456789abcdef1
```

Now, create a route table for your private subnets.

```
aws ec2 create-route-table \
  --vpc-id vpc-0123456789abcdef0 \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Private-RT}]'
```

Note the route table ID from the output. For this tutorial, we'll use "rtb-0123456789abcdef1" as an example for the private route table.

Associate the private subnets with the private route table.

```
aws ec2 associate-route-table --route-table-id rtb-0123456789abcdef1 --subnet-id
 subnet-0123456789abcdef2
aws ec2 associate-route-table --route-table-id rtb-0123456789abcdef1 --subnet-id
 subnet-0123456789abcdef3
```

Create a NAT Gateway

A NAT Gateway allows instances in private subnets to initiate outbound traffic to the internet while preventing inbound traffic from the internet. This is essential for instances that need to download updates or access external services.

Allocate an Elastic IP

First, allocate an Elastic IP address for your NAT Gateway.

```
aws ec2 allocate-address --domain vpc
```

Note the Allocation ID from the output. For this tutorial, we'll use "eipalloc-0123456789abcdef0" as an example.

Create the NAT Gateway

Create a NAT Gateway in one of your public subnets using the allocated Elastic IP.

```
aws ec2 create-nat-gateway \
  --subnet-id subnet-0123456789abcdef0 \
  --allocation-id eipalloc-0123456789abcdef0 \
  --tag-specifications 'ResourceType=natgateway,Tags=[{Key=Name,Value=MyNATGateway}]'
```

Note the NAT Gateway ID from the output. For this tutorial, we'll use "nat-0123456789abcdef0" as an example.

Wait for the NAT Gateway to become available before proceeding.

```
aws ec2 wait nat-gateway-available --nat-gateway-ids nat-0123456789abcdef0
```

Add a route to the NAT Gateway

Add a route to the NAT Gateway in the private route table to allow instances in private subnets to access the internet.

```
aws ec2 create-route --route-table-id rtb-0123456789abcdef1 --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id nat-0123456789abcdef0
```

Note: For production environments, consider creating a NAT Gateway in each Availability Zone where you have private subnets to eliminate single points of failure.

Configure subnet settings

Configure your public subnets to automatically assign public IP addresses to instances launched in them.

```
aws ec2 modify-subnet-attribute --subnet-id subnet-0123456789abcdef0 --map-public-ip-
on-launch
aws ec2 modify-subnet-attribute --subnet-id subnet-0123456789abcdef1 --map-public-ip-
on-launch
```

This ensures that instances launched in your public subnets receive a public IP address by default, making them accessible from the internet.

Create security groups

Security groups act as virtual firewalls for your instances to control inbound and outbound traffic. In this section, you'll create security groups for web servers and database servers.

Create a security group for web servers

```
aws ec2 create-security-group \
  --group-name WebServerSG \
  --description "Security group for web servers" \
  --vpc-id vpc-0123456789abcdef0
```

Note the security group ID from the output. For this tutorial, we'll use "sg-0123456789abcdef0" as an example for the web server security group.

Allow HTTP and HTTPS traffic to your web servers.

```
aws ec2 authorize-security-group-ingress --group-id sg-0123456789abcdef0 --protocol tcp
 --port 80 --cidr 0.0.0.0/0
aws ec2 authorize-security-group-ingress --group-id sg-0123456789abcdef0 --protocol tcp
 --port 443 --cidr 0.0.0.0/0
```

Note: For production environments, restrict inbound traffic to specific IP ranges rather than allowing traffic from 0.0.0.0/0 (any IP address).

Create a security group for database servers

```
aws ec2 create-security-group \
  --group-name DBServerSG \
  --description "Security group for database servers" \
  --vpc-id vpc-0123456789abcdef0
```

Note the security group ID from the output. For this tutorial, we'll use "sg-0123456789abcdef1" as an example for the database server security group.

Allow MySQL/Aurora traffic from web servers only.

```
aws ec2 authorize-security-group-ingress --group-id sg-0123456789abcdef1 --protocol tcp
 --port 3306 --source-group sg-0123456789abcdef0
```

This configuration ensures that only instances in the web server security group can connect to your database servers on port 3306, following the principle of least privilege.

Verify your VPC configuration

After creating all the necessary components, verify your VPC configuration to ensure everything is set up correctly.

Check your VPC

```
aws ec2 describe-vpcs --vpc-id vpc-0123456789abcdef0
```

Check your subnets

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-0123456789abcdef0"
```

Check your route tables

```
aws ec2 describe-route-tables --filters "Name=vpc-id,Values=vpc-0123456789abcdef0"
```

Check your Internet Gateway

```
aws ec2 describe-internet-gateways --filters "Name=attachment.vpc-
id,Values=vpc-0123456789abcdef0"
```

Check your NAT Gateway

```
aws ec2 describe-nat-gateways --filter "Name=vpc-id,Values=vpc-0123456789abcdef0"
```

Check your security groups

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=vpc-0123456789abcdef0"
```

These commands provide detailed information about each component of your VPC, allowing you to verify that everything is configured correctly.

Deploy EC2 instances

Now that you have created your VPC infrastructure, you can deploy EC2 instances to demonstrate how the architecture works. You'll launch a web server in a public subnet and a database server in a private subnet.

Create a key pair for SSH access

First, create a key pair to securely connect to your instances:

```
aws ec2 create-key-pair --key-name vpc-tutorial-key --query 'KeyMaterial' --output text
 > vpc-tutorial-key.pem
chmod 400 vpc-tutorial-key.pem
```

This command creates a new key pair and saves the private key to a file with restricted permissions.

Find the latest Amazon Linux 2 AMI

Find the latest Amazon Linux 2 AMI to use for your instances:

```
aws ec2 describe-images --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2" "Name=state,Values=available"
 \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" --output text
```

Note the AMI ID from the output. For this tutorial, we'll use "ami-0123456789abcdef0" as an example.

Launch a web server in the public subnet

Now, launch an EC2 instance in the public subnet to serve as a web server:

```
aws ec2 run-instances \
  --image-id ami-0123456789abcdef0 \
  --count 1 \
  --instance-type t2.micro \
  --key-name vpc-tutorial-key \
  --security-group-ids sg-0123456789abcdef0 \
  --subnet-id subnet-0123456789abcdef0 \
  --associate-public-ip-address \
  --user-data '#!/bin/bash
              yum update -y
```

```
                yum install -y httpd
                systemctl start httpd
                systemctl enable httpd
                echo "<h1>Hello from $(hostname -f)</h1>" > /var/www/html/index.html' \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=WebServer}]'
```

Note the instance ID from the output. For this tutorial, we'll use "i-0123456789abcdef0" as an example for the web server instance.

Launch a database server in the private subnet

Next, launch an EC2 instance in the private subnet to serve as a database server:

```
aws ec2 run-instances \
  --image-id ami-0123456789abcdef0 \
  --count 1 \
  --instance-type t2.micro \
  --key-name vpc-tutorial-key \
  --security-group-ids sg-0123456789abcdef1 \
  --subnet-id subnet-0123456789abcdef2 \
  --user-data '#!/bin/bash
                yum update -y
                yum install -y mariadb-server
                systemctl start mariadb
                systemctl enable mariadb' \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=DBServer}]'
```

Note the instance ID from the output. For this tutorial, we'll use "i-0123456789abcdef1" as an example for the database server instance.

Access your web server

Once your web server instance is running, you can access it using its public IP address:

```
aws ec2 describe-instances \
  --instance-ids i-0123456789abcdef0 \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text
```

This command will output the public IP address of your web server. For this tutorial, we'll use "203.0.113.10" as an example.

You can now open this URL in your web browser: http://203.0.113.10

Connect to your instances via SSH

To connect to your web server:

```
ssh -i vpc-tutorial-key.pem ec2-user@203.0.113.10
```

To connect to your database server, you need to SSH to your web server first and then to your database server:

```
# Get the private IP of the database server
aws ec2 describe-instances \
  --instance-ids i-0123456789abcdef1 \
  --query 'Reservations[0].Instances[0].PrivateIpAddress' \
  --output text
```

This command will output the private IP address of your database server. For this tutorial, we'll use "10.0.2.10" as an example.

```
# First SSH to web server, then to database server
ssh -i vpc-tutorial-key.pem -A ec2-user@203.0.113.10
ssh ec2-user@10.0.2.10
```

This demonstrates the network architecture you've created: the web server is publicly accessible, while the database server is only accessible from within the VPC.

Troubleshooting

Here are some common issues you might encounter when creating a VPC and how to resolve them:

CIDR Block Overlaps

If you receive an error about CIDR block overlaps, ensure that the CIDR blocks for your VPC and subnets don't overlap with existing VPCs or subnets in your account.

Permission Errors

If you encounter permission errors, verify that your IAM user or role has the necessary permissions to create and manage VPC resources. You might need to attach the `AmazonVPCFullAccess` policy or create a custom policy with the required permissions.

Resource Limits

AWS accounts have default limits on the number of VPCs, subnets, and other resources you can create. If you hit these limits, you can request an increase through the AWS Support Center.

Dependency Failures During Cleanup

When cleaning up resources, you might encounter dependency errors if you try to delete resources in the wrong order. Always delete resources in the reverse order of creation, starting with the most dependent resources.

Clean up resources

When you're finished with your VPC, you can clean up the resources to avoid incurring charges. Delete the resources in the reverse order of creation to handle dependencies correctly.

Terminate EC2 instances

```
aws ec2 terminate-instances --instance-ids i-0123456789abcdef0 i-0123456789abcdef1
aws ec2 wait instance-terminated --instance-ids i-0123456789abcdef0 i-0123456789abcdef1
```

Delete the key pair

```
aws ec2 delete-key-pair --key-name vpc-tutorial-key
rm vpc-tutorial-key.pem
```

Delete the NAT Gateway

```
aws ec2 delete-nat-gateway --nat-gateway-id nat-0123456789abcdef0
aws ec2 wait nat-gateway-deleted --nat-gateway-ids nat-0123456789abcdef0
```

Release the Elastic IP

```
aws ec2 release-address --allocation-id eipalloc-0123456789abcdef0
```

Delete security groups

```
aws ec2 delete-security-group --group-id sg-0123456789abcdef1
aws ec2 delete-security-group --group-id sg-0123456789abcdef0
```

Delete route tables

First, find the route table association IDs:

```
aws ec2 describe-route-tables --route-table-id rtb-0123456789abcdef0
aws ec2 describe-route-tables --route-table-id rtb-0123456789abcdef1
```

Then disassociate the route tables from the subnets (replace the association IDs with the ones from your output):

```
aws ec2 disassociate-route-table --association-id rtbassoc-0123456789abcdef0
aws ec2 disassociate-route-table --association-id rtbassoc-0123456789abcdef1
aws ec2 disassociate-route-table --association-id rtbassoc-0123456789abcdef2
aws ec2 disassociate-route-table --association-id rtbassoc-0123456789abcdef3
```

Then delete the route tables:

```
aws ec2 delete-route-table --route-table-id rtb-0123456789abcdef1
aws ec2 delete-route-table --route-table-id rtb-0123456789abcdef0
```

Detach and delete the Internet Gateway

```
aws ec2 detach-internet-gateway --internet-gateway-id igw-0123456789abcdef0 --vpc-id
 vpc-0123456789abcdef0
aws ec2 delete-internet-gateway --internet-gateway-id igw-0123456789abcdef0
```

Delete subnets

```
aws ec2 delete-subnet --subnet-id subnet-0123456789abcdef0
aws ec2 delete-subnet --subnet-id subnet-0123456789abcdef1
aws ec2 delete-subnet --subnet-id subnet-0123456789abcdef2
aws ec2 delete-subnet --subnet-id subnet-0123456789abcdef3
```

Delete the VPC

```
aws ec2 delete-vpc --vpc-id vpc-0123456789abcdef0
```

Going to production

This tutorial is designed to help you learn how to create a VPC using the AWS CLI. For production environments, consider the following security and architecture best practices:

1. **Security Group Rules**: Restrict inbound traffic to specific IP ranges rather than allowing traffic from 0.0.0.0/0.

2. **High Availability**: Deploy NAT Gateways in each Availability Zone where you have private subnets to eliminate single points of failure.
3. **Network ACLs**: Implement Network ACLs as an additional layer of security beyond security groups.
4. **VPC Flow Logs**: Enable VPC Flow Logs to monitor and analyze network traffic patterns.
5. **Resource Tagging**: Implement a comprehensive tagging strategy for better resource management.

For more information on building production-ready architectures, refer to AWS Well-Architected Framework and AWS Security Best Practices.

Next steps

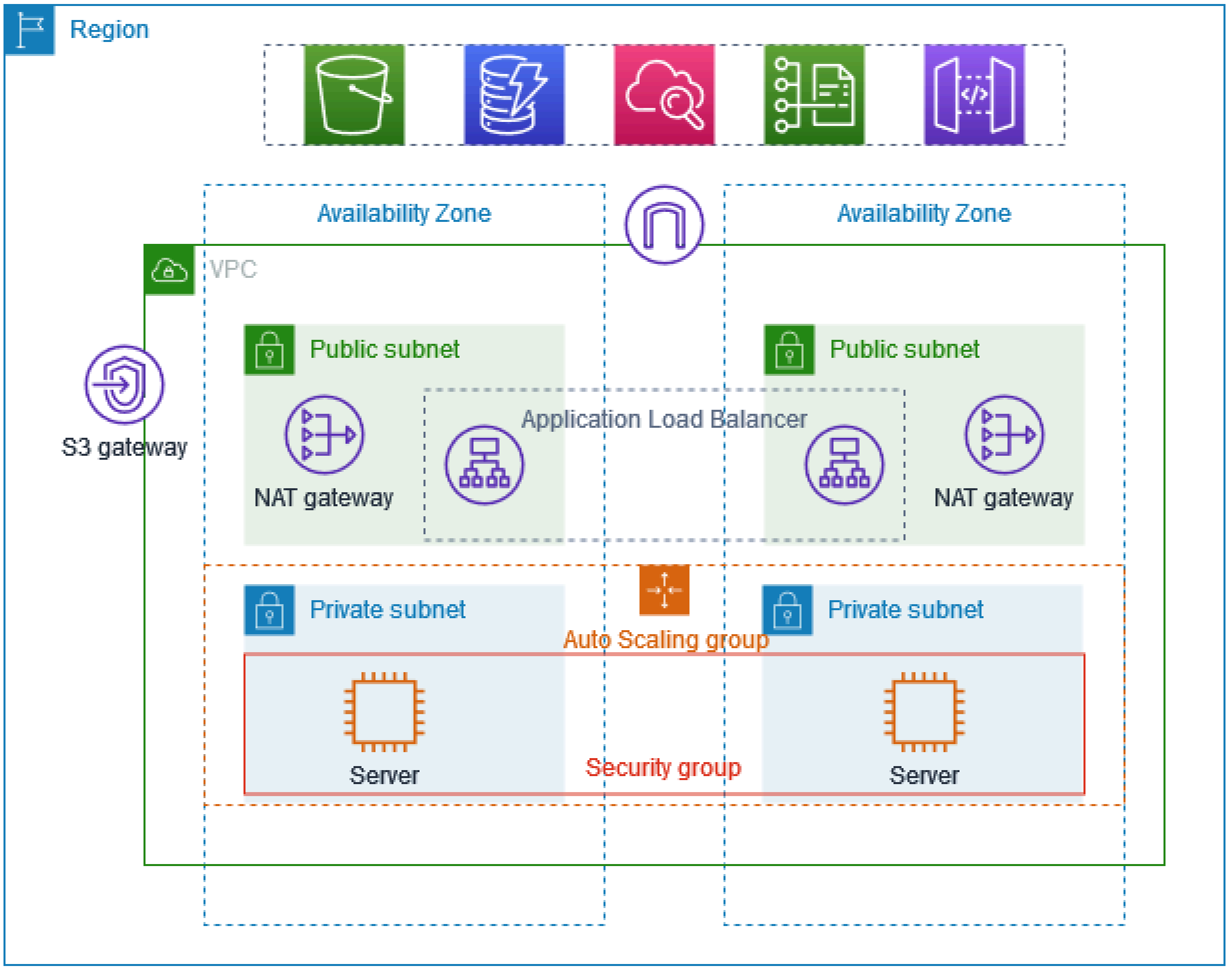
Now that you've created a VPC with public and private subnets, you can:

1. Launch EC2 instances in your public or private subnets.
2. Deploy load balancers to distribute traffic across multiple instances.
3. Set up Auto Scaling groups for high availability and scalability.
4. Configure RDS databases in your private subnets.
5. Implement VPC peering to connect with other VPCs.
6. Set up VPN connections to connect your VPC with your on-premises network.

Create a VPC with private subnets and NAT gateways using AWS CLI

This tutorial demonstrates how to create a VPC that you can use for servers in a production environment using the AWS CLI. To improve resiliency, you'll deploy servers in two Availability Zones, using an Auto Scaling group and an Application Load Balancer. For additional security, you'll deploy the servers in private subnets. The servers will receive requests through the load balancer and can connect to the internet using NAT gateways. To improve resiliency, you'll deploy a NAT gateway in each Availability Zone.

The following diagram provides an overview of the resources included in this tutorial. The VPC has public subnets and private subnets in two Availability Zones. Each public subnet contains a NAT gateway and a load balancer node. The servers run in the private subnets, are launched and terminated by using an Auto Scaling group, and receive traffic from the load balancer. The servers can connect to the internet by using the NAT gateway. The servers can connect to Amazon S3 by using a gateway VPC endpoint.



Prerequisites

Before you begin this tutorial, you need:

- The AWS CLI installed and configured with permissions to create VPC resources, EC2 instances, load balancers, and Auto Scaling groups. For information about installing the AWS CLI, see [Installing or updating the latest version of the AWS CLI](#).
- Basic knowledge of VPC concepts, including subnets, route tables, and internet gateways.
- The `jq` command-line JSON processor installed. This is used to parse the output of AWS CLI commands. For information about installing jq, see [Download jq](#).
- Sufficient service quotas for the resources you'll create, including:
- At least 2 available Elastic IP addresses

- At least 2 NAT gateways
- At least 1 VPC
- At least 4 subnets
- At least 1 Application Load Balancer

Estimated cost: The resources created in this tutorial will incur charges in your AWS account: *NAT Gateways: ~$0.045 per hour, plus data processing charges* Elastic IP addresses: Free when associated with running instances, ~$0.005 per hour when not associated *EC2 instances: Varies by instance type (t3.micro used in this tutorial)* Application Load Balancer: ~$0.0225 per hour, plus data processing charges

Create the VPC and subnets

First, you'll create a VPC with a CIDR block of 10.0.0.0/16, which provides up to 65,536 private IP addresses.

```
# Create a VPC with CIDR block 10.0.0.0/16
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 'ResourceType=vpc,Tags=[{Key=Name,Value=ProductionVPC}]'
```

The command returns output similar to the following:

```
{
    "Vpc": {
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-abcd1234",
        "State": "pending",
        "VpcId": "vpc-abcd1234",
        "OwnerId": "123456789012",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-abcd1234",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
```

```
        ],
        "IsDefault": false,
        "Tags": [
            {
                "Key": "Name",
                "Value": "ProductionVPC"
            }
        ]
    }
}
```

Note the VPC ID from the output (for example, `vpc-abcd1234`). You'll use this ID in subsequent commands.

Next, you'll identify two Availability Zones in your region to create a resilient architecture.

```
# Get available Availability Zones
aws ec2 describe-availability-zones --query 'AvailabilityZones[0:2].ZoneName' --output
 text
```

The command returns output similar to the following:

```
us-east-1a us-east-1b
```

Now, create four subnets: two public subnets for the load balancer and NAT gateways, and two private subnets for your application servers. Replace `vpc-abcd1234` with your actual VPC ID, and `us-east-1a` and `us-east-1b` with your actual Availability Zones.

```
# Create public subnet in first AZ
aws ec2 create-subnet \
  --vpc-id vpc-abcd1234 \
  --cidr-block 10.0.0.0/24 \
  --availability-zone us-east-1a \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet1}]'

# Create private subnet in first AZ
aws ec2 create-subnet \
  --vpc-id vpc-abcd1234 \
  --cidr-block 10.0.1.0/24 \
  --availability-zone us-east-1a \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet1}]'
```

```
# Create public subnet in second AZ
aws ec2 create-subnet \
  --vpc-id vpc-abcd1234 \
  --cidr-block 10.0.2.0/24 \
  --availability-zone us-east-1b \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet2}]'

# Create private subnet in second AZ
aws ec2 create-subnet \
  --vpc-id vpc-abcd1234 \
  --cidr-block 10.0.3.0/24 \
  --availability-zone us-east-1b \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet2}]'
```

Each command returns output containing the subnet ID. Note these IDs for use in subsequent commands:

- Public Subnet 1: `subnet-abcd1234`
- Private Subnet 1: `subnet-abcd5678`
- Public Subnet 2: `subnet-efgh1234`
- Private Subnet 2: `subnet-efgh5678`

Create and configure internet connectivity

In this section, you'll create an internet gateway to allow communication between your VPC and the internet, and attach it to your VPC.

```
# Create an Internet Gateway
aws ec2 create-internet-gateway --tag-specifications 'ResourceType=internet-
gateway,Tags=[{Key=Name,Value=ProductionIGW}]'
```

The command returns output containing the Internet Gateway ID. Note this ID (for example, `igw-abcd1234`).

Attach the Internet Gateway to your VPC. Replace `igw-abcd1234` with your actual Internet Gateway ID and `vpc-abcd1234` with your actual VPC ID.

```
# Attach the Internet Gateway to the VPC
```

```
aws ec2 attach-internet-gateway --internet-gateway-id igw-abcd1234 --vpc-id vpc-
abcd1234
```

Next, create route tables for your public and private subnets. Replace `vpc-abcd1234` with your actual VPC ID.

```
# Create a route table for public subnets
aws ec2 create-route-table --vpc-id vpc-abcd1234 --tag-specifications
 'ResourceType=route-table,Tags=[{Key=Name,Value=PublicRouteTable}]'

# Create route table for private subnet in first AZ
aws ec2 create-route-table --vpc-id vpc-abcd1234 --tag-specifications
 'ResourceType=route-table,Tags=[{Key=Name,Value=PrivateRouteTable1}]'

# Create route table for private subnet in second AZ
aws ec2 create-route-table --vpc-id vpc-abcd1234 --tag-specifications
 'ResourceType=route-table,Tags=[{Key=Name,Value=PrivateRouteTable2}]'
```

Each command returns output containing the route table ID. Note these IDs:

- Public Route Table: `rtb-abcd1234`
- Private Route Table 1: `rtb-efgh1234`
- Private Route Table 2: `rtb-ijkl1234`

Add a route to the Internet Gateway in the public route table to enable internet access. Replace `rtb-abcd1234` with your actual public route table ID and `igw-abcd1234` with your actual Internet Gateway ID.

```
# Add a route to the Internet Gateway
aws ec2 create-route --route-table-id rtb-abcd1234 --destination-cidr-block 0.0.0.0/0
 --gateway-id igw-abcd1234
```

Associate the subnets with their respective route tables. Replace the route table IDs and subnet IDs with your actual IDs.

```
# Associate public subnets with the public route table
aws ec2 associate-route-table --route-table-id rtb-abcd1234 --subnet-id subnet-abcd1234
aws ec2 associate-route-table --route-table-id rtb-abcd1234 --subnet-id subnet-efgh1234

# Associate private subnets with their respective route tables
```

```
aws ec2 associate-route-table --route-table-id rtb-efgh1234 --subnet-id subnet-abcd5678
aws ec2 associate-route-table --route-table-id rtb-ijkl1234 --subnet-id subnet-efgh5678
```

Create NAT gateways

NAT gateways allow instances in private subnets to connect to the internet or other AWS services, but prevent the internet from initiating connections with those instances. First, allocate Elastic IP addresses for your NAT gateways.

```
# Allocate Elastic IP for NAT Gateway in first AZ
aws ec2 allocate-address --domain vpc --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT1-EIP}]'

# Allocate Elastic IP for NAT Gateway in second AZ
aws ec2 allocate-address --domain vpc --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT2-EIP}]'
```

Each command returns output containing the allocation ID. Note these IDs:

- EIP 1 Allocation ID: `eipalloc-abcd1234`
- EIP 2 Allocation ID: `eipalloc-efgh1234`

Create NAT Gateways in each public subnet. Replace the subnet IDs and allocation IDs with your actual IDs.

```
# Create NAT Gateway in public subnet of first AZ
aws ec2 create-nat-gateway \
  --subnet-id subnet-abcd1234 \
  --allocation-id eipalloc-abcd1234 \
  --tag-specifications 'ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway1}]'

# Create NAT Gateway in public subnet of second AZ
aws ec2 create-nat-gateway \
  --subnet-id subnet-efgh1234 \
  --allocation-id eipalloc-efgh1234 \
  --tag-specifications 'ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway2}]'
```

Each command returns output containing the NAT Gateway ID. Note these IDs:

- NAT Gateway 1: `nat-abcd1234`

- NAT Gateway 2: `nat-efgh1234`

NAT Gateways take a few minutes to provision. Wait for them to be available before proceeding. Replace the NAT Gateway IDs with your actual IDs.

```
# Wait for NAT Gateways to be available
aws ec2 wait nat-gateway-available --nat-gateway-ids nat-abcd1234
aws ec2 wait nat-gateway-available --nat-gateway-ids nat-efgh1234
```

Add routes to the NAT Gateways in the private route tables to enable internet access for instances in private subnets. Replace the route table IDs and NAT Gateway IDs with your actual IDs.

```
# Add route to NAT Gateway 1 in private route table 1
aws ec2 create-route \
  --route-table-id rtb-efgh1234 \
  --destination-cidr-block 0.0.0.0/0 \
  --nat-gateway-id nat-abcd1234

# Add route to NAT Gateway 2 in private route table 2
aws ec2 create-route \
  --route-table-id rtb-ijkl1234 \
  --destination-cidr-block 0.0.0.0/0 \
  --nat-gateway-id nat-efgh1234
```

Create a VPC endpoint for Amazon S3

A VPC endpoint for Amazon S3 allows instances in your private subnets to access S3 without going through the NAT Gateway, which reduces data transfer costs and provides better network performance. Replace `vpc-abcd1234` with your actual VPC ID and the route table IDs with your actual IDs.

```
# Get the prefix list ID for S3 in your region
S3_PREFIX_LIST_ID=$(aws ec2 describe-prefix-lists --filters "Name=prefix-
list-name,Values=com.amazonaws.$(aws configure get region).s3" --query
 'PrefixLists[0].PrefixListId' --output text)

# Create the VPC endpoint for S3
aws ec2 create-vpc-endpoint \
  --vpc-id vpc-abcd1234 \
  --service-name com.amazonaws.$(aws configure get region).s3 \
```

```
  --route-table-ids rtb-efgh1234 rtb-ijkl1234 \
  --tag-specifications 'ResourceType=vpc-endpoint,Tags=[{Key=Name,Value=S3-Endpoint}]'
```

The command returns output containing the VPC endpoint ID. Note this ID (for example, `vpce-abcd1234`).

Configure security groups

Security groups act as virtual firewalls for your instances to control inbound and outbound traffic. Create a security group for the load balancer that allows inbound HTTP traffic from anywhere. Replace `vpc-abcd1234` with your actual VPC ID.

```
# Create security group for the load balancer
aws ec2 create-security-group \
  --group-name LoadBalancerSG \
  --description "Security group for the load balancer" \
  --vpc-id vpc-abcd1234 \
  --tag-specifications 'ResourceType=security-
group,Tags=[{Key=Name,Value=LoadBalancerSG}]'
```

The command returns output containing the security group ID. Note this ID (for example, `sg-abcd1234`).

Allow inbound HTTP traffic to the load balancer. Replace `sg-abcd1234` with your actual load balancer security group ID.

```
# Allow inbound HTTP traffic from anywhere
aws ec2 authorize-security-group-ingress \
  --group-id sg-abcd1234 \
  --protocol tcp \
  --port 80 \
  --cidr 0.0.0.0/0
```

Create a security group for the application servers that allows inbound traffic only from the load balancer. Replace `vpc-abcd1234` with your actual VPC ID.

```
# Create security group for the application servers
aws ec2 create-security-group \
  --group-name AppServerSG \
  --description "Security group for the application servers" \
  --vpc-id vpc-abcd1234 \
```

```
  --tag-specifications 'ResourceType=security-
group,Tags=[{Key=Name,Value=AppServerSG}]'
```

The command returns output containing the security group ID. Note this ID (for example, `sg-efgh1234`).

Allow inbound HTTP traffic from the load balancer security group to the application servers. Replace `sg-efgh1234` with your actual application server security group ID and `sg-abcd1234` with your actual load balancer security group ID.

```
# Allow inbound HTTP traffic from the load balancer security group
aws ec2 authorize-security-group-ingress \
  --group-id sg-efgh1234 \
  --protocol tcp \
  --port 80 \
  --source-group sg-abcd1234
```

Create a launch template for EC2 instances

A launch template contains the configuration information to launch an instance, such as the AMI ID, instance type, and security groups. First, create a user data script that will be executed when the instance launches.

```
cat > user-data.sh << 'EOF'
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>Hello from $(hostname -f) in $(curl -s http://169.254.169.254/latest/meta-
data/placement/availability-zone)</h1>" > /var/www/html/index.html
EOF
```

Encode the user data script in base64.

```
USER_DATA=$(base64 -w 0 user-data.sh)
```

Get the latest Amazon Linux 2 AMI ID.

```
# Get the latest Amazon Linux 2 AMI ID
```

```
aws ec2 describe-images --owners amazon --filters "Name=name,Values=amzn2-ami-hvm-
*-x86_64-gp2" "Name=state,Values=available" --query 'sort_by(Images, &CreationDate)
[-1].ImageId' --output text
```

Create a launch template with the AMI ID, instance type, security group, and user data. Replace `sg-efgh1234` with your actual application server security group ID and `$AMI_ID` and `$USER_DATA` with the values obtained from the previous commands.

```
# Create a launch template
aws ec2 create-launch-template \
  --launch-template-name AppServerTemplate \
  --version-description "Initial version" \
  --tag-specifications 'ResourceType=launch-
template,Tags=[{Key=Name,Value=AppServerTemplate}]' \
  --launch-template-data '{
    "NetworkInterfaces": [{
      "DeviceIndex": 0,
      "Groups": ["sg-efgh1234"],
      "DeleteOnTermination": true
    }],
    "ImageId": "ami-abcd1234",
    "InstanceType": "t3.micro",
    "UserData":
 "IyEvYmluL2Jhc2gKeXVtIHVwZGF0ZSAteQp5dW0gaW5zdGFsbCAteSBodHRwZApzeXN0ZW1jdGwgc3RhcnQgaHR0cGQKc
+SGVsbG8gZnJvbSAkKGhvc3RuYW1lIC1mKSBpbiAkKGN1cmwgLXMgaHR0cDovLzE2OS4yNTQuMTY5LjI1NC9sYXRlc3QvbW
    "TagSpecifications": [{
      "ResourceType": "instance",
      "Tags": [{
        "Key": "Name",
        "Value": "AppServer"
      }]
    }]
  }'
```

Create a load balancer and target group

A target group routes requests to registered targets, such as EC2 instances, using the protocol and port that you specify. Create a target group for your application servers. Replace `vpc-abcd1234` with your actual VPC ID.

```
# Create a target group
aws elbv2 create-target-group \
```

```
  --name AppTargetGroup \
  --protocol HTTP \
  --port 80 \
  --vpc-id vpc-abcd1234 \
  --target-type instance \
  --health-check-protocol HTTP \
  --health-check-path / \
  --health-check-port traffic-port
```

The command returns output containing the target group ARN. Note this ARN (for example, `arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/AppTargetGroup/abcd1234`).

Create an Application Load Balancer in the public subnets. Replace the subnet IDs and security group ID with your actual IDs.

```
# Create a load balancer
aws elbv2 create-load-balancer \
  --name AppLoadBalancer \
  --subnets subnet-abcd1234 subnet-efgh1234 \
  --security-groups sg-abcd1234 \
  --tags Key=Name,Value=AppLoadBalancer
```

The command returns output containing the load balancer ARN. Note this ARN (for example, `arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/AppLoadBalancer/abcd1234`).

Wait for the load balancer to be active before proceeding. Replace `arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/AppLoadBalancer/abcd1234` with your actual load balancer ARN.

```
# Wait for load balancer to be active
aws elbv2 wait load-balancer-available \
  --load-balancer-arns arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/AppLoadBalancer/abcd1234
```

Create a listener for the load balancer that forwards HTTP traffic to the target group. Replace the load balancer ARN and target group ARN with your actual ARNs.

```
# Create a listener
```

```
aws elbv2 create-listener \
  --load-balancer-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/AppLoadBalancer/abcd1234 \
  --protocol HTTP \
  --port 80 \
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/AppTargetGroup/abcd1234
```

Create an Auto Scaling group

An Auto Scaling group contains a collection of EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management. Create an Auto Scaling group that uses the launch template and places instances in the private subnets. Replace the subnet IDs and target group ARN with your actual IDs and ARN.

```
# Create an Auto Scaling group
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name AppAutoScalingGroup \
  --launch-template LaunchTemplateName=AppServerTemplate,Version='$Latest' \
  --min-size 2 \
  --max-size 4 \
  --desired-capacity 2 \
  --vpc-zone-identifier "subnet-abcd5678,subnet-efgh5678" \
  --target-group-arns arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
AppTargetGroup/abcd1234 \
  --health-check-type ELB \
  --health-check-grace-period 300 \
  --tags Key=Name,Value=AppServer,PropagateAtLaunch=true
```

Test your configuration

After the Auto Scaling group launches instances and they pass health checks, you can test your load balancer. Get the DNS name of the load balancer. Replace the load balancer ARN with your actual ARN.

```
# Get the DNS name of the load balancer
aws elbv2 describe-load-balancers \
  --load-balancer-arns arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/AppLoadBalancer/abcd1234 \
  --query 'LoadBalancers[0].DNSName' \
  --output text)
```

Use curl to test the application with the load balancer name.

```
curl http://LoadBalancerName
```

If you refresh the page multiple times, you should see responses from different instances in different Availability Zones.

Clean up resources

When you're finished with this tutorial, you should delete all the resources to avoid incurring charges. Replace all IDs with your actual resource IDs.

```
# Delete the Auto Scaling group
aws autoscaling delete-auto-scaling-group \
  --auto-scaling-group-name AppAutoScalingGroup \
  --force-delete

# Wait for the Auto Scaling group to be deleted
sleep 60

# Delete the load balancer
aws elbv2 delete-load-balancer \
  --load-balancer-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/AppLoadBalancer/abcd1234

# Wait for the load balancer to be deleted
sleep 30

# Delete the target group
aws elbv2 delete-target-group \
  --target-group-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
AppTargetGroup/abcd1234

# Delete the launch template
aws ec2 delete-launch-template \
  --launch-template-name AppServerTemplate

# Delete the NAT Gateways
aws ec2 delete-nat-gateway --nat-gateway-id nat-abcd1234
aws ec2 delete-nat-gateway --nat-gateway-id nat-efgh1234

# Wait for the NAT Gateways to be deleted
```

```
sleep 90

# Release the Elastic IPs
aws ec2 release-address --allocation-id eipalloc-abcd1234
aws ec2 release-address --allocation-id eipalloc-efgh1234

# Delete the VPC endpoint
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-abcd1234

# Wait for security group dependencies to clear
sleep 30

# Delete the security groups
aws ec2 delete-security-group --group-id sg-efgh1234
aws ec2 delete-security-group --group-id sg-abcd1234

# Detach the Internet Gateway
aws ec2 detach-internet-gateway --internet-gateway-id igw-abcd1234 --vpc-id vpc-
abcd1234

# Delete the Internet Gateway
aws ec2 delete-internet-gateway --internet-gateway-id igw-abcd1234

# Delete the route tables
aws ec2 delete-route-table --route-table-id rtb-efgh1234
aws ec2 delete-route-table --route-table-id rtb-ijkl1234
aws ec2 delete-route-table --route-table-id rtb-abcd1234

# Delete the subnets
aws ec2 delete-subnet --subnet-id subnet-abcd1234
aws ec2 delete-subnet --subnet-id subnet-efgh1234
aws ec2 delete-subnet --subnet-id subnet-abcd5678
aws ec2 delete-subnet --subnet-id subnet-efgh5678

# Delete the VPC
aws ec2 delete-vpc --vpc-id vpc-abcd1234
```

Next steps

Now that you've created a VPC with private subnets and NAT gateways, you might want to explore these related topics:

- Security best practices for your VPC

- [Logging IP traffic using VPC Flow Logs]
- [Auto Scaling group scaling policies]
- [Load balancer target group health checks]

Amazon VPC quotas

The following tables list the quotas, formerly referred to as limits, for Amazon VPC resources for your AWS account. Unless indicated otherwise, these quotas are per Region.

If you request a quota increase that applies per resource, we increase the quota for all resources in the Region.

VPC and subnets

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 hundreds 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	5	Yes (up to 50)	The number of CIDRs you can allocate to a single VPC.
VPC Block Public Access exclusions per account per Region	50	Yes. To request an increase, open a service limit increase case using the AWS Support Center Console.	The number of VPC BPA exclusions you can create in an account.

DNS

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

Elastic IP addresses

Name	Default	Adjustable	Comments
Elastic IP addresses per Region	5	Yes	This quota applies to individual AWS account VPCs and shared VPCs.
Elastic IP addresses per public NAT gateway	2	Yes	You can request a quota increase up to 8.

Gateways

Name	Default	Adjustable	Comments
Egress-only internet gateways per Region	5	Yes	To increase this quota, increase the quota for 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 for VPCs per Region. You can attach only one internet gateway to a VPC at a time.

Name	Default	Adjustable	Comments
NAT gateways per Availability Zone	5	Yes	NAT gateways only count toward your quota in the `pending`, `active`, and `deleting` states.
Private IP address quota per NAT gateway	8	Yes	
Carrier gateways per VPC	1	No	

Customer-managed prefix lists

While the default quotas for customer-managed prefix lists are adjustable, you cannot request an increase using the Service Quotas console. You must open a service limit increase case using the AWS Support Center Console.

Name	Default	Adjustable	Comments
Prefix lists per Region	100	Yes	
Versions per prefix list	1,000	Yes	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	Yes	You can resize a customer-managed prefix list up to 1000. For more information, see Resize a prefix list. 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

Name	Default	Adjustable	Comments
			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	10,000	Yes	This quota applies per resource type that can reference a prefix list. For example, you can have 10,000 references to a prefix list across all of your security groups plus 10,000 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 your 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 quota determines both the maximum number of inbound rules and the maximum number of outbound rules. This quota can be increased up to a maximum of 40 inbound rules and 40 outbound rules (for a total of 80 rules), but network performance might be impacted.

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. This limit is enforced per Availability Zone (AZ). If, for example, the network interfaces are in three AZs, each AZ will have a limit of 5,000 limit and the Region will have a limit of 15,000.

Route tables

Name	Default	Adjustable	Comments
Route tables per VPC	200	Yes	The main route table counts toward this quota. Note that if you request a quota increase for route tables, you may also want to request a quota increase for subnets. While route tables can be shared with multiple subnets, a subnet can only be associated with a single route table.
Routes per route table (non-propagated routes)	500	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.

Name	Default	Adjustable	Comments
			If you have more than 125 routes, we recommend that you paginate calls to describe your route tables for better performance.
Propagated routes per route table	100	No	If you require additional prefixes, advertise a default route.

Route servers

Name	Default	Adjustable	Comments
Route servers per VPC	5	Yes. To request an increase, [open a service limit increase case](#) using the AWS Support Center Console.	
Route server endpoints per route server	10	Yes. To request an increase, [open a](#)	

Name	Default	Adjustable	Comments
		service limit increase case using the AWS Support Center Console.	
Peering sessions per network interface	20	Yes. To request an increase, open a service limit increase case using the AWS Support Center Console.	
Route server endpoints per route server and subnet	2	No	You can only have two endpoints in the same subnet for the same route server for redundancy.
Routes per route server peer	100	No	This is the number of routes that can be dynamically advertised over a route server peer

Name	Default	Adjustable	Comments
Routes per route server	100	No	This is the number of routes that can be installed in the Forwarding Information Base (FIB) of a route server.

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	This quota is enforced separately for inbound and outbound rules. For an account with the default quota of 60 rules, a security group can have 60 inbound rules and 60 outbound rules. In addition, this quota is enforced separately for IPv4 rules and IPv6 rules. For an account with the default quota of 60 rules, a security group can have 60 inbound rules for IPv4 traffic and 60 inbound rules for IPv6 traffic. For more information, see the section called “Security group size”.

Name	Default	Adjustable	Comments
			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.
Security groups per network interface	5	Yes (up to 16)	This quota multiplied by the quota for rules per security group cannot exceed 1,000.

VPC subnet sharing

All standard VPC quotas apply to shared VPC subnets.

Name	Default	Adjustable	Comments
Participant accounts per VPC	100	Yes	The maximum number of distinct participant accounts that subnets in a VPC can be shared with. This is a per VPC quota and applies across all the subnets shared in a VPC. VPC owners can view the network interfaces and security groups that are attached to the participant resources.
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.

Network Address Usage

Network Address Usage (NAU) is comprised of IP addresses, network interfaces, and CIDRs in managed prefix lists. NAU is a metric applied to resources in a VPC to help you plan for and monitor the size of your VPC. For more information, see Network Address Usage.

The resources that make up the NAU count have their own individual service quotas. Even if a VPC has NAU capacity available, you won't be able to launch resources into the VPC if the resources have exceeded their service quotas.

Name	Default	Adjustable	Comments
Network Address Usage	64,000	Yes (up to to 256,000)	The maximum number of NAU units per VPC.
Peered Network Address Usage	128,000	Yes (up to 512,000)	The maximum number of NAU units for a VPC and all of its intra-Region peered VPCs. VPCs that are peered across different Regions do not contribute to this number.

Amazon EC2 API throttling

For information about Amazon EC2 throttling, see Request throttling in the *Amazon EC2 Developer Guide*.

Additional quota resources

For more information, see the following:

- AWS Client VPN quotas in the *AWS Client VPN Administrator Guide*
- AWS Direct Connect quotas in the *AWS Direct Connect User Guide*
- Peering quotas in the *Amazon VPC Peering Guide*
- PrivateLink quotas in the *AWS PrivateLink Guide*

- Site-to-Site VPN quotas in the *AWS Site-to-Site VPN User Guide*
- Traffic Mirroring quotas in the *Amazon VPC Traffic Mirroring Guide*
- Transit gateway quotas in the *Amazon VPC Transit Gateways Guide*

Document history

The following table describes the important changes in each release of the *Amazon VPC User Guide*.

Change	Description	Date
Route inbound VPC traffic to public IP addresses	You can now configure advanced routing rules to direct inbound traffic from the internet to specific public IP addresses within your VPC, enabling more granular control over traffic flow and routing decisions for ingress scenarios.	August 13, 2025
Dynamic routing in your VPC using Amazon VPC Route Server	Amazon VPC Route Server simplifies routing for traffic between workloads that are deployed within a VPC and its internet gateways. With this feature, VPC Route Server dynamically updates VPC and gateway route tables with your preferred IPv4 or IPv6 routes to achieve routing fault tolerance for those workloads. This enables you to automatically reroute traffic within a VPC, which increases the manageability of VPC routing and interoper ability with third-party workloads.	March 31, 2025
AWS managed policy update	Amazon VPC updated the AmazonVPCFullAccess and	December 9, 2024

	AmazonVPCReadOnlyAccess managed policies.	
Declarative policy support for VPC BPA	If you are using AWS Organizations to manage accounts in your organization, you can use a declarative policy to enforce VPC BPA on the accounts in the organization.	December 1, 2024
VPC Block Public Access (BPA)	VPC Block public Access (BPA) enables you to block resources in VPCs and subnets that you own in a Region from reaching or being reached from the internet through internet gateways and egress-only internet gateways.	November 19, 2024
Shared Security Groups	This feature enables you to share a security group with other AWS Organizations accounts.	October 30, 2024
Security Group VPC Associations	This feature enables you to associate a security group with multiple VPCs in the same Region.	October 30, 2024
NAT gateway MTU support	NAT gateways support traffic with a maximum transmission unit (MTU) of 8500.	September 10, 2024

Private IPv6 addressing	Information about private IPv6 addressing was added. Private IPv6 addresses are only available in Amazon VPC IP Address Manager.	August 8, 2024
IPv6 preferred lease time	You can now choose how frequently a running insta nce with an IPv6 assigned to it goes through DHCPv6 lease renewal.	February 20, 2024
Guide structure review and improvements	The structure of the guide was reviewed and improveme nts were made to improve the customer experience related to finding info for specific scenarios.	February 20, 2024
AWS managed policy update	Amazon VPC updated the AmazonVPCFullAccess and AmazonVPCReadOnlyAccess managed policies.	February 8, 2024
AWS managed policy update	Amazon VPC updated the AmazonVPCCrossAcco untNetworkInterfaceOperatio ns managed policy.	September 25, 2023

EC2-Classic is deprecated	With EC2-Classic, EC2 instances ran in a single, flat network shared with other customers. Amazon VPC replaces EC2-Classic. With Amazon VPC, your instances run in a virtual private cloud (VPC) that's logically isolated to your AWS account.	July 31, 2023
Add secondary IPv4 addresses to NAT gateways	You can add secondary private IPv4 addresses to public and private NAT gateways. Secondary IPv 4 addresses increase the number of available ports, and therefore they increase the limit on the number of concurrent connections that your workloads can establish using a NAT gateway.	January 31, 2023
Aligning with IAM best practices	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM.	January 4, 2023
Pick the private IP address of your NAT gateway	When you create a NAT gateway, you can now choose to pick the private IP address that's assigned to the NAT gateway. Previousl y, the private IP address was automatically assigned from the IP address range of the subnet.	November 17, 2022

IPv6 default gateway router configuration	Three IPv6 addresses are now reserved for use by the default VPC router.	November 11, 2022
Transfer Elastic IP addresses	You can now transfer Elastic IP addresses from one AWS account to another.	October 31, 2022
Network Address Usage metrics	You can enable Network Address Usage metrics for your VPC to help you plan for and monitor the size of your VPC.	October 4, 2022
Publish Flow Logs to Amazon Data Firehose	You can specify a Amazon Data Firehose delivery stream as a destination for flow log data.	September 8, 2022
NAT gateway bandwidth	NAT gateways now support bandwidth up to 100 Gbps (an increase from 45 Gbps) and can process up to ten million packets per second (up from four million packets).	June 15, 2022
Multiple IPv6 CIDR blocks	You can associate up to five IPv6 CIDR blocks to a VPC.	May 12, 2022
Reorganization	General reorganization of this Amazon Virtual Private Cloud User Guide.	January 2, 2022
NAT gateway IPv6 to IPv4	NAT gateway supports network address translation from IPv6 to IPv4, popularly known as NAT64.	November 24, 2021

IPv6-only subnets in VPCs	You can create IPv6-only subnets into which you can launch IPv6-only EC2 instances.	November 23, 2021
VPC Flow Logs delivery options to Amazon S3	You can specify the Apache Parquet log file format, hourly partitions, and Hive-compatible S3 prefixes.	October 13, 2021
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	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	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	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
Tag on create	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
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
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
VPC Sharing	You can share subnets that are in the same VPC with multiple accounts in the same AWS organization.	November 27, 2018
Create default subnet	You can create a default subnet in an Availability Zone that does not have one.	November 9, 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

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	The Amazon DNS server can now resolve private DNS hostnames to private IP addresses for all address spaces.	October 24, 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

ClassicLink	You can use ClassicLink to link your EC2-Classic instance to a VPC in your account. You can associate VPC security groups with the EC2-Classic instance, enabling communica tion between your EC2-Class ic 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
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	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	Dedicated Instances are Amazon EC2 instances launched within your VPC that run hardware dedicated to a single customer.	March 27, 2011