<u>This document refers to migrating from IPV4 to IPV6</u>

As we know the AWS going to charge for Public IPV4 from Feb1, 2024

"If you find it difficult to read the entire document, the objective is to determine whether Infor supports an IPv6-based network in the context of network infrastructure. The present instance type used at Infor will support V6, and the operating system's network protocols are the primary factors that ensure application communication compatibility. I believe that the current applications at Infor are not exclusively designed to function on IPv4 or IPv6."

Note: We will not support architectures 9,10,11,12 at Infor as we are not Egress any traffic from Infor. We expect Ingress traffic from end user to Infor

Question and answer:

1) Our customers are using the IPv4 and they are trying to come to Infor (If in case Infor is IPv6) we cannot be IPv6 only at initial phase – we should be in Dual stack (IPv4, IPv6))
   - Answer: "Yes, we do. Regardless of the customer's network/traffic (IPv4 / IPv6), the Dual stack mode enabled on the VPC will handle the incoming traffic and fulfill the requirements according to the network specifications."

2) We have certain products like LM, Mingle/ADFS, LSF. For the prod - customer expect this should be available over the internet

   "Case 1: At Infor, irrespective of the application/product intended to be accessible over the internet, the application-hosted instance will remain the same, with no EIP assigned to it. This is because at Infor, we utilize an Application Load Balancer to access this product. At this point, we are unaware of whether our customer is using IPv4 exclusively. One key point to understand here is that when a customer or end user attempts to access an application over the internet, the Internet Service Provider (ISP) will assign both IPv6 and IPv4 to that end user. In the scenario where we enable the VPC in Dual Stack mode and the subnet within that VPC has opted for IPv6, the ALB sitting in that VPC's IPv6 subnet will become associated with an IPv6 range. Consequently, regardless of the incoming traffic type, we can fulfill the requests.

   Case 2: In contrast to the aforementioned use case, we have certain applications intended to be accessible via an internal Load Balancer, meaning they won't be facing the internet directly. You might be wondering how customers can access these applications in such a setup. In this scenario, our end users connect through a VPN. For instance, if a customer is using IPv4 exclusively and we only support IPv6 (Note: AWS does not facilitate address translation for inbound traffic between IPv4 and IPv6), we employ a Bastion Host. This setup ensures that all IPv4-only traffic is directed to a Dual Stack EC2 instance (using both ENI v4 and ENI v6). this solution is an alternative to AWS-provided options and involves NATing."

Public IP Address Type	Current Price/Hour (USD)	New Price/Hour (USD) (Effective February 1, 2024)
In-use Public IPv4 address (including Amazon provided public IPv4 and Elastic IP) assigned to resources in your VPC, Amazon Global Accelerator, and AWS Site-to-site VPN tunnel	No charge	$0.005
Additional (secondary) Elastic IP Address on a running EC2 instance	$0.005	$0.005
Idle Elastic IP Address in account	$0.005	$0.005

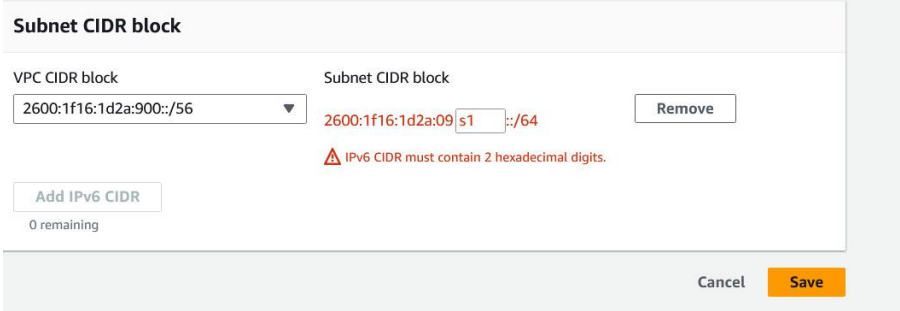
Note: By default the V6 CIDR on VPC is /56 and subnet is /64 and it is in Hexadecimal format.

For Instance: Example IPv6 address: `2001:0db8:85a3:0000:0000:8a2e:0370:7334`

Let's break it down:

- `2001` is the first group of hexadecimal digits.
- `0db8` is the second group.
- `85a3` is the third group.
- `0000` is the fourth group.
- `0000` is the fifth group.
- `8a2e` is the sixth group.
- `0370` is the seventh group.
- `7334` is the eighth group.

If the subnet is /64 then this group needs to be changed:



Moving from AWS Elastic IPv4 to IPv6 on the AWS side comes with its own set of challenges, despite the benefits that IPv6 can offer in terms of addressing limitations and enabling more devices to connect to the internet.

Here are some challenges we might encounter when making this transition within the AWS environment:

IPv6 Adoption: While IPv6 adoption is growing, it's not as widely supported as IPv4. This means that some services, applications, and devices might not fully support IPv6, potentially leading to compatibility issues.

Application Compatibility: Many applications and services are designed with IPv4 in mind. Moving to IPv6 might require updates to software and configurations to ensure they work seamlessly over IPv6.

Networking Complexity: IPv6 introduces new networking concepts and addressing formats that might be unfamiliar to our team. This could lead to misconfigurations or difficulties in troubleshooting.

Security Adjustments: IPv6 has its own set of security considerations that differ from those of IPv4. We need to adapt the security policies and practices to accommodate the specifics of IPv6.

Dual-Stack Configuration: During the transition phase, we might need to run both IPv4 and IPv6 (dual-stack) to ensure compatibility with clients and services using either protocol. Managing dual-stack configurations can be complex.

Route Advertisements: IPv6 uses ICMPv6 router advertisements for address configuration, which is different from the DHCP approach often used in IPv4. Configuring and managing these advertisements correctly is essential for proper IPv6 functionality.

AWS Service Compatibility: While AWS does support IPv6, not all AWS services might fully support it or have feature parity with IPv4. This could impact our choice of services and architectures.

Network Address Translation (NAT): In IPv4, NAT is often used to conserve address space. IPv6 doesn't require NAT in the same way, which might necessitate adjustments to our network architecture.

Firewall and Security Groups: Transitioning to IPv6 might require changes to our firewall and security group configurations to account for the different protocols and address formats.

DNS Configuration: Setting up IPv6 DNS records and ensuring they are correctly propagated can be challenging. Misconfigured DNS can lead to connectivity issues.

Application Testing: Infor Applications need to be tested thoroughly to ensure they work seamlessly in an IPv6 environment. This testing process can be time-consuming.

Monitoring and Troubleshooting: Monitoring tools and techniques for IPv6 might not be as mature as those for IPv4, making it potentially harder to troubleshoot issues.

Address Planning: IPv6 provides a vast address space, but proper planning is still essential to ensure efficient allocation and utilization of addresses.

Provider Support: While AWS supports IPv6, The customer's Internet Service Providers and other third-party services might not fully support it, which could cause connectivity issues.

Despite these challenges, transitioning to IPv6 is becoming increasingly important due to the depletion of IPv4 addresses. It's essential to thoroughly plan the transition, test the configurations, and ensure that our team has the necessary knowledge to manage IPv6 effectively within the AWS environment.

Here are Benefits along with challenges with IPV6:

Moving from IPv4 to IPv6 offers several benefits that address the limitations of the older protocol and accommodate the evolving needs of a growing and increasingly connected world. Here are some key benefits of transitioning to IPv6:

Vast Address Space: IPv6 provides an incredibly large address space, which eliminates the concern of IPv4 address exhaustion. With 128-bit addresses, IPv6 can support an almost unlimited number of unique IP addresses, enabling the connection of countless devices to the internet.

End of NAT: The need for Network Address Translation (NAT) is significantly reduced with IPv6 due to the abundance of available addresses. This simplifies network configuration, improves end-to-end connectivity, and enhances the overall quality of service.

Auto-Configuration: IPv6 includes built-in support for auto-configuration, which simplifies the process of assigning IP addresses to devices. Devices can generate their own unique addresses, reducing the need for manual configuration and making it easier to connect new devices to a network.

Improved Routing Efficiency: IPv6 introduces features like larger subnets and hierarchical addressing, leading to more efficient routing in larger networks. This helps reduce network congestion and enhances the scalability of the internet.

Better Support for Mobile Devices: As the number of mobile and IoT devices continues to grow, IPv6's larger address space and improved auto-configuration make it more suitable for accommodating these devices seamlessly.

Security Enhancements: While security considerations apply to any network protocol, IPv6 includes some security features as part of its design. For instance, IPSec, a suite of protocols for securing internet communications, is often built into IPv6 implementations.

Overall, transitioning to IPv6 is essential for the long-term sustainability and growth of the internet. While the transition can come with challenges, the benefits it brings in terms of address availability, improved performance, and support for emerging technologies make it a crucial step for organizations and networks of all sizes.

Steps to be followed for enabling IPv6 in a VPC with a public and private 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
2. In the navigation pane, choose **Your VPCs**.
3. Select your VPC, choose **Actions**, **Edit CIDRs**.
4. Choose **Add IPv6 CIDR**, choose one of the following options, and then choose **Select CIDR**:

- **Amazon-provided IPv6 CIDR block**: Requests an IPv6 CIDR block from Amazon's pool of IPv6 addresses. For **Network Border Group**, select the group from which AWS advertises IP addresses.
- **IPv6 CIDR owned by me**: (BYOIP) Allocates an IPv6 CIDR block from your IPv6 address pool. For **Pool,** choose the IPv6 address pool from which to allocate the IPv6 CIDR block.

To associate an IPv6 CIDR block with a subnet

1. Open the Amazon VPC console
2. In the navigation pane, choose **Subnets**.
3. Select your subnet, choose **Subnet Actions**, **Edit IPv6 CIDRs**.
4. Choose **Add IPv6 CIDR**. Specify the hexadecimal pair for the subnet (for example, 00) and confirm the entry by choosing the tick icon.
5. Choose **Close**. Repeat the steps for the other subnets in your VPC.

Step 2: Update your route tables

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

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

To update your route table for a public subnet

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

To update your route table for a private subnet

1. Open the Amazon VPC console
2. If you're using a NAT device in your private subnet, it does not support IPv6 traffic. Instead, create an egress-only internet gateway for your private subnet to enable outbound communication to the internet over IPv6 and prevent inbound communication. An egress-only internet gateway supports IPv6 traffic only.

3. In the navigation pane, choose **Route Tables** and select the route table that's associated with the private subnet.
4. On the **Routes** tab, choose **Edit routes**.
5. Choose **Add route**. For **Destination**, specify ::/0. For **Target**, select the ID of the egress-only internet gateway, and then choose **Save changes**.

Step 3: Update your security group rules

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

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

To update your security group rules

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

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

Update your network ACL rules.

If you associate an IPv6 CIDR block with your VPC, we automatically add rules to the default network ACL to allow IPv6 traffic, provided you haven't modified its default rules. If you've modified your default network ACL or if you've created a custom network ACL with rules to control the flow of traffic to and from your subnet, you must manually add rules for IPv6 traffic.

IPv6 Supported Instance type/Change your Instance type:

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

Note: To resize your instance, be aware of the compatibility limitations. Also, to resize your instance, you must stop it. Stopping and starting an instance changes the public IPv4 address for the instance, if it has one. If you have any data stored on instance store volumes, the data is erased.

To resize your instance

1. Open the Amazon EC2 console
2. In the navigation pane, choose **Instances**, and select the database instance.
3. Choose **Actions**, **Instance State**, **Stop**.
4. In the confirmation dialog box, choose **Yes, Stop**.
5. With the instance still selected, choose **Actions**, **Instance Settings**, **Change Instance Type**.
6. For **Instance Type**, choose the new instance type, and then choose **Apply**.
7. To restart the stopped instance, select the instance and choose **Actions**, **Instance State**, **Start**. In the confirmation dialog box, choose **Yes, Start**.

If your instance is an instance store-backed AMI, you can't resize your instance using the earlier procedure. Instead, you can create an instance store-backed AMI from your instance, and launch a new instance from your AMI using a new instance type

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

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

Step 5: Assign IPv6 addresses to your instances

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

To assign an IPv6 address to your instance

1. Open the Amazon EC2 console
2. In the navigation pane, choose **Instances**.

3. Select your instance, and choose **Actions**, **Networking**, **Manage IP Addresses**.
4. Under **IPv6 Addresses**, choose **Assign new IP**. You can enter a specific IPv6 address from the range of your subnet, or you can leave the default Auto-Assign value to let Amazon choose one for you.
5. Choose **Yes, Update**.

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

To assign an IPv6 address to an instance during launch

1. Open the Amazon EC2 console
2. Select your AMI and an IPv6-compatible instance type, and choose **Next: Configure Instance Details**.
3. On the **Configure Instance Details** page, select a VPC for **Network** and a subnet for **Subnet**. For **Auto-assign IPv6 IP**, select **Enable**.
4. Follow the remaining steps in the wizard to launch your instance.

You can connect to an instance using its IPv6 address. If you're connecting from a local computer, ensure that your local computer has an IPv6 address and is configured to use IPv6.

Step 6: (Optional) Configure IPv6 on your instances

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

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

To verify DHCPv6 on Linux

Use the **ping6** command as follows.

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

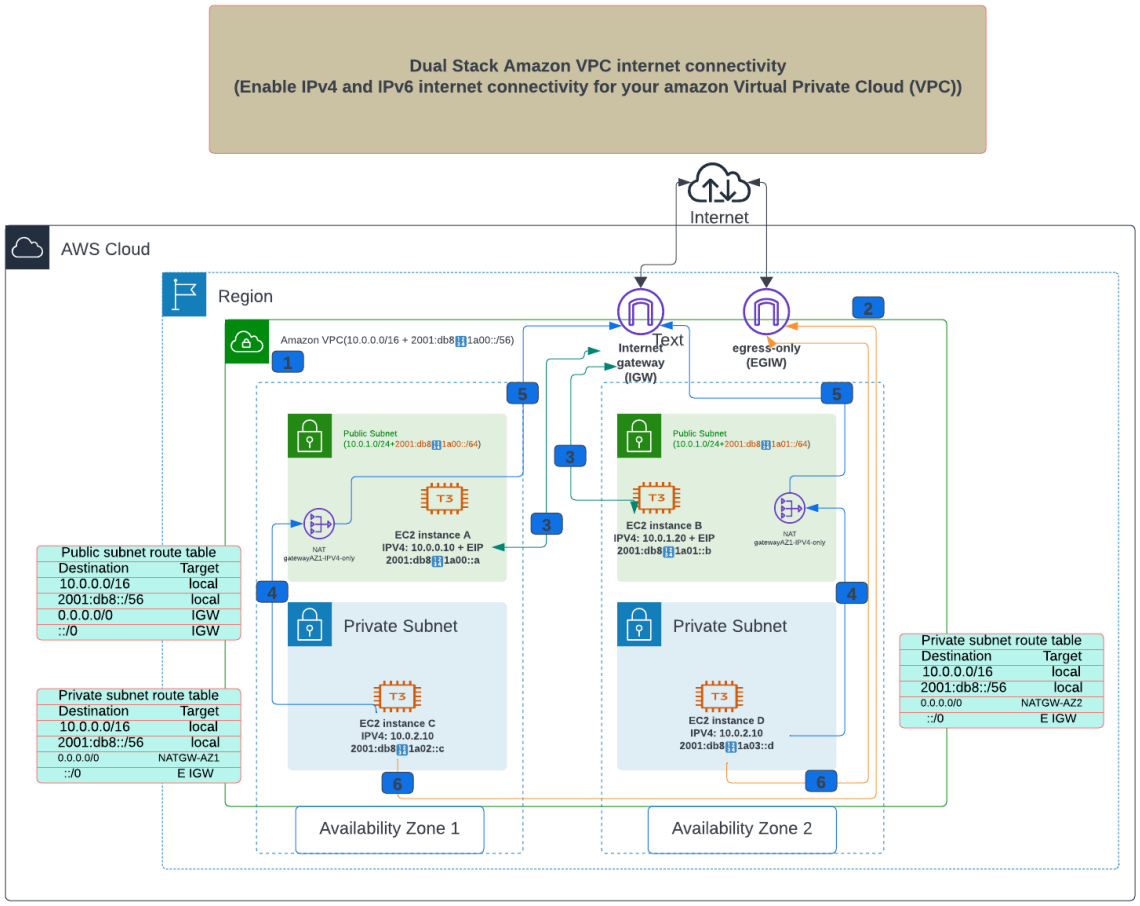
To verify DHCPv6 on Windows

Use the **ping** command as follows.

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

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

IPV6 reference architectures for AWS and hybrid networks

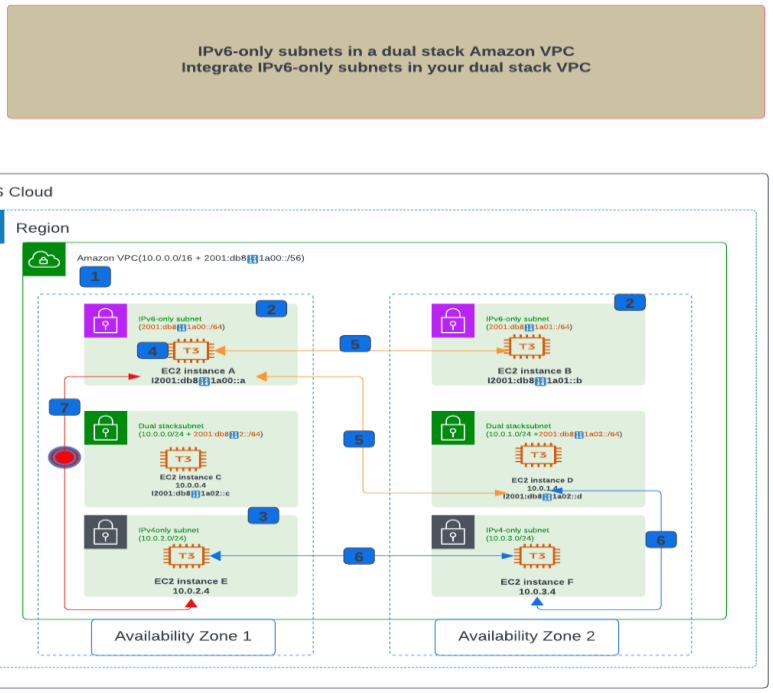


At Subnet level

Note:

1. Associate an IPv6 Classless Inter-Domain Routing (CIDR) block to your Amazon Virtual Private Cloud (Amazon VPC). This can be an AWS-assigned CIDR, or part of a Bring Your Own IPv6 Addresses (BYOIPv6) pool.
2. Associate an egress-only internet gateway (EIGW) to the VPC. This is the target for the IPv6 default route of private dual stack subnets.

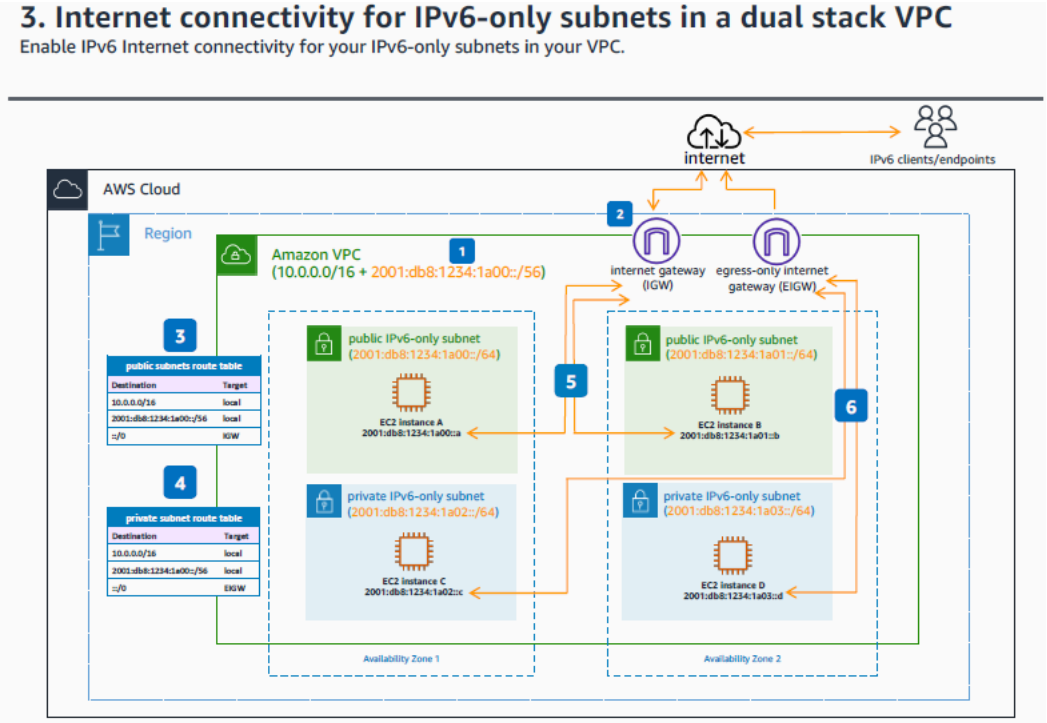
3. Compute resources in public dual stack subnets use the internet gateway for dualstack IPv4 and IPv6 internet connectivity. They can directly initiate outbound internet connections and accept inbound internet connections, to and from IPv4 and IPv6 hosts in the internet, using their associated Elastic IPv4 address or IPv6 addresses from the subnet CIDR. Note that security groups must allow both IPv4 and IPv6 traffic.
4. Resources in private dual stack subnets use the public NAT gateway in each Availability Zone for outbound IPv4 Internet connectivity. The NAT gateway allows only outbound IPv4 connections to be opened from private Amazon Elastic Compute Cloud (Amazon EC2) instances to internet IPv4 destinations, and the associated return traffic.
5. The NAT gateways send the translated IPv4 packets to the internet gateway, which sends the traffic out in the internet, to the respective IPv4 destinations.
6. Resources in private dual stack subnets use the egress-only internet gateway for outbound IPv6 internet connectivity. The egress-only internet gateway allows only outbound IPv6 connections to be opened from private EC2 instances to internet IPv6 destinations, and the associated return traffic



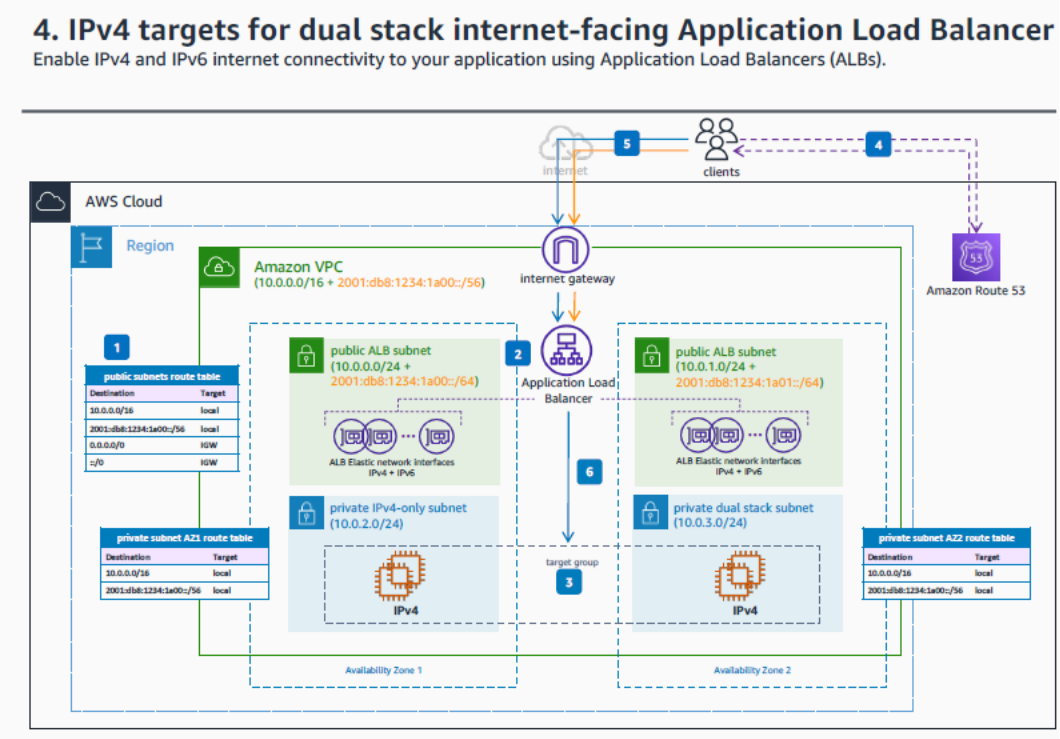
Note:

1. To create IPv6-only subnets, you start with a dual stack Amazon VPC (Virtual Private Cloud). The VPC needs to have a secondary IPv6 CIDR associated with it.
2. IPv6-only subnets only have an IPv6 CIDRs associated with them, and do not need associated IPv4 CIDRs. Also, IPv6-only subnets are bound to an Availability Zone the same as dual stack or IPv4-only subnets are.
3. IPv6-only subnets can coexist in the same VPC as IPv4-only subnets and dual stack subnets. You can also choose to have dual stack VPCs with IPv6-only subnets, but keep in mind that the primary VPC CIDR is an IPv4 CIDR.
4. When you create an EC2 instance in the IPv6-only subnet, the IPv6 address is either automatically assigned from the subnet CIDR through DHCPv6, or you can manually configure it.

5. IPv6-only resources in IPv6-only subnets can communicate natively over IPv6 with other IPv6-only resources or dual stack resources in other subnets in the VPC.
6. IPv4-only resources in the IPv4-only subnets can communicate natively over IPv4 with other IPv4-only resources or dual stack resources in other subnets in the VPC.
7. IPv6-only resources in the IPv6-only subnets cannot communicate with IPv4-only resources in other subnets in the VPC



1. Starting from your dual stack Amazon VPC, the primary CIDR is an IPv4 one, and the secondary IPv6 CIDR is used to create the IPv6-only subnets.
2. The internet gateway (IGW) and the egress-only internet gateway (EIGW) are attached to the dual stack VPC. Although the IPv6 addresses are Global Unicast Addresses (GUA), you can still create public and private subnets, controlling the IGW and EIGW routing and security group configuration.
3. The public IPv6-only subnets route tables have the default IPv6 route, ::/0, with the internet gateway as the target.
4. The private IPv6-only subnets route tables have the default IPv6 route, ::/0, with the EIGW as target.
5. Compute resources in public IPv6-only subnets use the internet gateway for IPv6 internet connectivity. They can directly initiate outbound internet connection and accept inbound internet connection, IPv6 endpoints in the internet, using their IPv6 addresses from the subnet CIDR. Note that security groups must allow IPv6 traffic.
6. Resources in private IPv6-only subnets use the EIGW for outbound IPv6 internet connectivity. The egress-only internet gateway allows only outbound IPv6 connections to be opened from private EC2 instances to internet IPv6 destinations, and the associated return traffic.

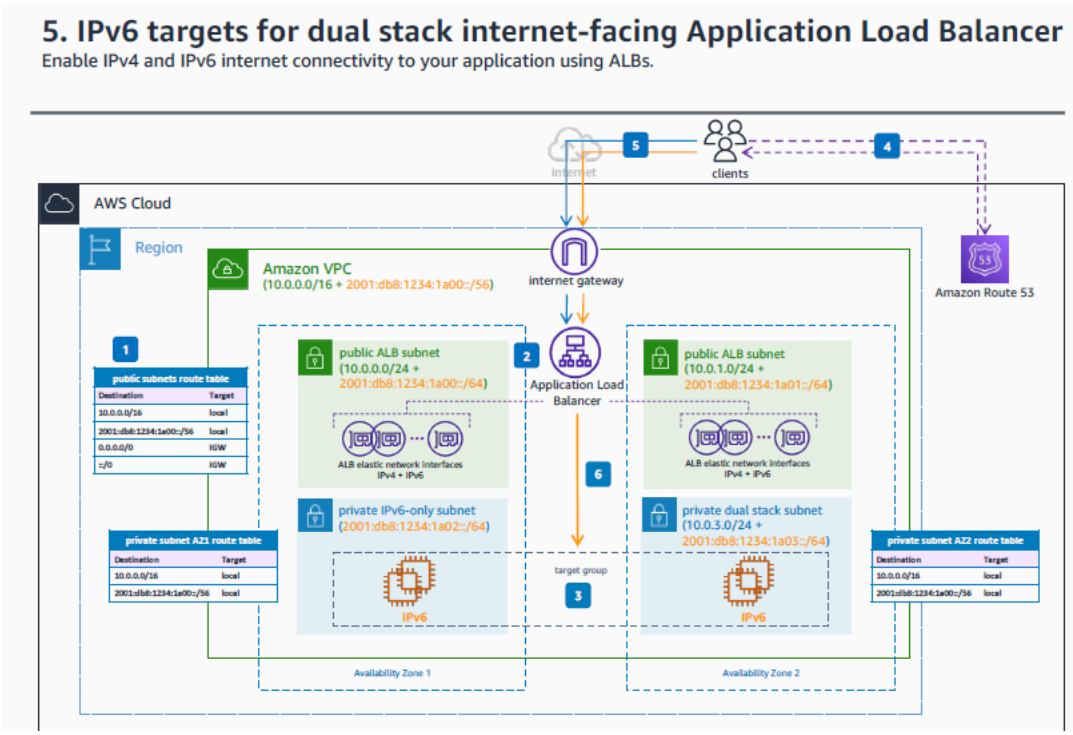



```
C:\Users\PC>nslookup Windows-alb-1228776283.us-east-1.elb.amazonaws.com
Server:  dsldevice6.attlocal.net
Address:  2600:1700:e74:910::1

Non-authoritative answer:
Name:    Windows-alb-1228776283.us-east-1.elb.amazonaws.com
Addresses:  2600:1f18:5a9f:bc12:c19:6754:17b4:b1cd
          2600:1f18:5a9f:bc11:4540:eb09:66bc:5931
          34.231.174.215
          100.24.207.235

C:\Users\PC>
```

1. Configure your VPC **ALB** subnets for dual stack internet connectivity by adding the default routes for IPv4 and IPv6.
2. Deploy your dual stack internet-facing **ALB**, and select the dual stack **ALB** subnets in the VPC.
3. Your application stack remains unchanged with the dual stack added functionality for the application endpoints with **ALB**. The **ALB** and target group instances continue using IPv4 for communication.
4. The application clients query for the application name and receive the IPv4 or IPv6 address of the endpoint, based their capabilities. For single stack customers, their stack determines the protocol to be used. For dual stack enabled clients, the operating system configuration determines the use of IPv4 or IPv6 for communication.
5. The clients open connections to the application endpoint, using either IPv4 or IPv6.
6. The **ALB** distributes traffic to the healthy targets in the target groups using IPv4 connections.

5. IPv6 targets for dual stack internet-facing Application Load Balancer

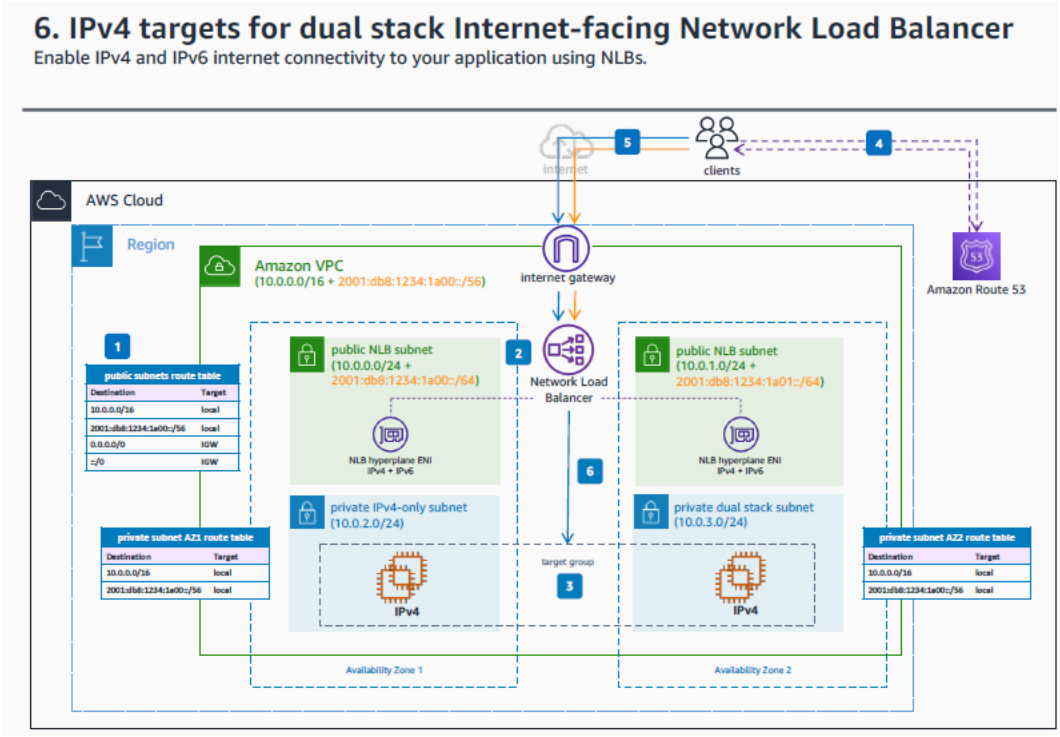
Enable IPv4 and IPv6 internet connectivity to your application using ALBs.



1. Configure your VPC **ALB** subnets for dual stack internet connectivity by adding the default routes for IPv4 and IPv6.
2. Deploy your dual stack internet-facing **ALB**, and select the dual stack **ALB** subnets in the VPC.
3. Your application stack can run natively on IPv6, as the **ALB** supports IPv6 targets. IPv6 target groups only support IP type targets.
4. The application clients query for the application name and receive the IPv4 or IPv6 address of the endpoint, based on their capabilities. For single stack customers, their stack determines the protocol to be used. For dual stack enabled clients, the operating system configuration determines the use of IPv4 or IPv6 for communication.
5. The clients open connections to the application endpoint, using either IPv4 or IPv6.
6. The **ALB** distributes traffic to the healthy targets in the target groups using IPv6 connections.

6. IPv4 targets for dual stack Internet-facing Network Load Balancer

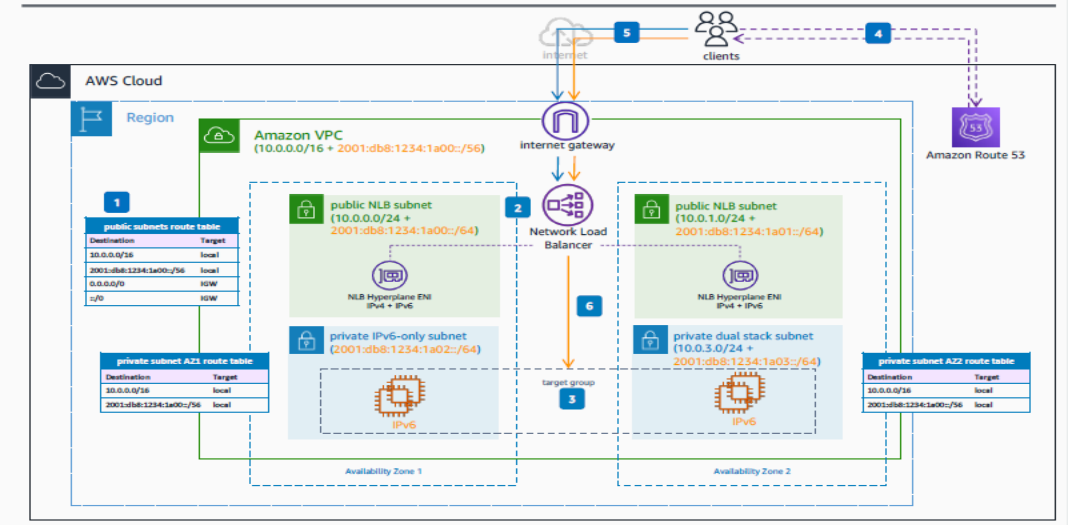
Enable IPv4 and IPv6 internet connectivity to your application using NLBs.



1. Configure your VPC **NLB** subnets for dual stack internet connectivity by adding the default routes for IPv4 and IPv6.
2. Deploy your dual stack internet-facing **NLB**, and select the dual stack **NLB** subnets in the VPC.
3. Your application stack remains unchanged with the dual stack added functionality for the for your application endpoints with **NLB**. The **NLB** and target group instances continue using IPv4 for communication.
4. The application clients query for the application name and receive the IPv4 or IPv6 address of the endpoint, based on their capabilities. For single stack customers, their stack determines the protocol to be used. For dual stack enabled clients, the operating system configuration determines the use of IPv4 or IPv6 for communication.
5. The clients open connections to the application endpoint, using wither IPv4 or IPv6.
6. The **NLB** distributes traffic to the healthy targets in the target groups using IPv4 connections.

7. IPv6 targets for dual stack internet-facing Network Load Balancer

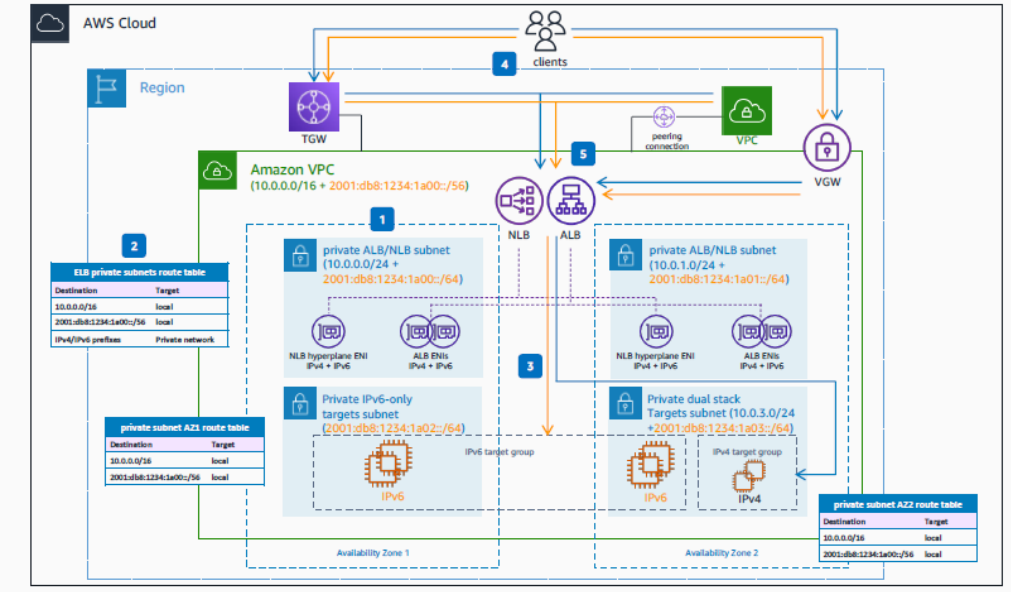
Enable IPv4 and IPv6 internet connectivity to your application using NLBs.



1. Configure your VPC **NLB** subnets for dual stack internet connectivity by adding the default routes for IPv4 and IPv6.
2. Deploy your dual stack internet-facing **NLB**, and select the dual stack **NLB** subnets in the VPC.
3. Your application stack can run natively on IPv6, as the **NLB** supports IPv6 targets. IPv6 target groups only support IP type targets.
4. The application clients query for the application name and receive the IPv4 or IPv6 address of the endpoint, based on their capabilities. For single stack customers, their stack determines the protocol to be used. For dual stack enabled clients, the operating system configuration determines the use of IPv4 or IPv6 for communication.
5. The clients open connections to the application endpoint, using wither IPv4 or IPv6.
6. The **NLB** distributes traffic to the healthy targets in the target groups using IPv6 connections.

8. Dual stack internal Application and Network Load Balancers

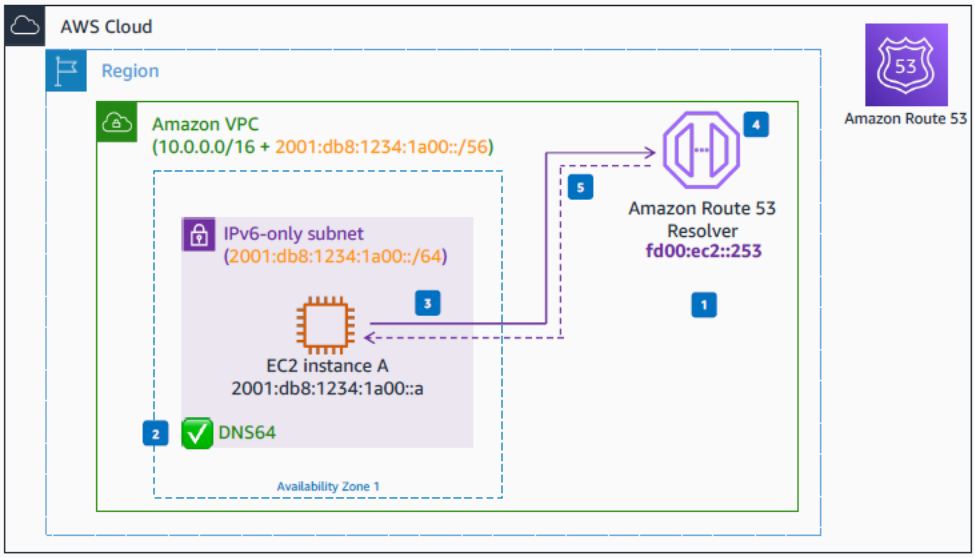
Enable IPv4 and IPv6 private connectivity to your application using Application and Network Load Balancers.

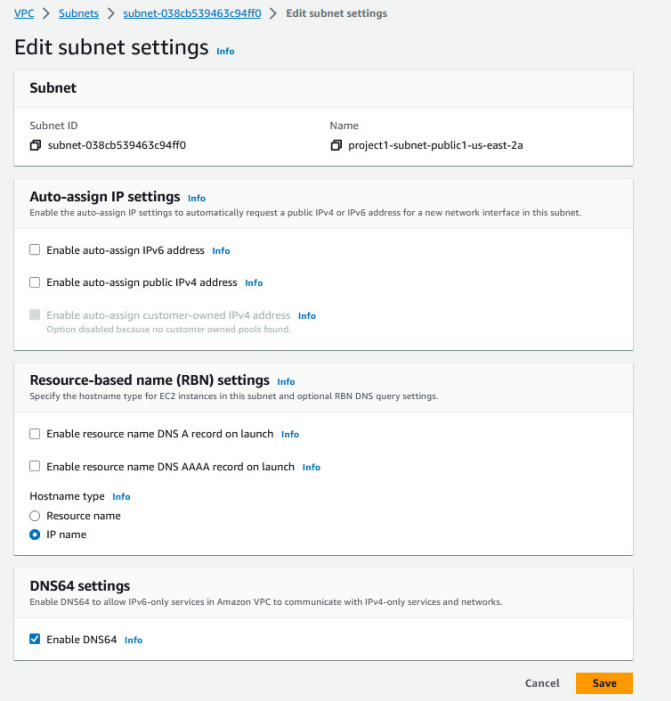


1. Configure your VPC **NLB** or **Application Load Balancer (ALB)** subnets as dual stack, to accommodate for the internal **Elastic Load Balancing** (**ELB**) instances.
2. Depending on the private connectivity method you have in place for your VPC – VPC peering, **AWS Transit Gateway (TGW)**, Virtual Private Gateway, VPN, or **AWS Direct Connect** – the ALB/NLB private subnets must be configured with the appropriate routes, for both IPv4 and IPv6 stacks.
3. Target groups for both **ALBs** and **NLBs** can contain either only IPv6 targets or only IPv4 targets. You can't register an IPv4 target with an IPv6 target group.
4. Clients reaching out to the private **ELB** endpoints can be both IPv4 and IPv6 and can have connectivity over any private connectivity method supported by the Amazon VPC – VPC peering, **AWS Transit Gateway**, Virtual Private Gateway, VPN or **AWS Direct Connect**.
5. For internal ALBs and NLBs, the new attribute flag **ipv6.deny-all-igwtraffic** blocks internet gateway (IGW) access to the load balancer, preventing unintended access to your internal load balancer through an internet gateway. It is set to false for internet-facing load balancers and true for internal load balancers. This attribute does not prevent non-IGW internet access (such as, through peering, **Transit Gateway**, **AWS Direct Connect**, or **AWS VPN**).

9. DNS64

Enable DNS resolution for queries from IPv6-only resources to IPv4-only names.





1. When you create a VPC, the **Amazon Route 53 Resolver** that is created by default, maps to a DNS server that runs on a reserved IPv4 address for the VPC network range, plus 2. For the IPv6 stack, the **Route 53 Resolver** can be reached at the local address fd00:ec2::253.
2. Without DNS64, a DNS query for an IPv4- only service will yield an IPv4 destination address in response and your IPv6-only service cannot communicate with it. To bridge this communication gap, you can enable DNS64 for a subnet and it applies to all the AWS resources within that subnet.
3. The IPv6-only instance sends a DNS query for on IPv4-only endpoint to the **Route 53 Resolver**.
4. With DNS64, the **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.
5. The **Amazon Route 53** resolver replies with a synthesized IPv6 address made of the well-known 64:ff9b::/96 prefix and the IPv4 address.

What is DNS 64?

DNS64 is a technology that complements NAT64 (Network Address Translation 64) in the context of IPv6 and IPv4 communication. While NAT64 is responsible for translating the actual network packets between IPv6 and IPv4, DNS64 focuses on translating domain names to IPv6 addresses when only IPv4 addresses are available.

In a scenario where a device using IPv6 needs to access a resource that only has an IPv4 address (common during the transition phase), DNS64 comes into play. It works as follows:

1. The IPv6-only device sends a DNS query for a domain name, requesting the IPv6 address of the resource.

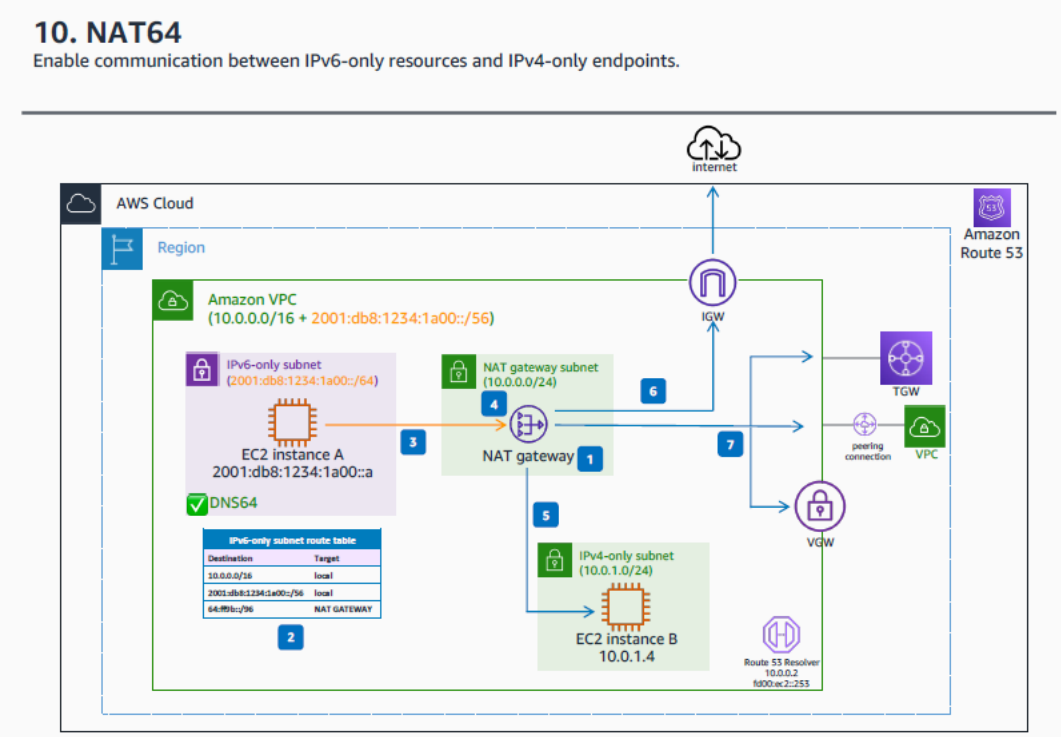
2. If the DNS server being queried has DNS64 enabled, and the queried domain only has an IPv4 address (no IPv6 address), the DNS64 server will respond with a synthesized AAAA record (IPv6 address record) for the domain.

3. This synthesized AAAA record will contain an IPv6 address derived from the IPv4 address of the NAT64 device and a special prefix. This IPv6 address is used solely for the purpose of initiating the communication through the NAT64 device.

4. The IPv6-only device then uses this synthesized IPv6 address to communicate with the NAT64 device.

5. The NAT64 device receives the incoming traffic with the synthesized IPv6 address and translates it to the appropriate IPv4 address. It also performs the necessary packet translation using NAT64.

In summary, DNS64 ensures that IPv6-only devices can access resources that are available only on IPv4 networks. It achieves this by providing a synthesized IPv6 address that points to the NAT64 device, allowing the NAT64 to handle the necessary address translation and communication between the two different IP versions. This combination of NAT64 and DNS64 is part of the toolkit used to facilitate the coexistence and interoperability of IPv6 and IPv4 networks during their transition phase.



1. 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.
2. You need to route traffic for the well-known 64:ff9b::/96 prefix through the NAT gateway, which performs the necessary translation on the traffic to allow IPv6 services to access IPv4 services outside that subnet.
   Note: The IPv6 address range "64:ff9b::/96" is a special reserved prefix used for IPv6 to IPv4 translation, specifically for a mechanism known as IPv6-to-IPv4 address mapping.

This prefix is used in a technique called "IPv6/IPv4 Translation Mechanism," which is defined in RFC 6052. It's used to facilitate communication between IPv6-only and IPv4-only devices by embedding IPv4 addresses within IPv6 addresses. This is often used in situations where both IPv6 and IPv4 networks need to interoperate.

The specific prefix "64:ff9b::/96" is reserved for stateless translation, where the last 32 bits (the IPv4 address part) are used to map to the IPv4 address of the corresponding device. The "64" indicates that it's a subnet prefix with a length of 96 bits, and "ff9b" is a specific reserved value. The rest of the bits are used to carry the IPv4 address.

In summary, "64:ff9b::/96" is used in IPv6-to-IPv4 translation mechanisms to allow IPv6 devices to communicate with IPv4 devices using a special addressing scheme.

3. The IPv6 packet from the IPv6-only instance is sent to the NAT64 gateway. The IP is the instance IPv6 address, and the destination IP is the DNS64 synthesized IPv6 address, that was returned by the **Route 53 Resolver**.
4. 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:
    a. The source IPv6 with its own private IPv4 address.
    b. The destination IPv6 to IPv4 by truncating the 64:ff9b::/96 prefix.
5. Traffic can go to IPv4-only resources in the same VPC.
6. Traffic can go to IPv4-only endpoints in the Internet, if the NAT gateway is public and has an elastic IP associated.
7. Traffic can go to IPv4-only resources in the private network, over VPN, **AWS Direct Connect**, VPC peering, or **Transit Gateway**.

Note: You can enable DNS 64 from EC2 itself by typing this command "aws ec2 modify-subnet-attribute --subnet-id subnet-x.x.x.x –-enable-dns64"

What is NAT 64?

NAT64, which stands for Network Address Translation 64, is a technology used to facilitate communication between IPv6 (Internet Protocol version 6) and IPv4 (Internet Protocol version 4) networks.

IPv6 and IPv4 are two different versions of the Internet Protocol, with IPv6 designed to address the limitations of IPv4, such as the depletion of available IPv4 addresses due to the exponential growth of internet-connected devices. IPv6 uses a much larger address space, allowing for a vastly increased number of unique addresses.

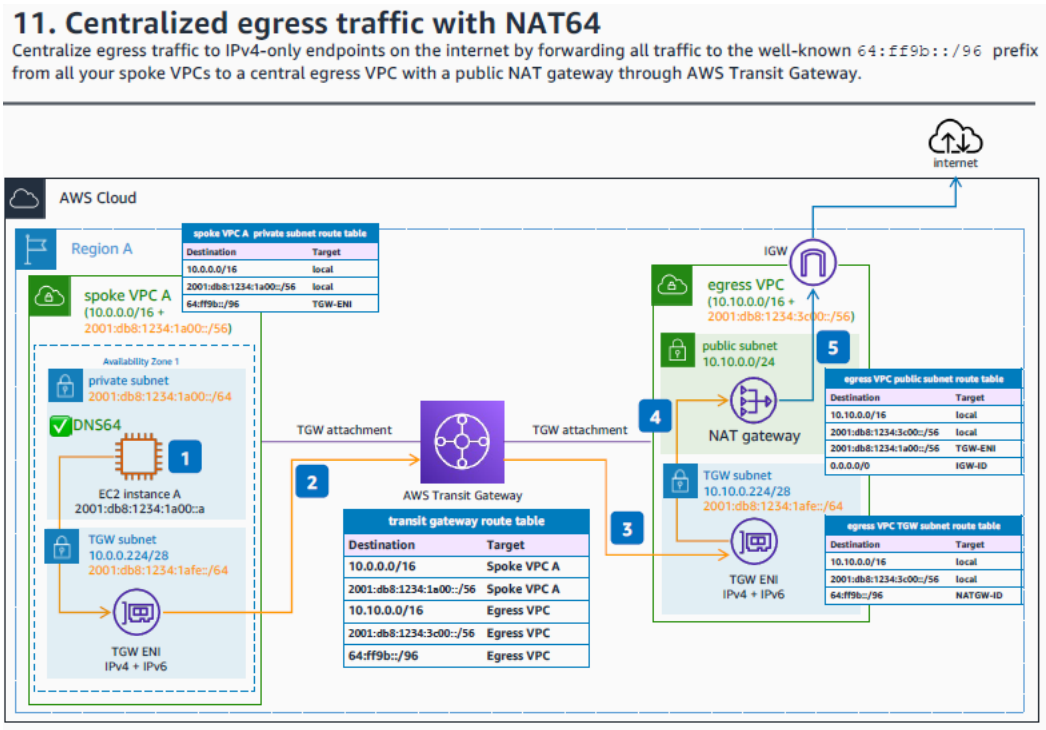
However, during the transition from IPv4 to IPv6, many networks and devices might still be using IPv4, while some are already on IPv6. This can create communication challenges, as the two protocols are not directly compatible. This is where NAT64 comes in.

NAT64 acts as a translator or gateway between IPv6 and IPv4 networks. It allows IPv6-only devices to communicate with IPv4-only devices by translating IPv6 packets to IPv4 packets and vice versa. When an IPv6 device wants to communicate with an IPv4 device, the NAT64 device translates the IPv6 addresses to IPv4 addresses and performs the necessary protocol translation.

In essence, NAT64 helps bridge the gap between IPv6 and IPv4 networks, enabling communication between devices using different IP versions. This is especially important during the transition period when both protocols are in use.

11. Centralized egress traffic with NAT64

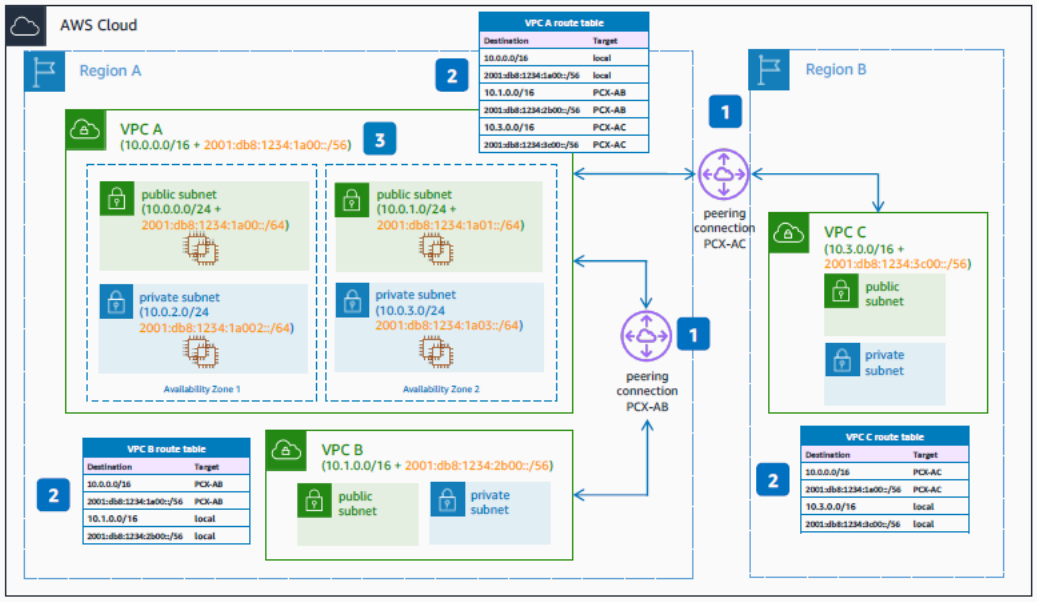
Centralize egress traffic to IPv4-only endpoints on the internet by forwarding all traffic to the well-known `64:ff9b::/96` prefix from all your spoke VPCs to a central egress VPC with a public NAT gateway through AWS Transit Gateway.



1. You need to enable DNS64 in all the subnets where you want your IPv6-only workloads to talk with IPv4-only destinations. That way, the **Amazon Route 53 Resolver** looks up the DNS record for the service queried. If there is no IPv6 address associated with the destination, it synthesizes one by prepending the well known 64:ff9b::/96 prefix, to the IPv4 address in the record.
2. You need to route traffic for the well-known 64:ff9b::/96 prefix through the NAT gateway located in the egress VPC. As per the spoke VPC A private subnet route table, all the traffic to the 64:ff9b::/96 prefix is routed first to the TGW ENI.
3. In the **AWS Transit Gateway** route table associated to the spoke VPC attachments, you need to add an static route sending all the traffic to the 64:ff9b::/96 prefix via the egress VPC attachment.
4. As per the egress VPC's TGW subnet route table, all the traffic to the 64:ff9b::/96 prefix is routed to the NAT gateway.
5. The NAT gateway recognizes that the original destination is IPv4 and translates the IPv6 packets to IPv4 by replacing:
    a. The source IPv6 with its own public EIP IPv4 address.
    b. The destination IPv6 to IPv4 by truncating the 64:ff9b::/96 prefix.
6. If only connectivity to IPv4 resources inside this topology is needed, a private NAT gateway can be used.

12. Dual stack peering connectivity for the Amazon VPC

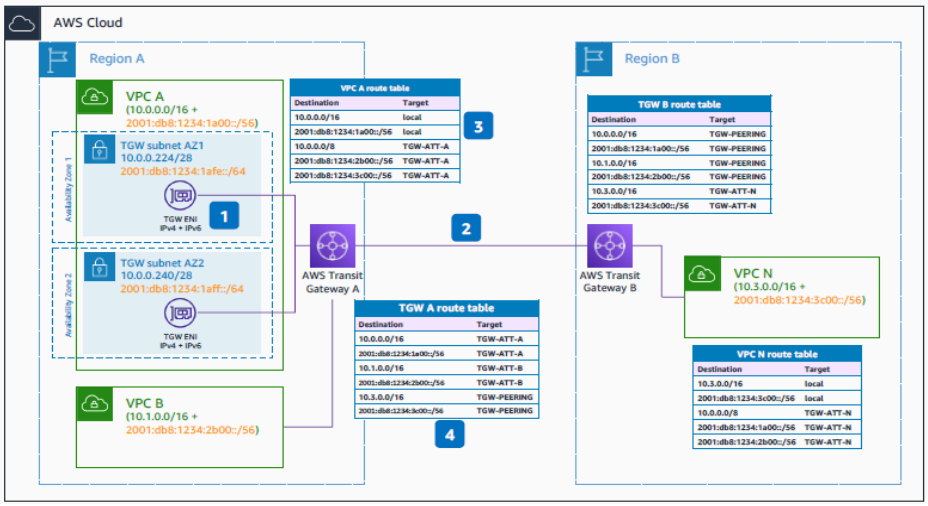
Configure IPv4 and IPv6 VPC peering connectivity.



1. **Amazon VPC** peering natively supports routing for both IP stacks, and you can create both intra-Region and cross-Region VPC peering connections.
2. Update the route tables of VPCs with both the IPv4 and IPv6 CIDR blocks of the peered VPCs, with the respective peering connection ID as a target.
3. With VPC peering, IPv4 and IPv6 CIDR blocks of peered VPCs cannot overlap.

13. Dual stack VPC connectivity with AWS Transit Gateway

Build global dual stack VPC connectivity with AWS Transit Gateway.

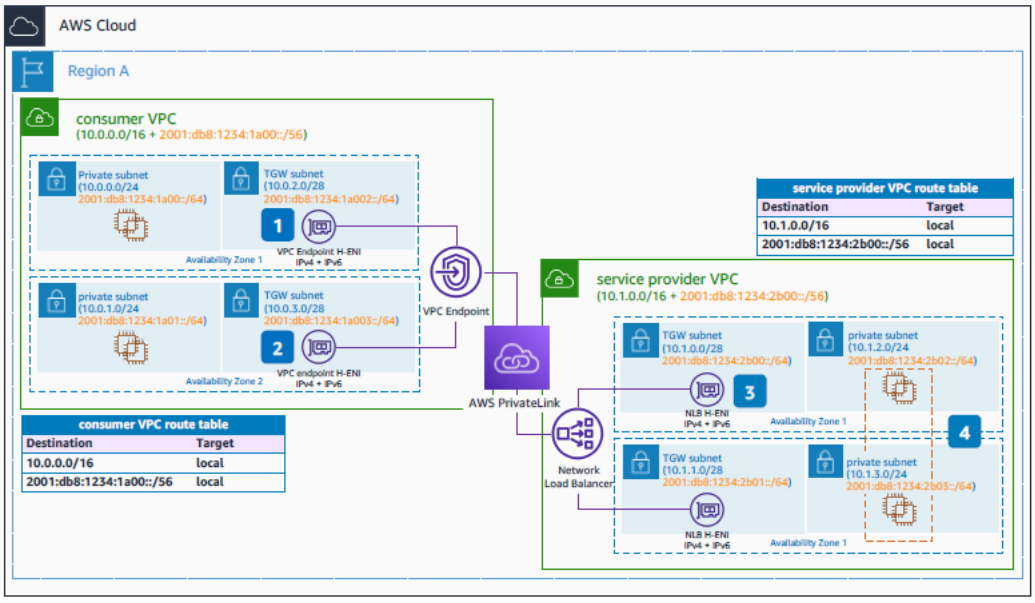


1. When you attach a VPC to an **AWS Transit Gateway**, you must specify one subnet from each Availability Zone to be used by the **Transit Gateway** to route traffic. For dual stack support, the **Transit Gateway** attachment subnets must have IPv6 CIDR blocks associated, and the attachment must be enable for IPv6 routing.

2. **Transit Gateway** peering connections natively support dual stack routing.
3. Update the VPC route tables with the necessary IPv4 and IPv6 routes.
4. If propagation is enabled for the VPC attachments to the transit gateway, the **Transit Gateway** route tables will be populated with both IPv4 and IPv6 VPC CIDR blocks.

14. Dual Stack VPC Connectivity with AWS PrivateLink

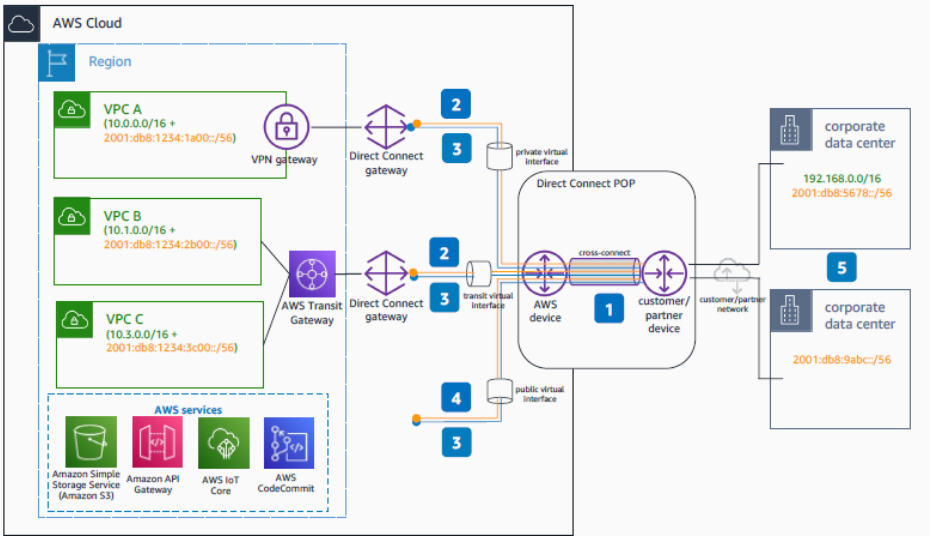
IPv6 support for AWS PrivateLink services and endpoints allows service providers to implement dual stack at the network border, and service consumers and service providers alike can adopt IPv6 at their own pace.



1. When you create **AWS PrivateLink** endpoints in the Consumer VPC, you need to specify the subnets to place them. For high-availability, we recommend you select at least two subnets in different Availability Zones. Those subnets can be IPv4-only, IPv6-only, or dual stack, and you can edit existing endpoints with IPv4 addresses to be dual stack.
2. **PrivateLink** endpoint's ENIs with IPv6 addresses always have **denyAllIGWTraffic** set on creation (this cannot be changed), making them unable to receive traffic from IGW, regardless of routing tables, and therefore private. The **denyAllIGWTraffic** attribute is not changeable after creation for any ENIs.
3. When you create the **Network Load Balancer** in the service provider VPC, its ENIs should be placed in dual stack subnets to enable the **NLB** to receive both IPv4 and IPv6 addresses. Remember that you should also create the **PrivateLink** endpoint service (which ties to the **NLB**) and share it with the consumers.
4. When placing your backend targets in the service provider VPC, you can use IPv4-only, IPv6-only or dual stack subnets. If you want to use IPv6 targets, remember that the target type of the **NLB** target group can only be IP.
    - Note that the **NLB** will use its own IP address if any IP translation is needed (even if client preservation is on): for example if the connection to the **NLB** is used using IPv6 addresses, and the backend use IPv4.

15. Dual stack hybrid connectivity with AWS Direct Connect

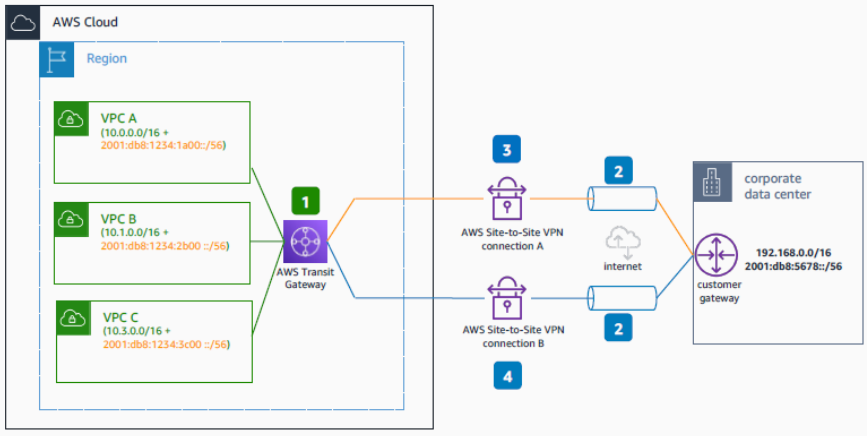
Build hybrid dual stack connectivity with AWS Direct Connect public, private, and transit virtual interfaces.



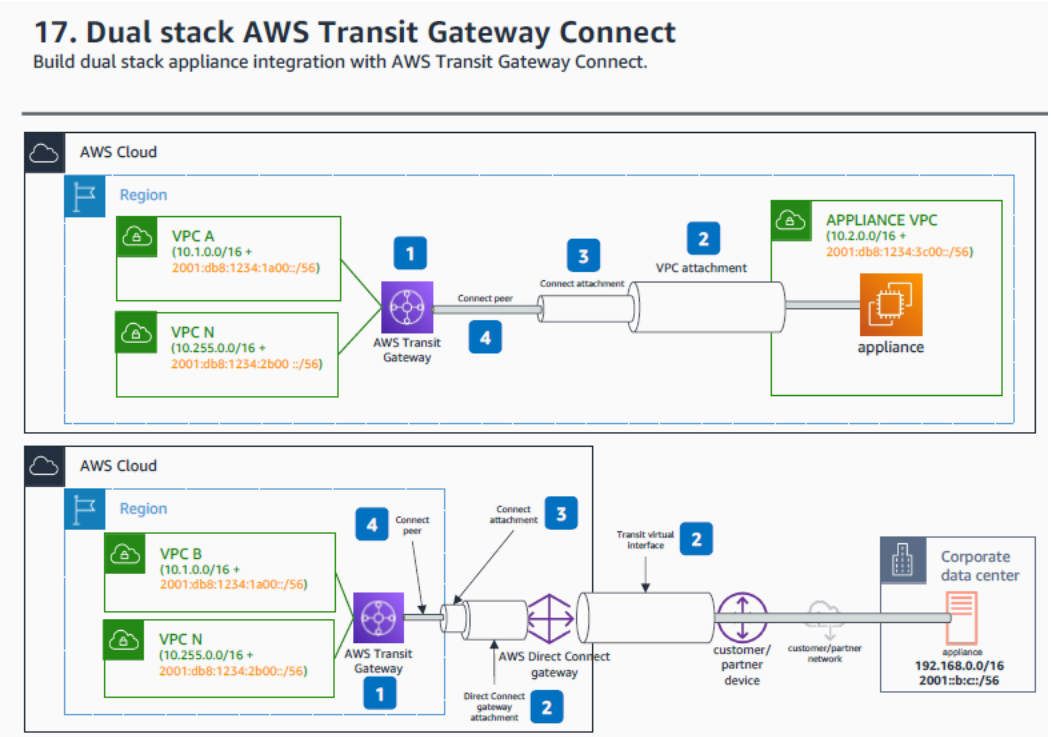
1. **AWS Direct Connect** virtual interfaces (VIFs) support both IPv4 and IPv6 Border Gateway Protocol (BGP) sessions for dual stack operation.
2. For all types of VIFs IPv6 BGP peering, Amazon assigns a /125 CIDR which is not configurable.
3. Private and transit VIFs IPv4 configuration make use of either Amazon-generated private IPv4 addresses, or addresses that you configure. If you specify your own, ensure that you specify private CIDRs for your router interface and the **AWS Direct Connect** interface only. (For example, do not specify other IP addresses from your local network.)
4. For public VIFs IPv4 BGP peering, you must specify unique public /31 IPv4 CIDR that you own, or submit a request to have a CIDR block assigned.
5. You can maintain both IPv6-only and dual stack on premises environment and use **AWS Direct Connect** for dedicated and private connectivity to your dual stack or IPv6-only workload footprint in AWS.

16. Dual stack VPN connectivity with AWS Transit Gateway

Build hybrid dual stack VPN connectivity with AWS Transit Gateway.



1. To configure dual stack support for VPN, create two transit gateway VPN attachments, one for each IP stack.
2. The outer IP addresses of the VPN connections are public IPv4 addresses.
3. One of the VPN tunnels is configured with inner IPv6 addresses, and routes IPv6 traffic.This enables you to maintain IPv6-only in on-premises environments, and configure IPv6-only connectivity with AWS environments, as long as you keep the outer VPN tunnel IPv4 public addresses.
4. The other VPN tunnel is configured with inner IPv4 addresses, routes IPv4 traffic, you require both IPv4 and IPv6 connectivity between your AWS environments and your on-premises workloads.



1. For **Transit Gateway** connect configuration, configure the transit gateway CIDR block. You can specify a size /24 CIDR block or larger for IPv4, or a size /64 CIDR block or larger for IPv6.
2. The VPC attachment to the **Transit Gateway** is dual stack enabled. The **Transit virtual interface (VIF)** can be configured with both IPv4 and IPv6 BGP peers.
3. The **Transit Gateway** connect attachment uses the VPC or **AWS Direct Connect Gateway** attachment of the as transport.
4. When you create the connect peers, you must specify the peer Generic Routing Encapsulation (GRE) address, which can be IPv4 or IPv6. The Border Gateway Protocol (BGP) inside CIDR blocks can be configured for both IPv4 and IPv6. For IPv4 you must specify a /29 CIDR, and for IPv6, a /125 CIDR.

References:
Amazon VPC connectivity options for IPv6 - IPv6 on AWS
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-migrate-ipv6.html#vpc-migrate-ipv6-instance-types
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html
IPV6 calculator: https://www.calculator.net/ip-subnet-calculator.html