IPv6 on AWS

Best practices for adopting and designing IPv6-based networks on AWS

First Published October 26, 2021

Last Updated January 06, 2022

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

Every node connected to an Internet Protocol (IP) network must have an IP address for communication purposes. As the internet continues to grow, so does the need for IP addresses. IPv6 is a version of the Internet Protocol that uses a larger address space than its predecessor, IPv4. This whitepaper focuses on best practices for adopting and designing IPv6-based networks on the Amazon Web Services (AWS) Cloud. It covers IPv6 dual-stack networks for both internet-facing and private IPv6 networks use cases.

Introduction

An increasing number of organizations operate dual-stack IPv4/IPv6 networks. Carrier networks and ISPs were the first groups to start deploying IPv6 on their networks, with mobile networks leading the charge. Adoption of IPv6 has been delayed, partly due to network address translation (NAT) with IPv4, which takes private IP addresses and turns them into public IP addresses.

An increasing number of organizations are adopting IPv6 in their environments, driven by the public IPv4 space exhaustion, private IPv4 scarcity, especially within large-scale networks, and the need to provide connectivity to IPv6-only clients. There is no one-size-fits-all approach with IPv6; however, there are best practices AWS customers can follow to plan and implement IPv6 into their existing cloud networks.

This whitepaper explains key drivers for adoption and highlights best practices to guide you while leaving enough space for you to decide, based on your specific use case and implementation, how to approach IPv6 in your network.

Tenets for IPv6 adoption

Adhering to the following tenets for IPv6 adoption can help make the process and decision more manageable:

- **Re-evaluate network controls** — IPv6 offers opportunities to rethink your approach to perimeter security and make design decisions that further improve your security posture.
- **Design for scale** — More usable IPv6 addresses doesn't mean you can shortcut IP allocation and planning.
- **Phase your IPv6 adoption** — Focus on your business needs to implement IPv6 where needed, and remember that IPv4 and IPv6 can be made to coexist as long as needed.

Intended audience

Considering the variety of drivers, both internal and external, IPv6 adoption affects organizations regardless of size, industry sector, or geography. The best practices and design considerations outlined in this paper are oriented toward cloud architects and network technical leads who are responsible for the overall network design and architecture of their AWS environment. Whether you are new to AWS or you have already been using AWS for years, your team will benefit from reviewing this paper and deciding what is applicable based on your requirements and your current AWS environment.

> **Note:** This paper doesn’t repeat the guidance found in other resources, but rather complements them by focusing on the IPv6-specific differences. Customers are advised to consult the guidance in this paper in addition to the existing literature that covers individual topics in detail.

Internet Protocol version 6

IPv6 is the next generation IP address standard intended to supplement and eventually replace IPv4, the original and ubiquitous IP address scheme. Every computer, mobile phone, home automation component, IoT sensor, and any other device connected to an IP-based network needs a numerical IP address to communicate with other devices.

Public reachable IPv4 addresses are in short supply due to their widespread usage and constantly increasing demand stemming from the proliferation of connected devices globally. The last available block of new IP version 4 (IPv4) addresses was allocated back in 2011, and from that time on, everyone has been reusing a finite set of available addresses. IP version 6 (IPv6) is the replacement for IPv4, and it is designed to address the depletion of IP addresses, and change the way traffic is managed.

Brief IPv6 overview

The Internet Layer of the TCP/IP model aligns with the Layer 3 (Network) of the Open Systems Interconnection model (OSI model) and is used for transmitting data between the two nodes communicating over the IP. The IPv4 addressing scheme, with a 32-bit address field, provides around 4,000,000,000 possible addresses, which cannot fulfill the task of addressing the rapidly increasing number of the hosts on the internet.

IPv6 offers the following significant features:

- A larger address space — 128-bit long addresses vs 32-bit long addresses in IPv4
- A large block of globally unique and hierarchically assigned addresses, based on prefixes rather than address classes, to keep routing tables small and backbone routing efficient

- A mechanism for the auto-configuration of network interfaces
- Support for encapsulation of itself and other protocols
- A class of service that distinguishes types of data
- Improved multicast routing capabilities (in preference to broadcasting)
- Built-in authentication and encryption
- Methods to adopt IPv6 and maintain interoperability with IPv4

IPv6 uses the term node for any system that runs IPv6; that is, a host or a router. An IPv6 host is a node that does not forward IPv6 packets that are not explicitly addressed to it. A router is a node that forwards IP packets not addressed to it. Some considerations regarding the main differences between IPv4 and IPv6 follow:

- **Headers** — IPv6 uses two distinct types of headers:
  - Main/regular IPv6 header
  - IPv6 extension headers

  The main IPv6 header is equivalent to the basic IPv4 version, despite some field differences that are the result of lessons learned from operating IPv4. IPv6 has a fixed 40-byte header that allows for faster processing of IP datagram.
- **Flow labeling** — Flow labeling is a new field used by a node to label packets of a flow. A flow label of '0' (zero) is used to indicate that packets have not been labeled. Packet classifiers can use the flow label, source address, and destination address fields to identify to which flow the packet belongs. Packets are processed in a flow-specific way by the nodes that are able to do it in a stateless manner or that have been set up with a flow-specific state.
- **Address assignment** — IPv6 address can be assigned statically, using DHCPv6 and through Stateless Address Auto-Config (SLAAC), which makes use of the Media Access Control (MAC) address of the network interface to derive an IPv6 address for the node.
- **Error correction** — IPv6 does not support header checksum capability, as it considered redundant, being processed both at the lower layer protocol (Ethernet) and upper layer protocols (TCP/UDP).

- **Neighbor Discovery Protocol** — IPv6 defines Internet Control Message Protocol v6 (ICMPv6) messages to take the place of IPv4's ARP and other network discovery functions. It provides many improvements over IPv4, which includes Neighbor Unreachability Detection (NUD), improving packet delivery in the presence of failing routers, links, or mobile nodes.

IPv6 addressing

The IPv6 address space is organized by using format prefixes, that logically divide it in the form of a tree so that a route from one network to another can easily be found.

The main categories of IPv6 addresses are:

- Aggregatable global unicast addresses (GUA) — `2000::/3`
- Unique-local unicast addresses (ULA) — `FC00::/7`
- Link-local unicast addresses — `FE80::/10`
- Multicast addresses — `FF00::/8`

Note that the rest of the IPv6 addresses are reserved by the IETF (Internet Engineering Task Force) and may be used for new technologies or to augment these space allocations in the future.

IPv6 adoption strategies and mechanisms

Working with customers, AWS observed the following two main drivers for IPv6 adoption.

- Network Address Translation is no longer sufficient to work around exhaustion and poses significant challenges with overlapping IP addresses.
- There are numerous organizational or regulatory mandates to adopt IPv6.

Following is a summary of IPv6 adoption drivers:

- **Mandated IPv6 endpoints** — Either mandated by a government policy or an industry regulator, and not necessarily tied to a particular use case.

- **Interoperability with IPv6 networks** — The last years have seen a growing population of IPv6-only clients, and as a result so have the number of organizations wanting to cater to this user base. With the number of mobile IPv6-only users, many companies find they can't afford to lose that section of their potential user base.
- **Public IPv4 exhaustion** — As public IPv4 addresses become more scarce, allocating contiguous IP address ranges for public routing becomes more difficult and costly.
- **Private IPv4 exhaustion** — As private IPv4 (RFC1918) addresses become exhausted and too fragmented within organizations' private networks, IPv6-only networks offer opportunities to address additional nodes.

At the time of this writing, global IPv6 adoption is primarily driven by the major internet providers, network device manufacturers, and organizations that need to grow their number of internet-reachable devices. Beyond these, rate of adoption is significantly slower. Reasons for this include the prevalence of NAT, and proxies used in combination with reusable private IPv4 address space (defined in RFC1918) which greatly reduces the number of unique internet-routable IP addresses. Also, the native lack of backwards compatibility between the two versions of IP protocol present barriers to adoption.

Although your drivers will likely be similar to the needs of other organizations, each organization also has some unique requirements. Accordingly, best practices and design guidance in this paper are intended to offer guidance rather than a one-size-fits-all solution to IPv6 adoption. The design of your IPv6 network on AWS may differ from the examples provided in this document. However, this paper can help you make informed decisions as you embark to adopt IPv6.

IPv6 adoption strategy depends on the driving force behind it. You may have an immediate goal such as addressing private IPv4 exhaustion, or the ability to provide IPv6 service endpoints as per government mandate, with the long-term goal of fully converting to an IPv6 only network. Following are the four main driving forces, and the corresponding adoption strategy:

- **Private IPv4 exhaustion** — The goal is to provision new nodes and allocate routable IP addresses without IP overlap, and without the challenge of sourcing contiguous and usable IP addresses.

Adoption strategy —Configure IPv6-only routing between dual stack network segments, to facilitate communication using the IPv6 stack. Provide IPv6 to IPv4 interoperability layer, such as dual-stack load balancers.

- **Public IPv4 exhaustion** — The goal is to support IPv6 only nodes connecting to your public endpoints. As an example, you may have an API endpoint for data upload from IoT devices which are connected to an IPv6-only network. The IoT devices have IPv6 addresses, and the network does not provide interoperability layer to IPv4. Other devices may operate in IPv4 networks.

  Adoption strategy — Create dual-stack VPCs and subnets. Configure AWS services such as load balancers and edge service in dual-stack mode with corresponding DNS record on the AWS Cloud. Optionally, provide separate endpoints for IPv4 and IPv6 in dedicated IPv4-only or IPv6-only deployments.

- **Interoperability with IPv6-only networks** — The goal is to support connectivity to IPv6-only nodes from your IPv4-only network. This is a less frequent use case, because most endpoints operate in dual-stack mode and provide an interoperability layer.

  Adoption strategy — Create dual-stack VPCs, and use them to host a NAT46 interoperability layer. Full dual-stack adoption is often not practical, so providing an interoperability layer is preferred.

- **Mandated IPv6 endpoints** — The goal is to support interoperability with IPv6-only nodes connecting to your services, and avoid building technical debt in new services. Have a plan and complete transition of older services which do not have a dedicated IPv6 or dual-stack capability.

  Adoption strategy — Create dual-stack VPCs and subnets. Configure AWS services such as load balancers and edge service in dual-stack model with corresponding DNS records on the AWS Cloud. Optionally, provide separate endpoints for IPv4 and IPv6 in dedicated IPv4-only and IPv6-only stacks.

Your organization should be prepared to operate in heterogeneous mode for many years to come. For over 20 years, IPv6 coexisted side-by-side with IPv4, and it will continue to do so. The long-term goal is to be IPv6 everywhere and retire the IPv4 stack, but you don't need to rush it. Instead, work backwards from your business requirements and act accordingly. If you are just starting with IPv6, small steps along with the power of the AWS Cloud will give you the needed confidence to transition more of your network to IPv6-only mode.

Interoperability

Although operating in dual-stack mode solves a lot of the problems with IPv4 and IPv6 interoperability, it creates management overhead. For example, security becomes harder because you have to manage two sets of security rules, one for each network stack. Routing and troubleshooting become harder, and you have to maintain additional records to existing DNS names.

You may be able to avoid making the entire network dual-stack by focusing on implementing dual-stack at your border via load balancers. Existing segments of your network can continue to operate as IPv4 in most cases, and new segments are built with IPv6. Focus on implementing and operating interoperability layer where AWS services such as dual-stack VPC and load balancers, to help you solve interoperability challenges.

Planning IPv6 adoption in the AWS Cloud network

Elastic network interfaces in an IP network could operate in three different modes:

- **IPv4-only mode** — Your resources can communicate over IPv4, and if communicating to IPv6 nodes, require an interoperability layer.
- **IPv6-only mode** — Your resources can communicate over IPv6, and if communicating to IPv4 nodes, require an interoperability layer.
- **Dual-stack mode** — Your resources can communicate over both IPv4 and IPv6. A separate interoperability layer is not required.

IPv6 addressing plan on AWS

Coming up with an IPv6 addressing plan is one of the most important initial tasks for any organization proceeding with IPv6 adoption. For most organizations, IPv6 is deployed in parallel with IPv4 in existing IPv4 AWS and hybrid networks. IPv4 addressing plans tend to grow over time, and consequently may be highly fragmented, not contiguous, or not big enough. Simply duplicating the IPv4 addressing scheme in some fashion in IPv6 might initially prove advantageous. However, any temporary advantage gained by such a shortcut will ultimately be surpassed by the ease and efficiency of operation and design offered by a proper IPv6 addressing plan that incorporates the key benefits of the larger allocations possible with IPv6.

The virtually limitless scale of the IPv6 address space allows for an addressing plan no longer constrained by the scarcity of IPv4 addresses. Techniques like Variable Length

Subnet Masking (VLSM) (previously required in IPv4 to economically match subnet size to host count on a given network segment) can be seen as unnecessary and obsolete in IPv6. Instead, it's possible to adopt a consistent addressing plan by assigning significance to groups of VPCs according to network and segmentation needs.

In AWS, IPv6 address space assigned to VPCs is assigned from the *globally unique unicast* address range. There are two options for IPv6 address assignment:

- AWS-assigned IPv6 VPC classless inter-domain routings (CIDRs)
- Bring your own IPv6 CIDR Blocks (BYOIPv6)

> **Note:** Amazon VPC doesn't support unique local address (ULA) CIDRs. All VPCs must have unique IPv6 CIDR. Two VPCs can't have the same IPv6 CIDR range.

AWS-assigned IPv6 VPC CIDR

By default, Amazon provides one fixed size (/56) IPv6 CIDR block to a VPC. This range is assigned by the service, and consequently, you can't assign contiguous IPv6 CIDR blocks to VPCs in the same Region or based on other custom-defined criteria.

For customers that have a large VPC footprint in AWS and prefer to use IP route summarization to simplify their overall environment, bring your own IPv6 (BYOIPv6) described, in the next section, may be the preferred solution.

BYOIPv6 VPC CIDR

Alternatively, if you own an IPv6 address space, you can import it into AWS using the Bring Your Own IPv6 service. The smallest IPv6 address range that you can bring is /48 for CIDRs that are publicly advertised by AWS, and /56 for CIDRs that are not publicly advertised by AWS. You can also choose to bring a /48 and mark it as non-advertisable, keeping control of IP advertisements on your on-premises setup. After importing it, you can assign /56 ranges from the space to individual VPCs in the same account.

For the process on how to "Bring Your Own IP (BYOIP)," refer to Configure your BYOIP address range.

As summarization can be easily configured with contiguous IPv6 blocks, for multi-Region deployments, you can bring one or more /48 or larger prefixes to each Region. Consider route summarization at the VPC route table level, as well as when using AWS connectivity options such as AWS Transit Gateway, AWS Direct Connect, and AWS VPN.

VPC subnet addressing

Although you can assign one /56 IPv6 CIDR block to a VPC, the VPC subnets are /64 fixed in length. This yields to the interface ID being /64 in length, in accordance with the general format of the IPv6 unicast addresses. Given the fixed size of the VPC CIDR and the subnet prefix, you have 8 bits for subnet allocation in the VPC, enabling you to create 256 subnets in the VPC.

Designing an IPv6 AWS Cloud network

Amazon VPC design

Planning and implementing network connectivity in AWS is usually one of the foundational tasks you perform when deploying workloads in AWS. Following are some of the aspects typically considered:

- Amazon VPC CIDR range and IP address allocation including Bring Your Own IP (BYOIP) for public connectivity

- Number and type of subnets

- Number of availability zones to cover

- Permitted traffic paths

- Internet incoming and outgoing traffic options

- Hybrid connectivity

- Inter-VPC connectivity

- Scalability and expansion

Although most of these aspects apply equally to both IPv4 and IPv6, existing literature is mostly written in the context of IPv4 only. This section augments these existing resources by providing IPv6 specifics. AWS recommends you read the existing, IPv4-centric material first to get the most out of this content.

VPC IP address assignment

Your VPC can operate in dual-stack mode—your resources can communicate over IPv4, IPv6, or both. IPv4 and IPv6 communication are independent of each other. You cannot disable IPv4 support for your VPC and subnets; you are required to allocate at least one IPv4 CIDR range to your VPC. In addition, you may associate up to one IPv6 CIDR block range per VPC.

> **Note:** While you can add multiple IPv4 CIDR ranges to a VPC, the same is not true for IPv6. This is a fixed quota documented in the Quota section of the VPC documentation.

Subnet address assignment

After you have associated an IPv6 prefix to a VPC, you can begin to assign one /64 IPv6 prefix to each subnet. Note that these assignments are configured on a per-subnet basis, and it's possible to have a mix of subnets with and without IPv6 within the same VPC. This is useful in scenarios where you merely require IPv6 capability for a subset of the network as described in the drivers for adoption section.

The address assignment of resource within a subnet occurs at two levels:

- The Amazon VPC elastic network interface construct configuration
- A resource's networking stack configuration

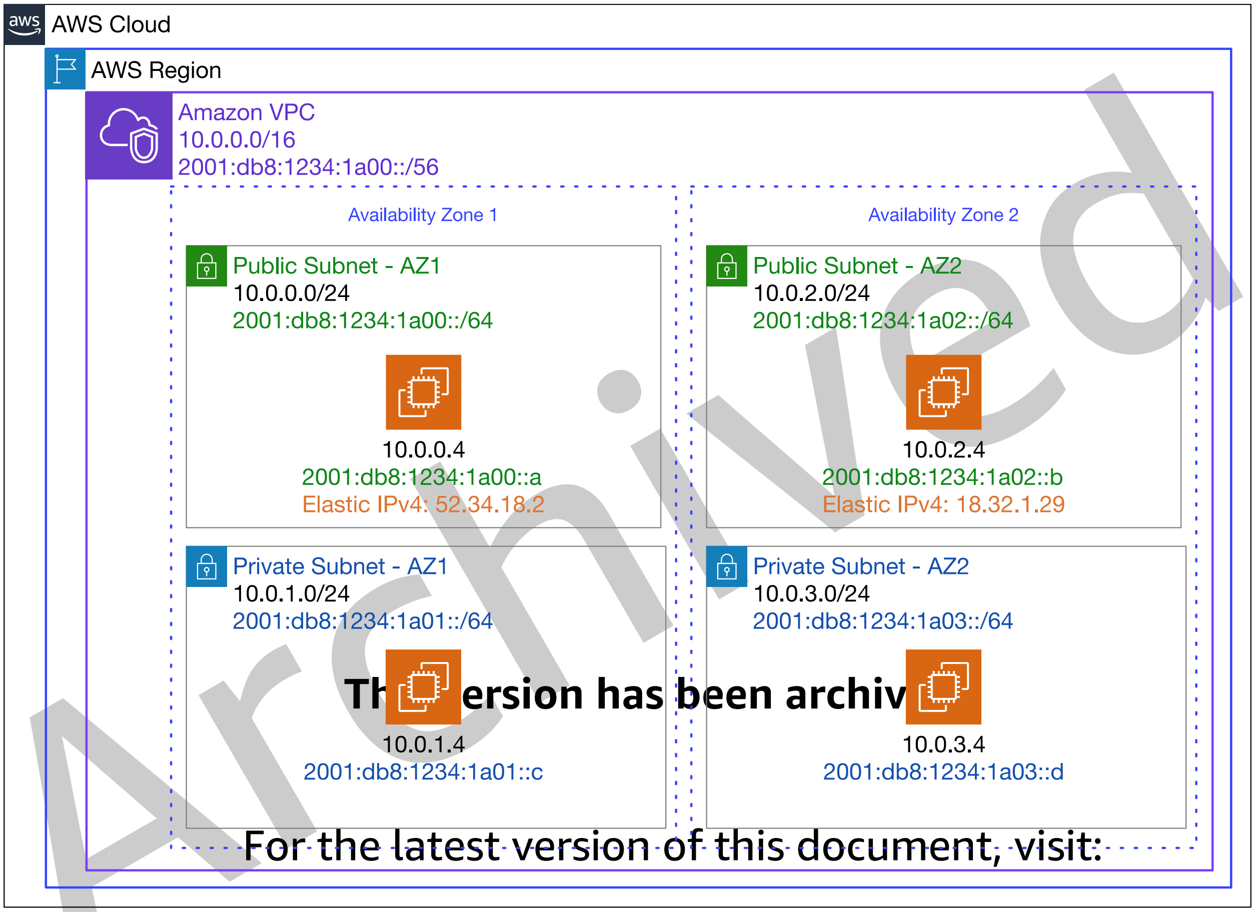
IP addressing of the elastic network interface

Network-addressable resources deployed within a VPC must have an elastic network interface. Examples of resources include:

- Amazon Elastic Compute Cloud (Amazon EC2) instances
- Interface VPC endpoints
- AWS Lambda functions (deployed in VPCs)
- Amazon Relational Database Service (Amazon RDS) database instances

Elastic network interfaces are logical constructs in the VPC which represent a resource's network adapter at runtime. Each elastic network interface may have one or more IPv4 addresses as well as one or more IPv6 addresses. This means you are not required to provision separate elastic network interfaces for IPv4 and IPv6, and there is no need to configure additional elastic network interfaces on your workloads to enable IPv6.

An elastic network interface placed into an IPv6 enabled subnet may be created with and modified to have an IPv6 address assigned. You can configure this behavior per elastic network interface, and you can choose to either have AWS auto assign one for you or specify an unused address in the subnet's allocated range. In either case, the address configured remains constant throughout the elastic network interface's life unless explicitly modified.



Dual-stack Amazon VPC

IP addressing at the resource's networking stack

In IPv4, the preferred method for assigning IPv4 addresses is to use Dynamic Host Configuration Protocol (DHCP). DHCP is based on IPv4's broadcast mechanism that allows hosts to announce themselves to DHCP servers. These servers can then offer an IP address lease to the client. IPv6 has no concept of broadcast (refer to the Brief IPv6 overview section of this document) and initially did not feature DHCP capability.

However, the community has become used to DHCP, and so RFC 8415 was developed to introduce DHCPv6. In the absence of broadcast capability, DHCPv6 makes use of the well-known multicast address for all DHCP servers/relays, `ff02::1:2`.

Amazon VPC has built in support for address assignment via DHCP for both IPv4 and IPv6. Address allocation works similar to static address reservation in traditional DHCP servers: the IP address assigned to the elastic network interface construct determines the IP address the VPC DHCP infrastructure offers the resource requesting an address. Amazon VPC also offers the ability to configure DHCP option sets which can be used to provide additional configuration information such as domain name or DNS servers to use. In a dual-stack design all IP addresses used in an option set need to be IPv4.

> **Note:** Be aware that if you're trying to add IPv6 to an existing or migrated workload, the host OS may not yet be set up to use DHCPv6. Consult the documentation to learn how to enable and verify DHCPv6 operation for your chosen operating system.

> **Note:** The use of DHCP is optional, and you may statically configure the host OS with an IP address. However, the VPC anti-spoofing mechanism enforces the IP address configured on the network interface controller (NIC) to match the one configured on the underlying elastic network interface. AWS recommends enabling DHCP, even for resources that require static IP addresses, and managing static IP assignment at the elastic network interface level.

Supporting Amazon VPC services

AWS exposes a set of supporting services within customer VPCs at well-known/reserved addresses. These services are traditionally exposed from the IPv4 link-local address range (169.254.0.0/16). For AWS Nitro System instances, AWS also provides these services using IPv6 ULAs.

Instance Metadata Service (IMDS)

The instance metadata is information about your instance. Instances can introspect this at runtime by querying the IMDS available to it at 169.254.169.254. For Nitro-based instances, AWS also provides this service at the fd00:ec2::254 IPv6 endpoint.

For more details, refer to Use IMDSv2.

Route 53 DNS resolver

Amazon VPC features a built-in DNS resolver which resides at VPC_CIDR_BASE + 2 and `169.254.169.253`. IPv6 enabled Nitro instances can access the service via fd00:ec2::253. Amazon Route 53 Resolver and DNS in general is discussed at greater length in the Designing DNS for IPv6 section of this document.

Network Time Protocol server

Amazon VPC provides a Stratum-3 NTP server at `169.254.169.123`. Nitro-based IPv6 enabled instances can reach this server via `fd00:ec2::123`.

Amazon VPC connectivity options for IPv6

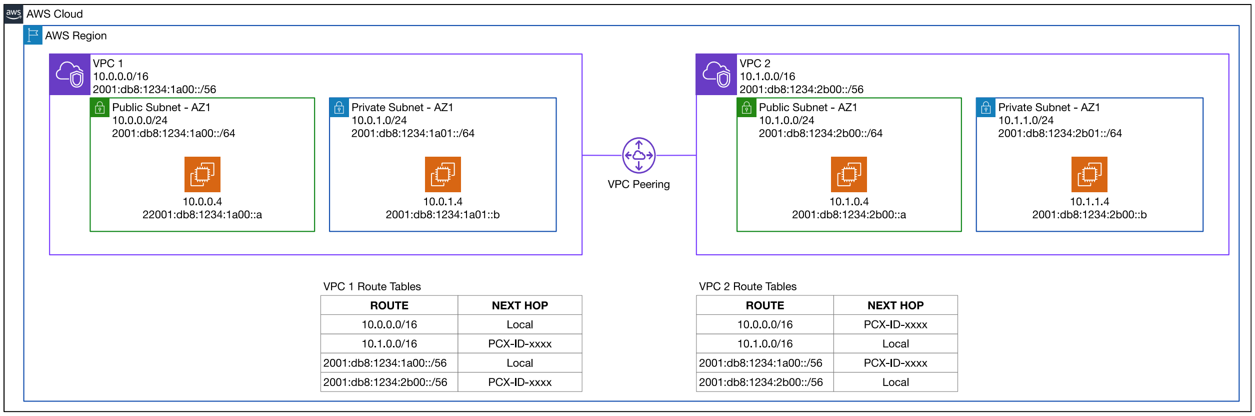
There are a growing number of ways in which Amazon VPCs can connect to each other. Many of these options are detailed in the VPC to VPC connectivity section of the Building a Scalable and Secure Multi-VPC AWS Network Infrastructure whitepaper. AWS recommends you read the following subsections alongside, and it follows the same structure while providing additional insight regarding IPv6 operation as both papers cover:

- VPC peering
- AWS Transit Gateway
- VPC subnet sharing
- AWS PrivateLink

VPC peering

VPC peering is the simplest method for VPC-to-VPC connectivity. It supports both intra- and inter-Region connectivity. The peering itself is IP protocol agnostic. After you establish peering, you must configure one or more static routes defining which prefixes are reachable. Both IPv4 and IPv6 prefixes may be routed across the same peering.

The following diagram depicts a VPC peering between two VPCs supporting IPv4 and IPv6 simultaneously. The peering is agnostic, and the subnet route tables are the deciding factor for which prefixes are reachable.



VPC 1 Route Tables

ROUTE	NEXT HOP
10.0.0.0/16	Local
10.1.0.0/16	PCX-ID-xxxx
2001:db8:1234:1a00::/56	Local
2001:db8:1234:2b00::/56	PCX-ID-xxxx

VPC 2 Route Tables

ROUTE	NEXT HOP
10.0.0.0/16	PCX-ID-xxxx
10.1.0.0/16	Local
2001:db8:1234:1a00::/56	PCX-ID-xxxx
2001:db8:1234:2b00::/56	Local

Dual-stack IPv6 VPC peering

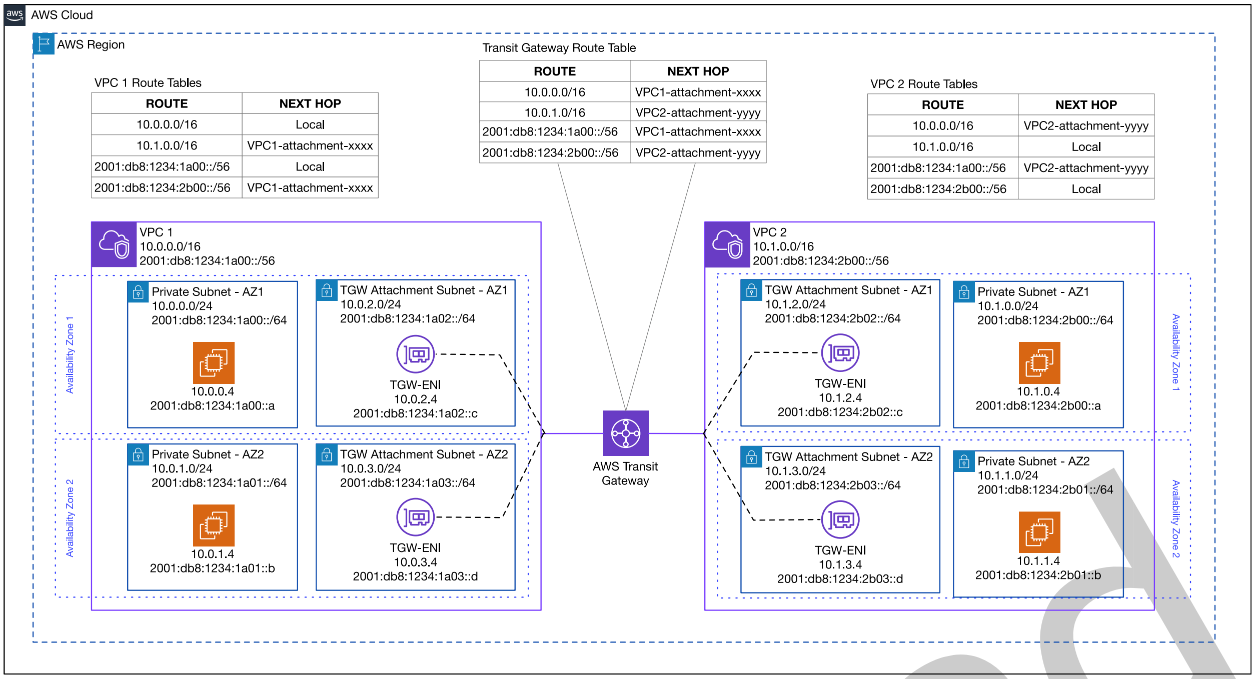
AWS Transit Gateway

AWS Transit Gateway is a scalable highly available way to establish network connectivity between multiple VPCs. A Transit Gateway is a Regional construct, and attaches VPCs within the same Region. Transit Gateways located in different AWS Regions can establish a peering relationship, enabling global connectivity for your network.

IPv6 connectivity into Transit Gateway

You use a Transit Gateway attachment to connect a VPC to a Transit Gateway. An attachment deploys an elastic network interface into each subnet you select. Traffic is routed into Transit Gateways using static routes in VPC subnet routing tables with the attachment as the next-hop. The attachments themselves are IP protocol agnostic, and you can route IPv4 and IPv6 prefixes via the same attachment. To support IPv6, the elastic network interface s used by the attachments need to have IPv6 addresses assigned to them.

> **Note:** If you retrofit IPv6 into an existing VPC with a Transit Gateway attachment, its elastic network interfaces won't be auto-assigned IPv6 addresses; you need to explicitly configure assignment for the elastic network interfaces. If you don't, IPv6 traffic cannot use the attachment.



VPC 1 Route Tables

ROUTE	NEXT HOP
10.0.0.0/16	Local
10.1.0.0/16	VPC1-attachment-xxxx
2001:db8:1234:1a00::/56	Local
2001:db8:1234:2b00::/56	VPC1-attachment-xxxx

Transit Gateway Route Table

ROUTE	NEXT HOP
10.0.0.0/16	VPC1-attachment-xxxx
10.1.0.0/16	VPC2-attachment-yyyy
2001:db8:1234:1a00::/56	VPC1-attachment-xxxx
2001:db8:1234:2b00::/56	VPC2-attachment-yyyy

VPC 2 Route Tables

ROUTE	NEXT HOP
10.0.0.0/16	VPC2-attachment-yyyy
10.1.0.0/16	Local
2001:db8:1234:1a00::/56	VPC2-attachment-yyyy
2001:db8:1234:2b00::/56	Local

Dual-stack AWS Transit Gateway routing

IPv6 traffic within and between Transit Gateways

A Transit Gateway attachment is both a source and a destination of packets. You can attach the following resources to your Transit Gateway:

- VPCs
- One or more VPN connections
- One or more AWS Direct Connect gateways
- One or more Transit Gateway Connect attachments
- One or more Transit Gateway peering connections

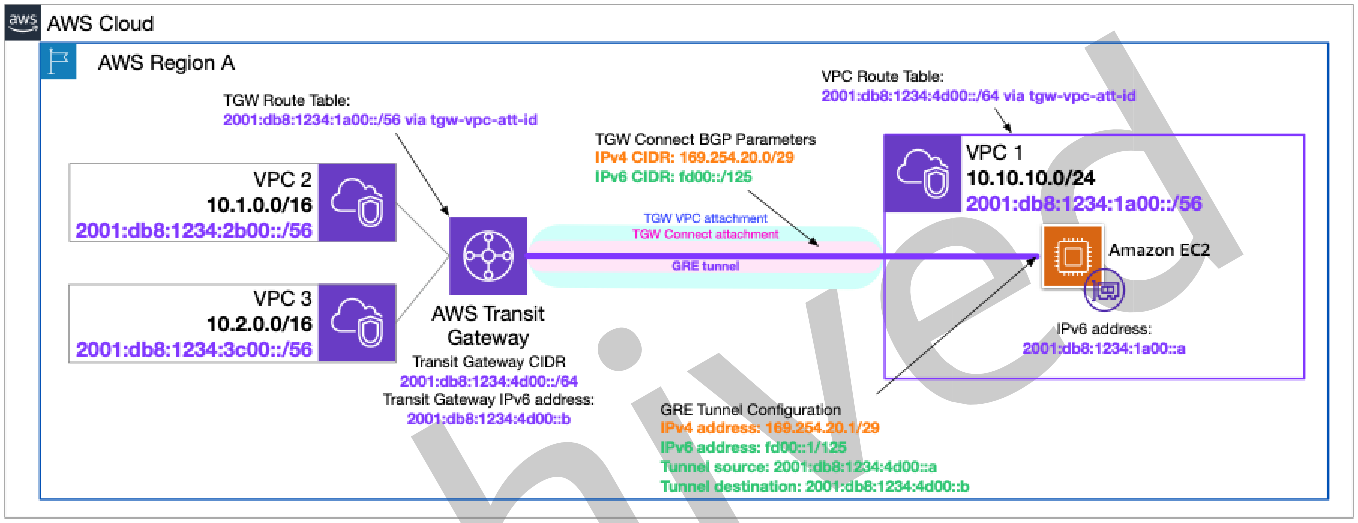
A Transit Gateway has one or more routing tables. A routing table can receive its entries through a combination of static route configuration and dynamic propagations from other attachments (VPC, Direct Connect, Site-to-Site VPN, or Connect Peering). In either case, IPv6 routes are supported.

AWS Transit Gateway Connect Attachments for IPv6

You can create a T*ransit Gateway Connect attachment* to establish a connection and dynamic routing between a transit gateway and third-party virtual appliances (such as SD-WAN appliances).

These attachments take the form of IP Generic Routing Encapsulation (GRE) protocol tunnels and enable dynamic exchange of routing information between an EC2 instances in a VPC and a TGW. Route exchange is facilitated by a Border Gateway Protocol (BGP) peering. TGW connect peers support IPv6 using Multi-Protocol BGP (MP-BGP) and a /125 CIDR block from the well-known `fd00::/8` unique local address range.

Multiprotocol BGP (MP-BGP) is an extension to BGP that enables BGP to carry routing information for multiple network layers and address families. MP-BGP can carry the unicast routes used for multicast routing separately from the routes used for unicast IP forwarding.

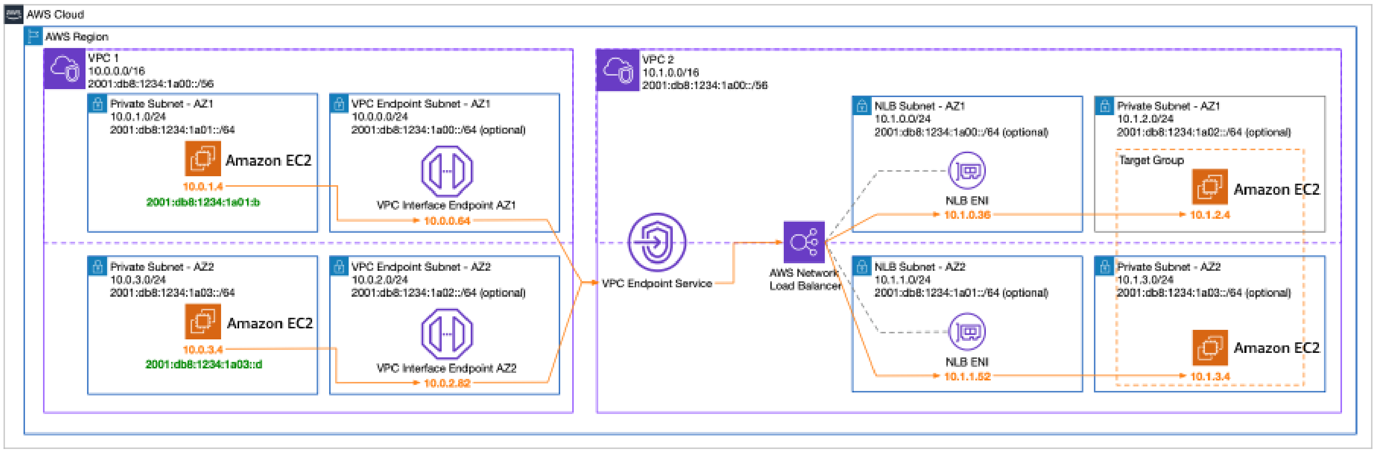


AWS Transit Gateway Connect dual-stack IPv6 routing

AWS PrivateLink

AWS PrivateLink provides private connectivity between VPCs, AWS services, and customer on-premises networks, without exposing traffic to the public internet. AWS PrivateLink makes it easy to connect services across different accounts and VPCs to significantly simplify your network architecture.

AWS PrivateLink does not currently support IPv6. However, PrivateLink has the useful property of abstracting the IP addressing used between source and destination. In the meantime, it’s possible to operate a dual-stack setup for the purpose of communicating via PrivateLink endpoints. It is possible for a workload to use IPv6 for most communication and use IPv4 purely for accessing the IPv4 address of the PrivateLink endpoint.



AWS PrivateLink in a dual-stack scenario

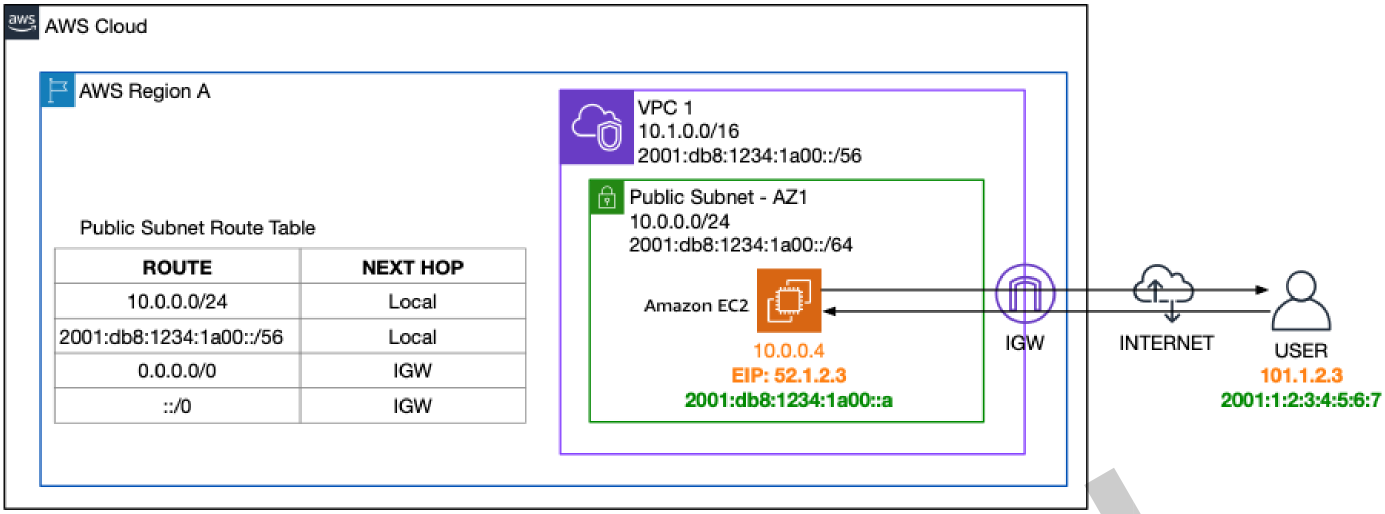
VPC sharing

VPC sharing allows VPC owners to share a subnet across AWS accounts. You may share dual-stack subnets the same way as IPv4-only ones. IPv6 resources deployed into a shared subnet function identical to those deployed into non-shared subnets.

Amazon VPC internet access

Internet reachable IPv6 resources

As previously stated, elastic network interfaces retain their IPv6 addresses throughout their lifetime. For IPv4, elastic network interfaces can have zero or more Elastic IP addresses associated with them. An Elastic IP address defines a static 1:1 NAT relationship between an elastic network interface's IPv4 address and a public internet-routable address.

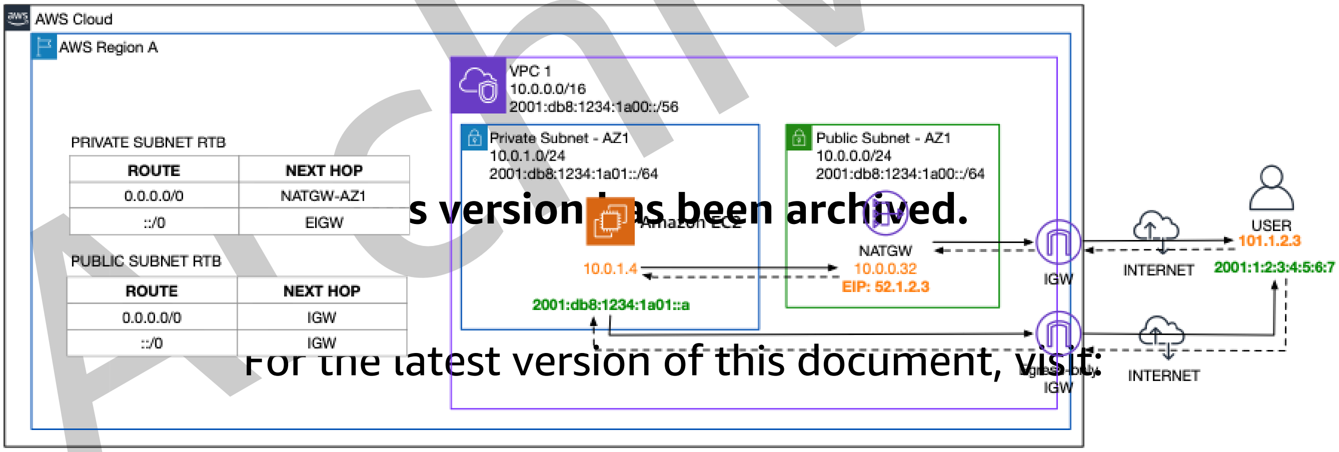


Public Subnet Route Table

ROUTE	NEXT HOP
10.0.0.0/24	Local
2001:db8:1234:1a00::/56	Local
0.0.0.0/0	IGW
::/0	IGW

Internet access for a VPC public subnet

In IPv6, VPC addressing is already globally unique, and therefore Elastic IP addresses are not required. Amazon assigned IPv6 addresses are automatically publicly advertised whereas for BYOIP ranges this is optional. In either case, resources deployed only have IPv6 internet reachability if their subnet's routing table contains IPv6 destinations (such as `::/0`) via either an internet gateway or outbound traffic-only internet gateway.



PRIVATE SUBNET RTB

ROUTE	NEXT HOP
0.0.0.0/0	NATGW-AZ1
::/0	EIGW

PUBLIC SUBNET RTB

ROUTE	NEXT HOP
0.0.0.0/0	IGW
::/0	IGW

Internet access from a VPC private subnet

Hybrid connectivity design

Hybrid connectivity scenarios are a reality for many customers. We offer two methods for addressing these: AWS Direct Connect and AWS managed Site-to-Site VPN.

AWS previously published the Hybrid Connectivity whitepaper, which focused on designs and considerations around these solutions, and most of that content remains relevant. However, that paper does not consider IPv6. This section assumes you are acquainted with the aforementioned document, and it focuses only on the best practices and differences compared to IPv4 implementations.

AWS Direct Connect

AWS Direct Connect is a cloud service solution that makes it easy to establish a dedicated network connection from customer premises to AWS.

Different aspects of the Direct Connect service deal with different layers of the OSI model. The choice of IPv6 only affects configuration related to Layer 3, so many aspects of Direct Connect configuration such as physical connections, link aggregation, VLANs, and jumbo frame are no different from IPv4 use cases.

Where IPv6 does differ is when it comes to addressing and configuration of BGP peerings on top of a virtual network interface (VIF). There are three types of VIFs:

- Private
- Transit
- Public

The remainder of this section covers all of these.

> **Note:** You can provision either 0 or 1 peering per address family on to a given VIF, so it is possible to retrofit IPv6 onto an existing VIF without the need to reprovision or deploy a new one.

Transit and Private VIF IPv6 peerings – Whereas in IPv4 you are free to choose your own addressing for the logical point-to-point, in IPv6, AWS automatically allocates a /125 CIDR for each VIF, and it's not possible to specify custom IPv6 addresses.

Advertising IPv6 prefixes from AWS

When associating a Direct Connect Gateway directly with a Virtual Private Gateway (VGW) you can specify "Allowed Prefixes". Think of this like a traditional "prefix-list" filter, controlling the prefixes advertised to your customer gateway. With IPv4, specifying no filter equates to `0.0.0.0/0` — no filtering. With IPv6, not specifying a value here results in all advertisements being implicitly blocked.

Associating a Direct Connect Gateway with Transit Gateway “Allowed Prefixes” acts not as a filter but as static origination of routes advertised to your customer gateway.

> **Note:** An “Allowed Prefix” IPv6 CIDR must be of length /64 or less specific, but you cannot specify “`::/0`”.
>
> VGW and Transit Gateway behave differently in regard to the “Allowed Prefixes” parameter, as previously described.

Advertising IPv6 prefixes to AWS

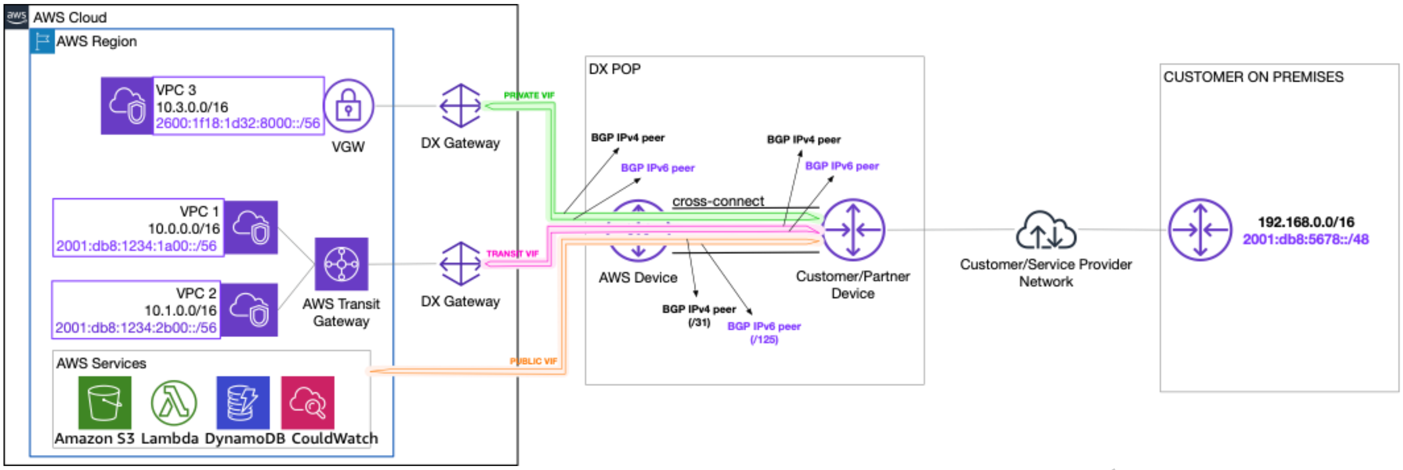
In terms of advertising prefixes from your on-premises facility to AWS, there are no CIDR restrictions, though standard Direct Connect quotas apply. Any prefix advertised is local to your AWS network and not propagated beyond its boundary.

Public VIF IPv6 peerings — As with transit and private VIF peerings, public IPv6 peerings are allocated /125s from AWS owned ranges automatically. These IPs are used to support the BGP peering, and you need to use your own IPv6 prefixes to communicate over the peering.

> **Note**: With IPv4 public VIFs, it’s possible to use the AWS assigned public IPv4 /31 in combination with NAT to enable access from privately addressed on-premises resources (to support use cases where you are not in a position to provide public IPv4 address space). With IPv6, however, you must own and specify an IPv6 prefix at creation. Note that if you specify an IPv6 prefix that you don’t provably own, the VIF will fail to provision and remain in the “verifying” state.

Routing IPv6 over public and private or transit VIFs

If you’re using AWS-assigned CIDR blocks for the VPCs or BYOIPv6 with CIDRs that are marked as *advertisable* by AWS, you will receive over the public VIF peering the summary prefixes that contain the CIDRs of your VPCs. When using public VIFs in conjunction with private/transit VIFs, take into account that your device will receive the same prefixes (the ones for your VPCs) over both types of VIFs. At this point, route filtering on your customer device needs to be taken into consideration, to ensure symmetric flow of traffic over the different virtual interfaces.



Direct Connect dual-stack support

Amazon-managed VPN

AWS Site-to-Site VPN connectivity configuration comprises multiple parts:

- The customer gateway, which is the logical representation of the on-premises VPN end point
- The VPN connection
- The local device configuration on the VPN appliance, represented by the customer gateway

Any AWS S2S VPN connection consists of two tunnels. It is this connection that defines the IP addressing, ISAKMP, IPsec, and BGP peering parameters.

> **Note:** AWS supports IPv6 for IPsec tunnels terminating in Transit Gateway VPN attachments only.

Customer gateways

While AWS supports IPv6 within IPsec tunnels, the underlying connectivity occurs via IPv4. This means that both the AWS and customer VPN terminating devices need to be addressable via public IPv4 addresses. On the AWS side, this IP is automatically allocated from the AWS Region's public EC2 IP space. On the customer side, this means you require an internet reachable IPv4 address for your appliance.

VPN connections

A single VPN connection can carry IPv4 or IPv6 traffic, but not both at the same time. When configuring a connection, you specify either IPv4 or IPv6 for use inside the tunnel. You are free to choose this address, or let Amazon generate it. If you choose to specify a CIDR, it has to be a /126 taken from the unique-local address range `fd00::/8`. If you don't specify one Amazon selects a /128 from this range for you. In either case, the Amazon side of the connection is the first and your side the second usable IPv6 address in the subnet.

Another element to consider are the local and remote IPv6 Network CIDR ranges. These can either be `::/0` or specific prefixes. This configuration defines the IPsec Phase 2 Security Association (SA) that can be negotiated. If you specify a prefix, be sure to configure your customer gateway appliance to negotiate this accordingly.

> **Note:** Even if you specify a /126, the console displays the IPv6 tunnel CIDR as a /128. While an IPv4 address is always configured, traffic on this range will not be able to flow, as the SA settings specified only permit negotiating the IPv6 SA.

Customer gateway configuration

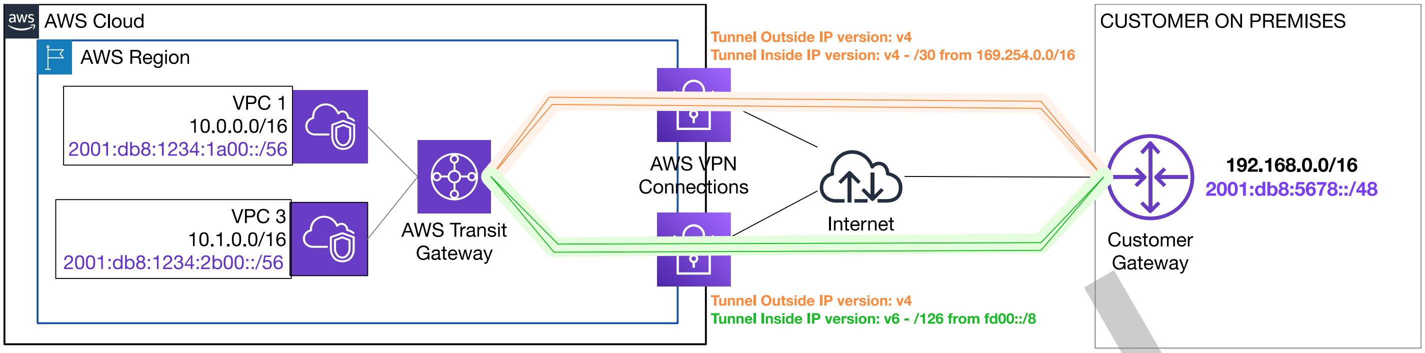
For device-specific configuration of the customer gateway, AWS recommends you consult the vendor's documentation on how to set up IPv6 VPNs. There are vendor-independent nuances to IPv6 S2S VPNs not present in IPv4, and the remainder of this section outlines these.

AWS provides generated configurations for IPsec to download, but currently this covers only IPv4 configuration. Some settings and configuration parameters are independent of IP protocol version. However, settings related to tunnel addressing, phase 2 SA negotiation, and peering configuration do differ.

When using the AWS templates, be sure to omit IPv4 specific parts, and replace them with the IPv6 equivalents.

When specifying the local and remote CIDRs, make sure to configure your VPN appliance to negotiate the IPsec Phase 2 SA to match the AWS side. If you want to use BGP rather than static routing on the VPN and use LOCAL REMOTE CIDR ranges to scope the Phase 2 SA, then the P2P IPs must be included in the CIDR block. As a result, it is only possible to delimit SA scope if using `fd00::/8` on both sides of the VPN, and if non-unique local addressing is used, the SA has to be negotiated as `::/0`.

When using an AWS-generated tunnel IP, or specifying a /128 CIDR range establishment of the BGP, peering will fail by default. The reason is that a /128, like a /32 in IPv4, is a host route. You will need to define a static route pointing at the AWS side of the tunnel to establish the BGP peering.



AWS VPN dual-stack configuration

Designing DNS for IPv6

The core concept of DNS is unchanged from IPv4. From a Layer 3 perspective, DNS is just another application, and therefore, by virtue of the OSI/ISO model provided abstraction, agnostic to the chosen network layer protocol.

Regardless of the IP version, there is a deep link between DNS and the IP layer. The DNS specification has adapted and introduced an additional type to accommodate IPv6 addresses. In IPv6 the equivalent of the IPv4 "A" records are AAAA records. This means that it is possible to use IPv4 as the network protocol to connect to a DNS server and resolve an IPv6 (AAAA) record.

PTR records

A pointer (PTR) record translates an IP address to its domain name. IPv6 addresses are reverse mapped under the domain IP6.ARPA. IPv6 reverse maps use a sequence of nibbles separated by dots with the suffix ".IP6.ARPA" as defined in RFC 3596. For example, the reverse lookup domain name corresponding to the address `2001:db8:1234:1a00::1:2:3:4` would be `4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.a.1.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa`.

Alias records

Amazon Route 53 supports alias records. Route 53 alias records are mapped internally to the DNS name of alias targets such as AWS resources. Route 53 monitors the IP address associated with an alias target's DNS name for scaling actions and software updates. The authoritative response from Route 53 name servers contains an A record (for IPv4 addresses) or AAAA record (for IPv6 addresses) with the IP address of the alias target.

DNS resolution within a host

External configuration aside it is up to a host's networking stack at runtime to resolve DNS records. When configured as dual-stack, most modern operating systems default to preferring IPv6. In other words, when a query for a FQDN returns both an A and AAAA record the OS prefers to use the AAAA record and establishes IPv6 connectivity to the target.

> **Note:** It is up to an IPv6 enabled host's operating system and network stack whether it will attempt to map a given FQDN to an IPv4 or IPv6 address. Ensure that IPv6-enabled hosts are able to resolve AAAA records, and that the network between it and its destination is routable to IPv6. A commonly observed misconfiguration in dual-stack setups is hosts resolving FQDNs to IPv6 addresses, but no end-to-end IPv6 routable path existing between source and destination. Happy Eyeballs (Fast Fallback) is an algorithm published by the IETF which can make dual-stack applications (those that understand both IPv4 and IPv6) more responsive to users by attempting to connect using both IPv4 and IPv6 at the same time (preferring IPv6). This avoids the usual problems faced by users with imperfect IPv6 connections or setups.

Amazon Route 53 DNS records

In AWS, Amazon Route 53 provides DNS capabilities. Route 53 provides features for two use cases:

- Public DNS for externally hosted content
- DNS capability within a VPC both from a resolver and authoritative name server standpoint

Public IPv6 DNS resolution

For externally queryable DNS, you can use Route 53 public hosted zones, with both A and AAAA records. Route 53 health checks support health checking IPv6 services. The name servers exist both for IPv4 and IPv6, meaning clients wanting to resolve a FQDN hosted on Route 53 public hosted zone can do so natively.

DNS resolution within a VPC

Amazon VPC comes with Route 53 Resolver, which provides a built-in capability for resolving DNS names. This resolver is reachable either on 169.254.169.253 or VPC_CIDR_NETWORK + 2 for IPv4 and `fd00:ec2::253` for Nitro-based IPv6 hosts. Requests sent to this resolver are resolved against the combination of private hosted zones associated with the VPC, and any (shared) resolver rules. For more information, refer to Resolving DNS queries between VPCs and your network.

> **Note:** By default, a VPC's DHCP option set will specify these VPC R53 DNS resolvers to DHCP clients. However, if you don't want to use AWS provided DNS resolvers, you can change the DHCP options to point clients at self-hosted or third-party DNS resolvers.

Private DNS resolution

Amazon Route 53 can be configured to act as an authoritative name server for one or more zones. You can configure this by creating Private Hosted Zones (PHZs) and associating it with one or more VPCs. Route 53 supports the creation of AAAA records so that it can be used to resolve FQDNs to IPv6 addresses.

For more information, refer to Working with private hosted zones.

IPv6 security and monitoring considerations

Network-level access control

Amazon VPCs feature two network access control mechanisms, and these exist irrespectively of which version of the IP protocol is used (IPv4 or IPv6):

- Security groups (SGs) at the elastic network interface level
- Network access control lists (network ACLs) at the subnet level

Security groups — A security group acts as a stateful virtual firewall for your instance to control inbound and outbound traffic. Each elastic network interface must have at least one security group associated with it. As security groups default to deny all inbound flows, additional IPv6 inbound rules need to be created when operating IPv6. For example, a web server security group that currently permits `0.0.0.0/0` on port 80 doesn't permit IPv6 traffic. For example, if you wanted to allow all IPv6 traffic you would need an additional rule allowing traffic from `::/0`.

Network access control lists — Network ACLs differ from security groups in several ways:

- They are applied to VPC subnet instead of individual elastic network interface
- They are stateless
- They can have explicit `DENY` added to them
- They default to `ALLOW ANY` for both inbound and outbound connectivity

As a result of the last point, you don't need to update default network ACLs to enable IPv6 connectivity explicitly.

VPC Flow Logs

VPC Flow Logs is a feature that enables you to capture information about the IPv6 traffic going to and from network interfaces in your VPC. VPC Flow Logs for IPv6 traffic works the same as IPv4 where you can create flow logs at the VPC level, the subnet level, or the network interface level. If you create VPC Flow Logs at a VPC or subnet level, every network interface in that VPC or subnet is monitored.

The flow log records can use the *default format* or the *custom format*. With a custom format, you specify which fields are included in the IPv6 flow log records and in which order.

Following is an example of a flow log record for IPv6 traffic using the default format. This is an example, of a default format capture for an ICMP ping traffic from `2406:da1c:491:7402:60ce:a99b:742c:1248` to `2406:da1c:491:7402:4427:239a:3656:4f6a` that was permitted.

Table 1 — VPC Flow Logs default format

Column 1	Column 2
version	2
account-id	944045752502
interface-id	eni-0237699701b6463ba
srcaddr	2406:da1c:491:7402:60ee:a99b:749c:c248
dstaddr	2406:da1c:491:7402:4427:239a:8656:4f3a
srcport	0
dstport	0
protocol	58
packets	6
bytes	624
start	1621949678
end	1621949738
action	ACCEPT
log-status	OK

> **Note:** If a network interface has multiple IPv6 addresses and traffic is sent to a secondary private IPv6 address, the VPC flow log displays the primary private IPv6 address if you simply use the default format "dstaddr" field.

To capture the original destination IPv6 address, you can use a custom format flow log with the "pkt-dstaddr" field. It applies the same for "pkt-srcaddr" field. For other flow log considerations, refer to Flow log limitations.

Flow log data can be published to Amazon CloudWatch Logs or Amazon Simple Storage Service (Amazon S3).

VPC Traffic Mirroring

VPC Traffic Mirroring is a complementary feature to flow logs that copies entire packets, including their payload of network traffic from a specified elastic network interface of an Amazon EC2 instance. Traffic Mirroring copies inbound and outbound IPv4 and IPv6 traffic from the network interfaces that are attached to your Amazon EC2 instances. You can send the mirrored traffic to the network interface of another EC2 instance, or a Network Load Balancer that has a UDP listener (listening on UDP port 4789 - VXLAN).

The mirrored traffic is sent to the traffic mirror target by means of the source VPC IPv4 route table. Note that all mirrored traffic is encapsulated in an IPv4 packet. Traffic Mirroring mirrors both your IPv4 and IPv6 traffic. No special configuration is necessary to enable Traffic Mirroring for your IPv6 traffic, whether the traffic mirror source and the target are in the same VPC, or in a different VPC connected via VPC peering or a Transit Gateway (as long as the traffic mirror source can route to the traffic mirror target by IPv4).

AWS Web Application Firewall

AWS Web Application Firewall (AWS WAF) lets you monitor the HTTP(S) requests that are forwarded to an Amazon CloudFront distribution, an Amazon API Gateway REST API, an Application Load Balancer, or an AWS AppSync GraphQL API. With AWS WAF, the services that are associated with the protected resources can respond either with the requested content or with HTTP 403 status code based on conditions that are specified, such as the IP addresses (either IPv4 or IPv6) that the request originate from.

Web ACL

You use the rules in a web ACL to block or allow web requests based on criteria which includes IP addresses or address ranges (either IPv4 or IPv6 addresses as specified in the IP set) that requests originate from. The IP set match statement inspects the IP address of a web request against a set of IP addresses and address ranges. Use this to allow or block web requests based on the IP addresses (either IPv4 or IPv6) that the requests originate from. AWS WAF IP sets support all IPv4 and IPv6 CIDR ranges except for `0.0.0.0/0` and `::/0`.

AWS Shield

AWS Shield Standard and AWS Shield Advanced provides protection against DDoS attacks. A DDoS attack can prevent legitimate users from accessing a service and can cause the system to crash due to the overwhelming traffic volume. All of the AWS

Shield detection and mitigations work with IPv4 and IPv6 without any impact to performance, scalability, or availability of the service.

AWS Network Firewall

AWS Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service for Amazon Virtual Private Cloud (Amazon VPC). AWS Network Firewall does not currently support IPv6. In the dual-stack mode, you can still use AWS Network Firewall to filter IPv4 traffic going to and coming from an internet gateway, NAT gateway, or over VPN or AWS Direct Connect.

AWS Systems Manager

Resources managed by AWS Systems Manager must have IPv4 connectivity to Systems Manager's endpoints. For example, to connect to an EC2 instance using Systems Manager Session Manager, the instance must be running dual-stack and must have an IPv4 connectivity to the internet or AWS PrivateLink VPC endpoint. Similarly, on-premises resources must also be in dual-stack network mode.

Scaling the dual-stack network design in AWS

Elastic Load Balancing

Elastic Load Balancing (ELB) automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It supports the following types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers.

> **Note:** Application Load Balancers and Network Load Balancers are purpose-built load balancers replacing the functionality of Classic Load Balancers. You can review the benefits of using Application Load Balancers and Network Load Balancers over Classic Load Balancers in Benefits of migrating from a Classic Load Balancer (for Application Load Balancers) and Benefits of migrating from a Classic Load Balancer (for Network Load Balancers). This whitepaper focuses only on Application Load Balancers and Network Load Balancers.

Both load balancer types support internet-facing and internal load balancer schemes. You create internet-facing Application Load Balancers and internet-facing Network Load

Balancers when you have clients communicating on the internet using IPv4 and IPv6. You associate IPv6 CIDR blocks with your VPC and choose the subnets where you launch internet-facing Application Load Balancers or internet-facing Network Load Balancers.

To support IPv6, configure your Application Load Balancers or Network Load Balancers with the “dualstack” IP address type. This means that clients can communicate with the load balancers using both IPv4 and IPv6 addresses. In a dual-stack IP address type, the DNS name of the load balancer provides both IPv4 and IPv6 addresses, and creates A and AAAA records respectively.

Application Load Balancers and Network Load Balancers communicate with their targets using IPv4, regardless of whether the client communicates with the load balancer using IPv4 or IPv6. This allows IPv6 client connections to connect to Application Load Balancers and Network Load Balancers without modifying your application instances and supports backward compatibility with their IPv4 targets.

The Application Load Balancer supports both HTTP and HTTPS listeners to support IPv6. The Network Load Balancer supports IPv6 on frontend connections for only TCP and TLS listeners. Internal Application Load Balancers and Internal Network Load Balancers do not support IPv6 addresses.

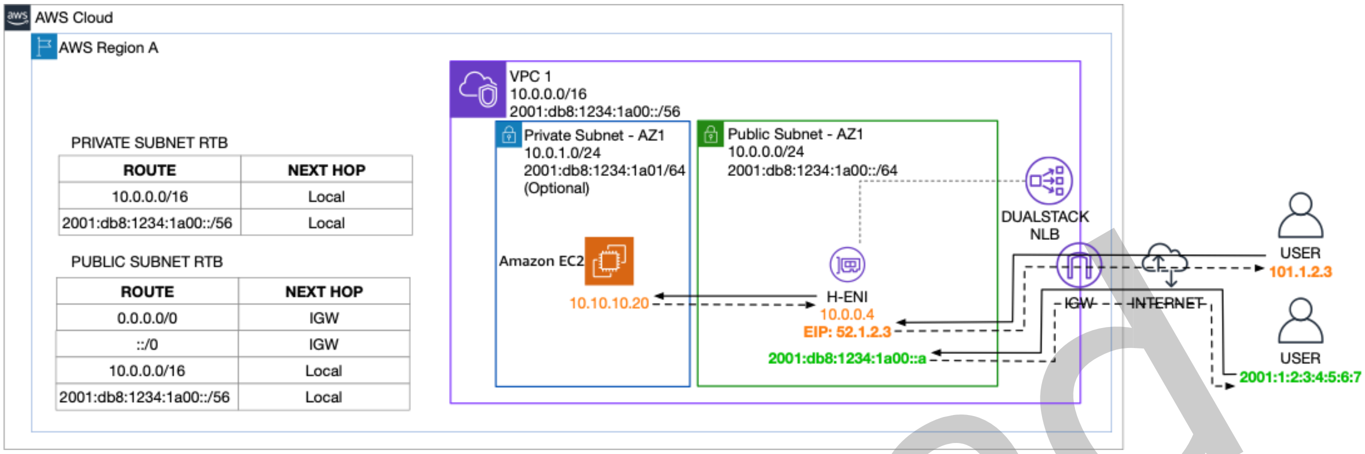
The following table outlines IPv6 support for both listeners and targets across currently existing ELB variants: Classic Load Balancers, Network Load Balancers, Application Load Balancers, and Gateway Load Balancers.

Table 2 - ELB IPv6 support

	Public IPv6 listeners	Private IPv6 listeners	IPv6 targets
Classic Load Balancers	No	No	No
Network Load Balancers	Yes*	No	No
Application Load Balancers	Yes	No	No
Gateway Load Balancers	No	No	No

**TCP and TLS only*

When you enable dual-stack mode on Network Load Balancers, and have the targets reachable over IPv4, Network Load Balancing does the IPv6 to IPv4 conversion and therefore, the source IP is not preserved. However, the load balancer will add the IPv6 source IP to the proxy protocol (PPv2) header and send it to the targets. In this case, the targets need to parse the PPv2 header to obtain the actual source IP of the client.



PRIVATE SUBNET RTB

ROUTE	NEXT HOP
10.0.0.0/16	Local
2001:db8:1234:1a00::/56	Local

PUBLIC SUBNET RTB

ROUTE	NEXT HOP
0.0.0.0/0	IGW
::/0	IGW
10.0.0.0/16	Local
2001:db8:1234:1a00::/56	Local

Dual-stack IPv4/IPv6 internet-facing Network Load Balancer

AWS Global Accelerator

Currently, AWS Global Accelerator does not support IPv6 dual-stack implementations.

Amazon CloudFront

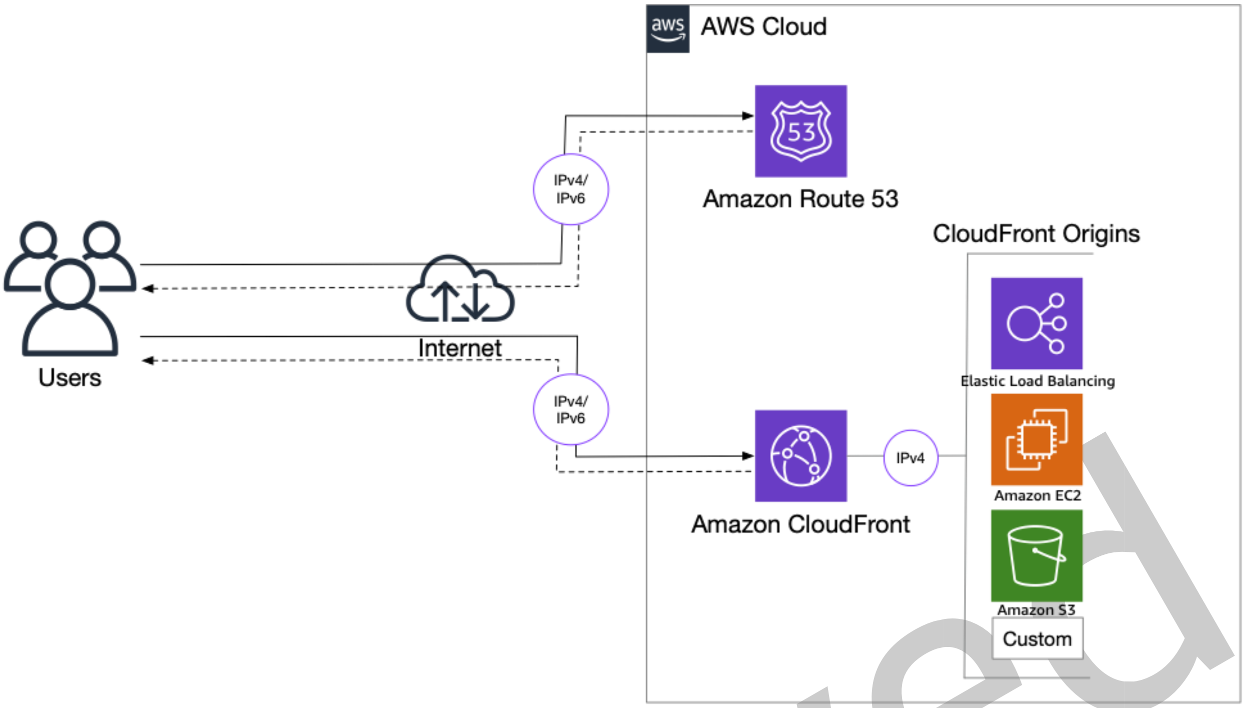
Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds, all within a developer-friendly environment. CloudFront delivers the content through a worldwide network of data centers called edge locations or points of presence (PoPs) that are interconnected via the AWS backbone delivering ultra-low latency performance and high availability to your viewers (end users). It supports both IPv4 and IPv6 from viewer to CloudFront edge locations. This means that a viewer can establish either an IPv4 or IPv6 connection with an edge location.

When a viewer makes an HTTP/HTTPS request over IPv6 to CloudFront, Amazon CloudFront automatically maps network conditions and intelligently routes the user's traffic to the most performant AWS edge location to serve up cached or dynamic content. This edge location responds to the end user natively over IPv6. Clients receive the same content and same security, availability, performance, and scalability whether they are using IPv4 or IPv6.

CloudFront communicates with origin server using IPv4 only. IPv6 origins are not supported. This applies to all types of origins for CloudFront distribution, including Amazon S3 buckets, AWS Elemental MediaStore containers, an AWS Elemental MediaPackage channel, and custom origins.

Following is the request flow for life of an HTTP request in IPv6:

1. An IPv6 viewer (for example, a browser) accesses your website or application and requests an object such as an image file from Amazon CloudFront distribution with FQDN `d111111abcdef8.cloudfront.net` over IPv6.
2. The DNS query first lands at a recursive resolver. This resolver queries Amazon Route 53 for AAAA records for `d111111abcdef8.cloudfront.net`, as Route 53 is the authoritative name server for the `cloudfront.net` domain.
3. Amazon Route 53 name servers are accessible over IPv4 or IPv6. DNS resolves AAAA query and routes the request to the CloudFront POP (edge location) that can best serve the request. Amazon CloudFront automatically maps network conditions and intelligently routes your user's traffic to the most performant AWS edge location to serve up cached or dynamic content.
4. The viewer connects to the respective CloudFront POP.
5. In the POP, CloudFront checks its cache for the requested content. If the image file is in the cache, CloudFront returns them to the viewer. If the files are not in the cache, CloudFront compares the request with the specifications in your distribution and forwards the HTTP request to your origin server over IPv4 for the image file.
6. The origin server responds the image file request back to the edge location.
7. As soon as the first byte arrives from the origin, CloudFront begins to respond the files to the viewer. CloudFront also adds the image file to the cache in the edge location based on the cache headers.



Amazon CloudFront dual-stack IPv4/IPv6 support

Advanced dual-stack network designs

AWS Transit Gateway Connect attachment

AWS Transit Gateway Connect simplifies the branch connectivity through native integration of Software-Defined Wide Area Network (SD-WAN) network virtual appliances into AWS Transit Gateway using Generic Route Encapsulation (GRE) tunnels. For dynamic route exchange over the GRE tunnels AWS Transit Gateway Connect supports the following modes of BGP:

- Exterior BGP (eBGP)
- Interior BGP (iBGP)
- Multi-protocol extension for BGP (MP-BGP)

Multi-protocol BGP (MP-BGP) is an extension to BGP that enables BGP to carry routing information for multiple Network Layer protocols. The types of traffic that BGP can carry are known as *address families*, and the advantage of MP-BGP is being able to route multiple address families over one peering. This also means you can retrofit IPv6 connectivity on an existing connection. Refer to the Integrate SD-WAN devices with AWS Transit Gateway and AWS Direct Connect blog post to learn how.

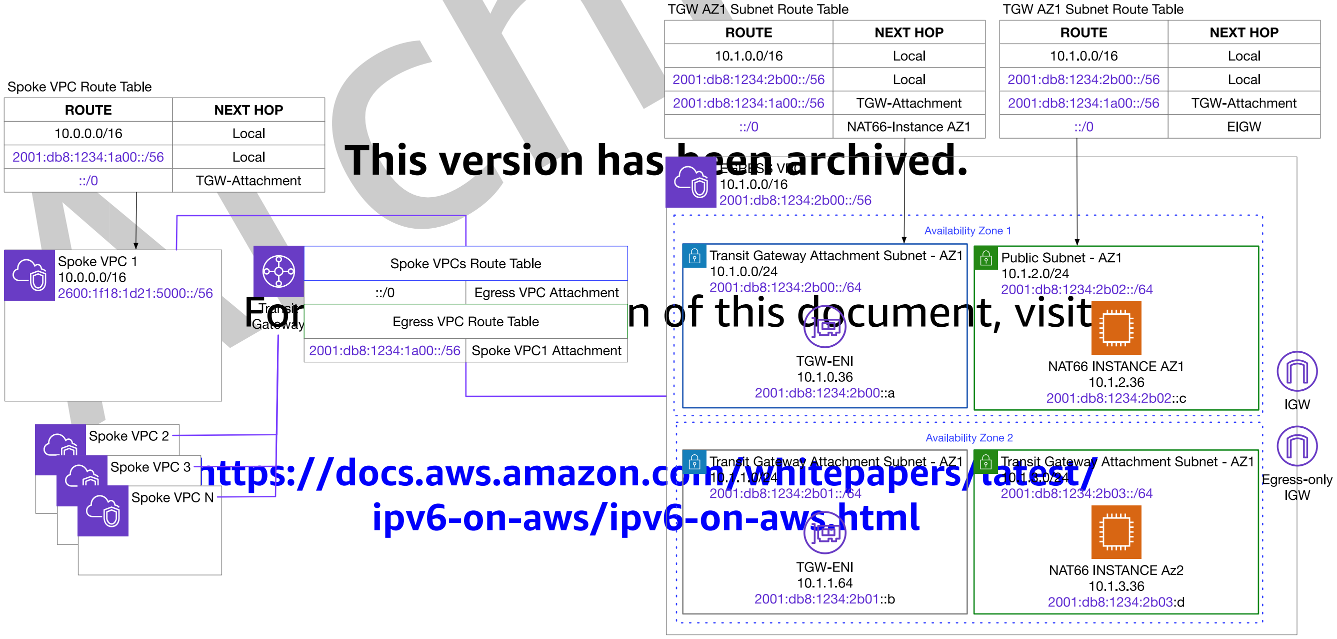
Centralized internet outbound traffic with IPv6

Centralized outbound traffic is a popular pattern with IPv4 as described in Centralized egress to internet. A similar pattern can be implemented with two different approaches:

- Transparent proxy with IPv6 using a technology called *IPv6-to-IPv6 Network Prefix Translation* (NPTv6 or NAT66).
- Explicit forward proxy configured in your OS or application settings.

The first pattern requires use of AWS Transit Gateway and an EC2-based router appliance to perform the NAT66 function. The following figure shows a high-level configuration using a Linux-based NAT66 instance. Traffic from a client follows the default route to the Transit Gateway from where it is forwarded to a centralized outgoing traffic VPC.

Once inside the outbound traffic VPC, it is forwarded to the NAT66 instance which translates the packet's source IP to its own. For additional security, the NAT66 instance may provide firewall functionality. To achieve high availability, you can add multiple Availability Zones, or attach routing appliances using Transit Gateway Connect.



TGW AZ1 Subnet Route Table

ROUTE	NEXT HOP
10.1.0.0/16	Local
2001:db8:1234:2b00::/56	Local
2001:db8:1234:1a00::/56	TGW-Attachment
::/0	NAT66-Instance AZ1

TGW AZ1 Subnet Route Table

ROUTE	NEXT HOP
10.1.0.0/16	Local
2001:db8:1234:2b00::/56	Local
2001:db8:1234:1a00::/56	TGW-Attachment
::/0	EIGW

Spoke VPC Route Table

ROUTE	NEXT HOP
10.0.0.0/16	Local
2001:db8:1234:1a00::/56	Local
::/0	TGW-Attachment

Spoke VPCs Route Table	
::/0	Egress VPC Attachment

Egress VPC Route Table	
2001:db8:1234:1a00::/56	Spoke VPC1 Attachment

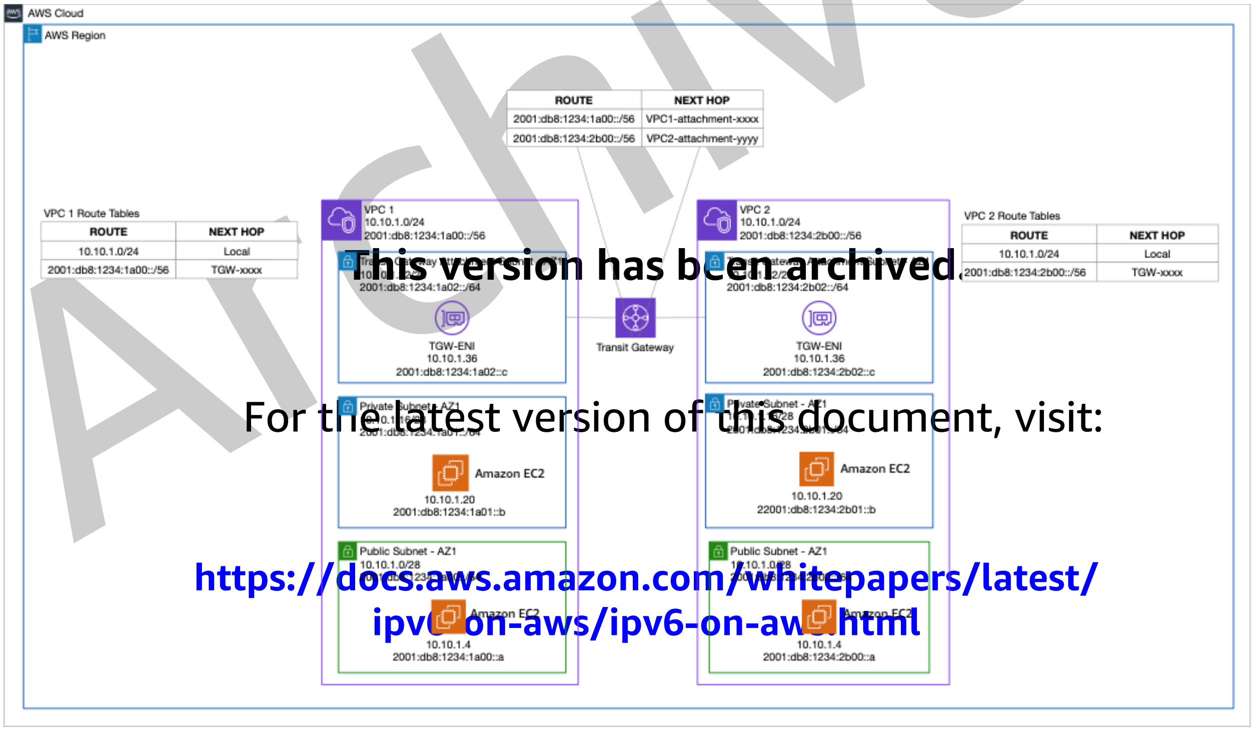
Centralized internet outbound traffic with NAT66

> **Note:** Centralized outbound traffic is applicable if you require centralized traffic inspection. With an outbound traffic only internet gateway, AWS makes it easy to achieve distributed outbound traffic. However, if there is an outbound traffic inspection requirement and it cannot be done in each VPC, the centralized pattern is useful.

The second pattern requires a set of explicit proxies which can be configured in a centralized outbound traffic VPC. Squid proxy is a commonly used open source proxy solution. After forward proxy is configured, and OS or applications settings are updated, you will be able to connect to IPv6 resources on the internet.

Using TGW with IPv6 to work around IPv4 address overlap

Transit gateways can attach to VPCs with overlapping IP address CIDRs. This is because the TGW isn't affected by the address space used in a VPC apart from when it comes to configuring routes. While it is possible to attach a TGW to VPCs with overlapping CIDRs it is not possible to have conflicting routes. However, because IPv6 prefixes are globally unique it can provide connectivity between hosts residing in VPCs with overlapping IP CIDRs.



VPC 1 Route Tables

ROUTE	NEXT HOP
10.10.1.0/24	Local
2001:db8:1234:1a00::/56	TGW-xxxx

ROUTE	NEXT HOP
2001:db8:1234:1a00::/56	VPC1-attachment-xxxx
2001:db8:1234:2b00::/56	VPC2-attachment-yyyy

VPC 2 Route Tables

ROUTE	NEXT HOP
10.10.1.0/24	Local
2001:db8:1234:2b00::/56	TGW-xxxx

IPv6-only routing with AWS Transit Gateway

Conclusion

There are multiple driving forces behind IPv6 adoption. This paper describes what they are and explains how you can respond to them. It also explains differences between IPv4 and IPv6 where applicable, and covers interoperability between both network stacks. As always, security remains paramount and so this paper covered how to evolve your perimeter design to take advantage of IPv6 protocol features.

Remember that IPv6 only makes a difference at the network layer of the networking stack. Many connectivity and security elements, especially in cloud native applications, are handled at higher layers and are therefore not affected.

AWS offers comprehensive IPv6 support in Amazon VPC and AWS services running at the edge of AWS Cloud. You can adopt IPv6 at your own pace and focus on use cases where you will benefit the most from the adoption.

Contributors

Contributors to this document include:

- Alexandra Huides, Senior Networking Specialist Solutions Architect, Strategic Accounts, Amazon Web Services
- Evgeny Vaganov, Head of Networking SA – APJ, Amazon Web Services
- Freddy Hartono, Senior Solutions Architect, Amazon Web Services
- Madhura Kale, Principal Product Manager Network and Compute, Amazon Web Services
- Matt Lehwess, Principal Solutions Architect – EC2, Amazon Web Services
- Rohit Aswani, Specialist Solutions Architect – Networking, Amazon Web Services
- Severin Gassauer-Fleissner, Principal Solutions Architect, Global Financial Services, Amazon Web Services
- Viktor Goldberg, Senior Product Manager, Amazon Web Services

Further reading

For additional information, refer to:

- AWS Hybrid Connectivity whitepaper
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options

Document revisions

Date	Description
January 06, 2022	Minor updates for technical accuracy throughout
October 26, 2021	First publication