NAT gateways

PDF
Kindle
RSS

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

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

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

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

Pricing

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

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

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

Contents

NAT gateway basics

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

If you have resources in multiple Availability Zones and they share one NAT gateway, and if the NAT gateway’s Availability Zone is down, resources in the other Availability Zones lose internet access. To create an Availability Zone-independent architecture, create a NAT gateway in each Availability Zone and configure your routing to ensure that resources use the NAT gateway in the same Availability Zone.

The following characteristics and rules apply to NAT gateways:

- A NAT gateway supports the following protocols: TCP, UDP, and ICMP.
- NAT gateways are supported for IPv4 or IPv6 traffic. For IPv6 traffic, NAT gateway performs NAT64. By using this in conjunction with DNS64 (available on Route 53 resolver), your IPv6 workloads in a subnet in Amazon VPC can communicate with IPv4 resources. These IPv4

services may be present in the same VPC (in a separate subnet) or a different VPC, on your on-premises environment or on the internet.

- A NAT gateway supports 5 Gbps of bandwidth and automatically scales up to 45 Gbps. If you require more bandwidth, you can split your resources into multiple subnets and create a NAT gateway in each subnet.
- A NAT gateway can process one million packets per second and automatically scales up to four million packets per second. Beyond this limit, a NAT gateway will drop packets. To prevent packet loss, split your resources into multiple subnets and create a separate NAT gateway for each subnet.
- A NAT gateway can support up to 55,000 simultaneous connections to each unique destination. This limit also applies if you create approximately 900 connections per second to a single destination (about 55,000 connections per minute). If the destination IP address, the destination port, or the protocol (TCP/UDP/ICMP) changes, you can create an additional 55,000 connections. For more than 55,000 connections, there is an increased chance of connection errors due to port allocation errors. These errors can be monitored by viewing the `ErrorPortAllocation` CloudWatch metric for your NAT gateway. For more information, see Monitor NAT gateways with Amazon CloudWatch.
- You can associate exactly one Elastic IP address with a public NAT gateway. You cannot disassociate an Elastic IP address from a NAT gateway after it's created. To use a different Elastic IP address for your NAT gateway, you must create a new NAT gateway with the required address, update your route tables, and then delete the existing NAT gateway if it's no longer required.
- A private NAT gateway receives an available private IP address from the subnet in which it is configured. You cannot detach this private IP address and you cannot attach additional private IP addresses.
- You cannot associate a security group with a NAT gateway. You can associate security groups with your instances to control inbound and outbound traffic.
- You can use a network ACL to control the traffic to and from the subnet for your NAT gateway. NAT gateways use ports 1024–65535. For more information, see Control traffic to subnets with Network ACLs.
- A NAT gateway receives a network interface that's automatically assigned a private IP address from the IP address range of the subnet. You can view the network interface for the NAT gateway using the Amazon EC2 console. For more information, see Viewing details about a network interface. You cannot modify the attributes of this network interface.
- A NAT gateway cannot be accessed through a ClassicLink connection that is associated with your VPC.

- You cannot route traffic to a NAT gateway through a VPC peering connection, a Site-to-Site VPN connection, or AWS Direct Connect. A NAT gateway cannot be used by resources on the other side of these connections.

Control the use of NAT gateways

By default, IAM users do not have permission to work with NAT gateways. You can create an IAM user policy that grants users permissions to create, describe, and delete NAT gateways. For more information, see Identity and access management for Amazon VPC.

Work with NAT gateways

You can use the Amazon VPC console to create and manage your NAT gateways. You can also use the Amazon VPC wizard to create a VPC with a public subnet, a private subnet, and a NAT gateway. For more information, see VPC with public and private subnets (NAT).

Tasks

Create a NAT gateway

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

To create a NAT gateway

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

a. (Optional) Specify a name for the NAT gateway. This creates a tag where the key is `Name` and the value is the name that you specify.
b. Select the subnet in which to create the NAT gateway.
c. For **Connectivity type**, select **Private** to create a private NAT gateway or **Public** (the default) to create a public NAT gateway.
d. (Public NAT gateway only) For **Elastic IP allocation ID**, select an Elastic IP address to associate with the NAT gateway.
e. (Optional) For each tag, choose **Add new tag** and enter the key name and value.
f. Choose **Create a NAT Gateway**.

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

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

Tag a NAT gateway

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

Cost allocation tags are supported for NAT gateways. Therefore, you can also use tags to organize your AWS bill and reflect your own cost structure. For more information, see Using cost allocation tags in the *AWS Billing and Cost Management User Guide*. For more information about setting up a cost allocation report with tags, see Monthly cost allocation report in *About AWS Account Billing*.

Delete a NAT gateway

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

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

To delete a NAT gateway

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

NAT gateway scenarios

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

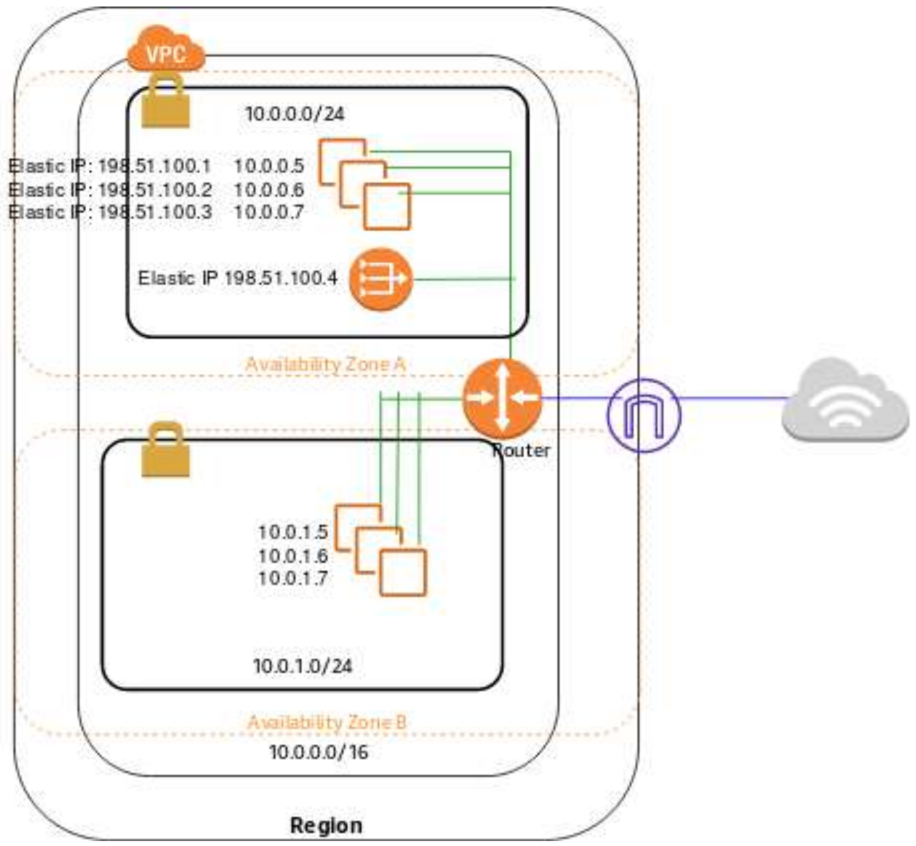
Scenarios

- Access the internet from a private subnet
- Allow access to your network from allow-listed IP addresses

Scenario: Access the internet from a private subnet

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

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



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

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

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

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

Test the public NAT gateway

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

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

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

If these tests fail, see Troubleshoot NAT gateways.

Test the internet connection

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

1. Launch an instance in your public subnet (use this as a bastion host). For more information, see Launch an instance into your subnet. In the launch wizard, ensure that you select an Amazon Linux AMI, and assign a public IP address to your instance. Ensure that your security group rules allow inbound SSH traffic from the range of IP addresses for your local network, and outbound SSH traffic to the IP address range of your private subnet (you can also use `0.0.0.0/0` for both inbound and outbound SSH traffic for this test).
2. Launch an instance in your private subnet. In the launch wizard, ensure that you select an Amazon Linux AMI. Do not assign a public IP address to your instance. Ensure that your security group rules allow inbound SSH traffic from the private IP address of your instance that you launched in the public subnet, and all outbound ICMP traffic. You must choose the same key pair that you used to launch your instance in the public subnet.
3. Configure SSH agent forwarding on your local computer, and connect to your bastion host in the public subnet. For more information, see To configure SSH agent forwarding for Linux or macOS or To configure SSH agent forwarding for Windows (PuTTY).

4. From your bastion host, connect to your instance in the private subnet, and then test the internet connection from your instance in the private subnet. For more information, see To test the internet connection.

To configure SSH agent forwarding for Linux or macOS

1. From your local machine, add your private key to the authentication agent.

   For Linux, use the following command.

   ```
   ssh-add -c mykeypair.pem
   ```

   For macOS, use the following command.

   ```
   ssh-add -K mykeypair.pem
   ```
2. Connect to your instance in the public subnet using the `-A` option to enable SSH agent forwarding, and use the instance's public address, as shown in the following example.

   ```
   ssh -A ec2-user@54.0.0.123
   ```

To configure SSH agent forwarding for Windows (PuTTY)

1. Download and install Pageant from the PuTTY download page, if not already installed.
2. Convert your private key to .ppk format. For more information, see Converting your private key using PuTTYgen in the *Amazon EC2 User Guide for Linux Instances*.
3. Start Pageant, right-click the Pageant icon on the taskbar (it may be hidden), and choose **Add Key**. Select the .ppk file that you created, enter the passphrase if necessary, and choose **Open**.
4. Start a PuTTY session and connect to your instance in the public subnet using its public IP address. For more information, see Connecting to your Linux instance. In the **Auth** category, ensure that you select the **Allow agent forwarding** option, and leave the **Private key file for authentication** box blank.

To test the internet connection

1. From your instance in the public subnet, connect to your instance in your private subnet by using its private IP address as shown in the following example.

   ```
   ssh ec2-user@10.0.1.123
   ```
2. From your private instance, test that you can connect to the internet by running the `ping` command for a website that has ICMP enabled.

   ```
   ping ietf.org
   PING ietf.org (4.31.198.44) 56(84) bytes of data.
   64 bytes from mail.ietf.org (4.31.198.44): icmp_seq=1 ttl=47 time=86.0 ms
   ```

```
64 bytes from mail.ietf.org (4.31.198.44): icmp_seq=2 ttl=47 time=75.6 ms
...
```

   Press **Ctrl+C** on your keyboard to cancel the `ping` command. If the `ping` command fails, see Instances cannot access the internet.

3. (Optional) If you no longer require your instances, terminate them. For more information, see Terminate your instance in the *Amazon EC2 User Guide for Linux Instances*.

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

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

Migrate from a NAT instance to a NAT gateway

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

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

API and CLI overview

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

Create a NAT gateway

- create-nat-gateway (AWS CLI)
- New-EC2NatGateway (AWS Tools for Windows PowerShell)
- CreateNatGateway (Amazon EC2 Query API)

Describe a NAT gateway

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

Tag a NAT gateway

- create-tags (AWS CLI)
- New-EC2Tag (AWS Tools for Windows PowerShell)
- CreateTags (Amazon EC2 Query API)

Delete a NAT gateway

- delete-nat-gateway (AWS CLI)
- Remove-EC2NatGateway (AWS Tools for Windows PowerShell)
- DeleteNatGateway (Amazon EC2 Query API)

DNS64 and NAT64

NAT gateway supports network address translation from IPv6 to IPv4, popularly known as NAT64. NAT64 helps your IPv6 AWS resources communicate with IPv4 resources in the same VPC or a different VPC, in your on-premises network or over the internet. You can use NAT64 with DNS64 on Amazon Route 53 Resolver or use your own DNS64 server.

Contents

What is DNS64?

Your IPv6-only workloads running in VPCs can only send and receive IPv6 network packets. Without DNS64, a DNS query for an IPv4-only service will yield an IPv4 destination address in response and your IPv6-only service 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. With DNS64, the Amazon Route 53 Resolver looks up the DNS record for the service you queried for and does one of the following:

- If the record contains an IPv6 address, it returns the original record and the connection is established without any translation over IPv6.
- If there is no IPv6 address associated with the destination in the DNS record, the Route 53 Resolver synthesizes one by prepending the well-known `/96` prefix, defined in RFC6052 (`64:ff9b::/96`), to the IPv4 address in the record. Your IPv6-only service sends network packets to the synthesized IPv6 address. You will then need to route this traffic through the NAT gateway, which performs the necessary translation on the traffic to allow IPv6 services in your subnet to access IPv4 services outside that subnet.

You can enable or disable DNS64 on a subnet using the modify-subnet-attribute using the AWS CLI or with the VPC console by selecting a subnet and choosing **Actions** > **Modify DNS64 settings**.

What is NAT64?

NAT64 enables your IPv6-only services in Amazon VPCs to communicate with IPv4-only services within the same VPC (in different subnets) or connected VPCs, in your on-premises networks, or over the internet.

NAT64 is automatically available on your existing NAT gateways or on any new NAT gateways you create. It's not a feature you enable or disable.

Once you have enabled DNS64 and your IPv6-only service sends network packets to the synthesized IPv6 address through the NAT gateway, the following happens:

- From the `64:ff9b::/96` prefix, the NAT gateway recognizes that the original destination is IPv4 and translates the IPv6 packets to IPv4 by replacing:
  - Source IPv6 with its own private IP which is translated to Elastic IP address by the internet gateway.
  - Destination IPv6 to IPv4 by truncating the `64:ff9b::/96` prefix.
- The NAT gateway sends the translated IPv4 packets to the destination through the internet gateway, virtual private gateway, or transit gateway and initiates a connection.
- The IPv4-only host sends back IPv4 response packets. Once a connection is established, NAT gateway accepts the response IPv4 packets from the external hosts.
- The response IPv4 packets are destined for NAT gateway, which receives the packets and de-NATs them by replacing its IP (destination IP) with the host's IPv6 address and prepending back `64:ff9b::/96` to the source IPv4 address. The packet then flows to the host following the local route.

In this way, the NAT gateway enables your IPv6-only workloads in an Amazon VPC subnet to communicate with IPv4-only services anywhere outside the subnet.

Configure DNS64 and NAT64

Follow the steps in this section to configure DNS64 and NAT64 to enable communication with IPv4-only services.

Contents

Enable communication with IPv4-only services on the Internet with the AWS CLI

If you have a subnet with IPv6-only workloads that needs to communicate with IPv4-only services outside the subnet, this example shows you how to enable these IPv6-only services to communicate with IPv4-only services on the Internet.

You should first configure a NAT gateway in a public subnet (separate from the subnet containing the IPv6-only workloads). For example, the subnet containing the NAT gateway should have a `0.0/0` route pointing to the internet gateway.

Complete these steps to enable these IPv6-only services to connect with IPv4-only services on the Internet:

1. Add the following three routes to the route table of the subnet containing the IPv6-only workloads:

- IPv4 route (if any) pointing to the NAT gateway.
- `64:ff9b::/96` route pointing to the NAT gateway. This will allow traffic from your IPv6-only workloads destined for IPv4-only services to be routed through the NAT gateway.
- IPv6 `::/0` route pointing to the egress-only internet gateway (or the internet gateway).

Note that pointing `::/0` to the internet gateway will allow external IPv6 hosts (outside the VPC) to initiate connection over IPv6.

```
aws ec2 create-route --route-table-id rtb-34056078 --destination-cidr-block
0.0.0.0/0 --nat-gateway-id nat-05dba92075d71c408
aws ec2 create-route --route-table-id rtb-34056078 --destination-ipv6-cidr-
block
64:ff9b::/96 --nat-gateway-id nat-05dba92075d71c408
aws ec2 create-route --route-table-id rtb-34056078 --destination-ipv6-cidr-
block
::/0 --egress-only-internet-gateway-id eigw-c0a643a9
```

2. Enable DNS64 capability in the subnet containing the IPv6-only workloads.

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --enable-dns64
```

Now, resources in your private subnet can establish stateful connections with both IPv4 and IPv6 services on the internet. Configure your security group and NACLs appropriately to allow egress and ingress traffic to `64:ff9b::/96` traffic.

Enable communication with IPv4-only services in your on-premises environment

Amazon Route 53 Resolver enables you to forward DNS queries from your VPC to an on-premises network and vice versa. You can do this by doing the following:

- You create a Route 53 Resolver outbound endpoint in a VPC and assign it the IPv4 addresses that you want Route 53 Resolver to forward queries from. For your on-premises DNS resolver, these are the IP addresses that the DNS queries originate from and, therefore, should be IPv4 addresses.

- You create one or more rules which specify the domain names of the DNS queries that you want Route 53 Resolver to forward to your on-premises resolvers. You also specify the IPv4 addresses of the on-premises resolvers.
- Now that you have set up a Route 53 Resolver outbound endpoint, you need to enable DNS64 on the subnet containing your IPv6-only workloads and route any data destined for your on-premises network through a NAT gateway.

How DNS64 works for IPv4-only destinations in on-premises networks:

1. You assign an IPv4 address to the Route 53 Resolver outbound endpoint in your VPC.
2. The DNS query from your IPv6 service goes to Route 53 Resolver over IPv6. Route 53 Resolver matches the query against the forwarding rule and gets an IPv4 address for your on-premises resolver.
3. Route 53 Resolver converts the query packet from IPv6 into IPv4 and forwards it to the outbound endpoint. Each IP address of the endpoint represents one ENI that forwards the request to the on-premises IPv4 address of your DNS resolver.
4. The on-premises resolver sends the response packet over IPv4 back through the outbound endpoint to Route 53 Resolver.
5. Assuming the query was made from a DNS64-enabled subnet, Route 53 Resolver does two things:

a. Checks the content of the response packet. If there’s an IPv6 address in the record, it keeps the content as is, but if it contains only an IPv4 record. It synthesizes an IPv6 record as well by prepending `64:ff9b::/96` to the IPv4 address.

b. Repackages the content and sends it to the service in your VPC over IPv6