aws

User Guide

Amazon VPC Lattice

Amazon VPC Lattice: User Guide

Table of Contents

What is Amazon VPC Lattice?

Amazon VPC Lattice is a fully managed application networking service that you use to connect, secure, and monitor the services and resources for your application. You can use VPC Lattice with a single virtual private cloud (VPC) or across multiple VPCs from one or more accounts.

Modern applications can consist of multiple small and modular components which are often called *microservices*, such as an HTTP API, resources such as databases, and custom resources consisting of DNS and IP address endpoints. While modernization has its advantages, it can also introduce networking complexities and challenges when you connect these microservices and resources. For example, if the developers are spread across different teams, they might build and deploy microservices and resources across multiple accounts or VPCs.

In VPC Lattice, we refer to a microservice as a *service* and represent a resource only as a *resource configuration*. These are the terms that you see in the VPC Lattice user guide.

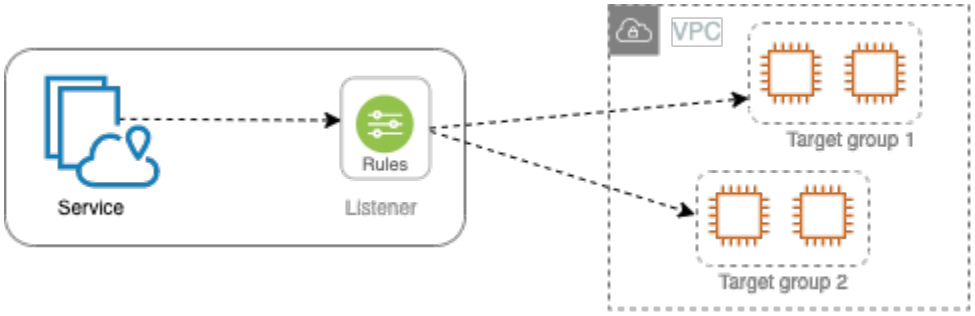
Contents

Key components

To use Amazon VPC Lattice, you should be familiar with its key components.

Service

An independently deployable unit of software that delivers a specific task or function. A service can run on EC2 instances or ECS/EKS/Fargate containers, or as Lambda functions, within an account or a virtual private cloud (VPC). A VPC Lattice service has the following components: target groups, listeners, and rules.



Target group

A collection of resources, also known as targets, that run your application or service. These are similar to the target groups provided by Elastic Load Balancing, but they are not interchangeable. The supported target types include EC2 instances, IP addresses, Lambda functions, Application Load Balancers, Amazon ECS tasks, and Kubernetes Pods.

Listener

A process that checks for connection requests, and routes them to targets in a target group. You configure a listener with a protocol and a port number.

Rule

A default component of a listener that forwards requests to the targets in a VPC Lattice target group. Each rule consists of a priority, one or more actions, and one or more conditions. Rules determines how the listener routes client requests.

Resource

A resource is an entity such as an Amazon Relational Database Service (Amazon RDS) database, an Amazon EC2 instance, an application endpoint, a domain-name target, or an IP address. You can share a resource in your VPC by creating a resource share in AWS Resource Access Manager (AWS RAM), creating a resource gateway, and defining a resource configuration.

Resource gateway

A resource gateway is a point of ingress into the VPC in which resources reside.

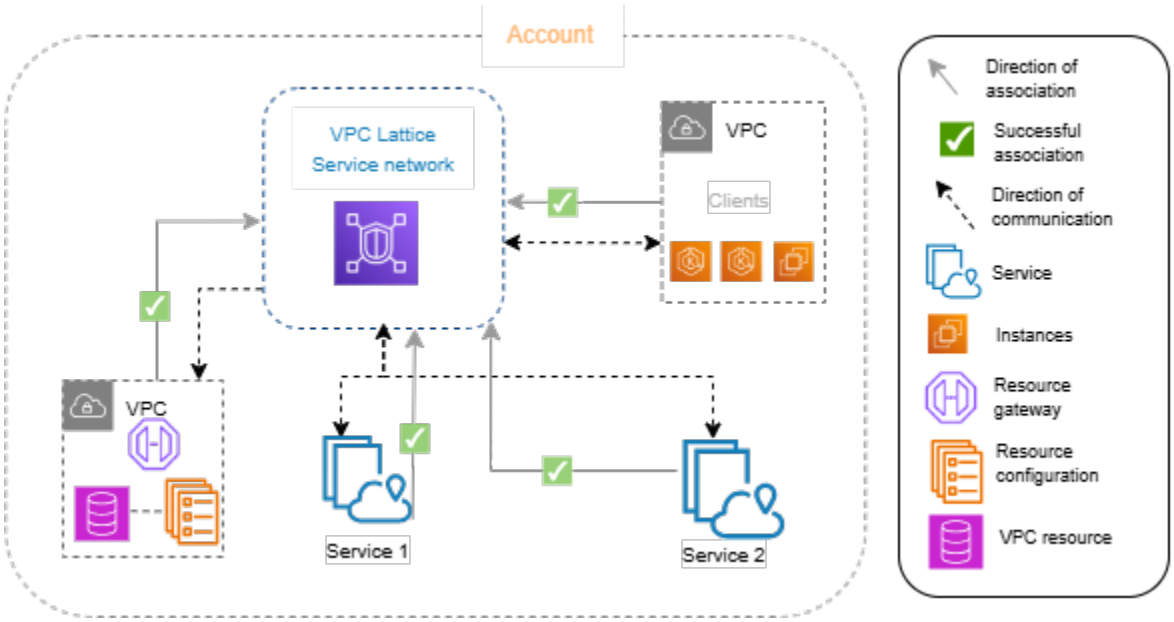
Resource configuration

A resource configuration is a logical object that represents either a single resource or a group of resources. A resource can be an IP address, a domain-name target, or an Amazon RDS database.

Service network

A logical boundary for a collection of services and resource configurations. A client can be in a VPC that is associated with the service network. Clients and services that are associated with the same service network can communicate with each other if they are authorized to do so.

In the following figure, the clients can communicate with both services, because the VPC and services are associated with the same service network.



Service directory

A central registry of all VPC Lattice services that you own or are shared with your account through AWS RAM.

Auth policies

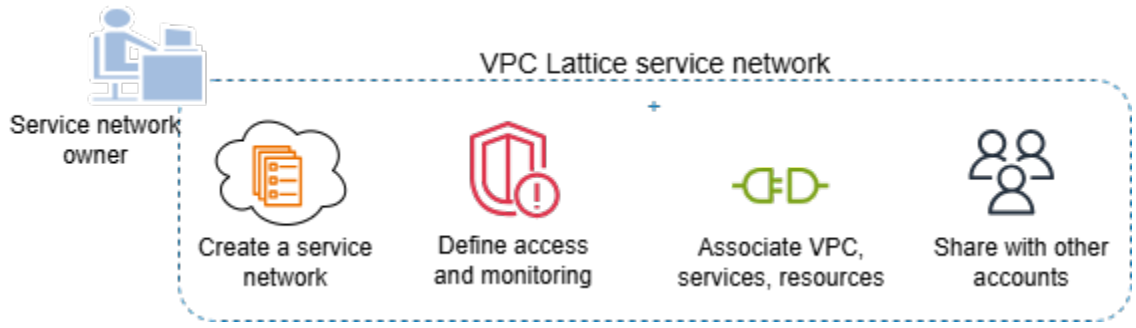
Fine-grained authorization policies that can be used to define access to services. You can attach separate auth policies to individual services or to the service network. For example, you can create a policy for how a payment service running on an auto scaling group of EC2 instances should interact with a billing service running in AWS Lambda.

Auth-policies are not supported on resource configurations. Auth policies of a service-network are not applicable to resource configurations in the service network.

Roles and responsibilities

A role determines who is responsible for the setup and flow of information within Amazon VPC Lattice. There are typically two roles, service network owner and service owner, and their responsibilities can overlap.

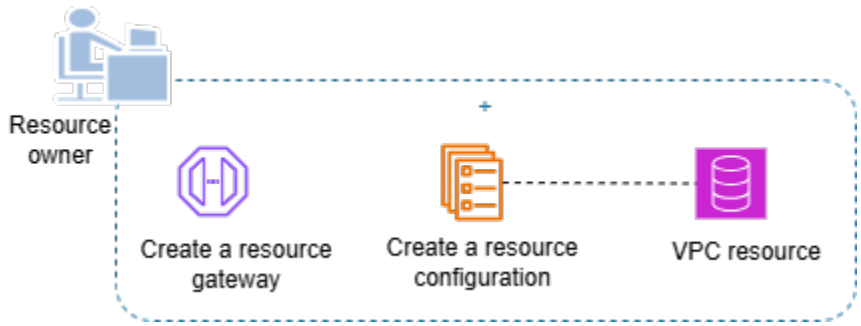
Service network owner – The service network owner is usually the network administrator or the cloud administrator in an organization. Service network owners create, share, and provision the service network. They also manage who can access the service network or services within VPC Lattice. The service network owner can define coarse-grained access settings for the services associated with the service network. These controls are used to manage communication between clients and services using authentication and authorization policies. The service network owner can also associate a service or resource configuration with a single or multiple service networks, if the service or resource configuration is shared with the service network owner's account.



Service owner – The service owner is usually a software developer in an organization. Service owners create services within VPC Lattice, define routing rules, and also associate services with the service network. They can also define fine-grained access settings, which can restrict access to only authenticated and authorized services and clients.

Resource owner – The resource owner is usually a software developer in an organization and serves as an admin for a resource such as a database. The resource owner creates a resource

configuration for the resource, defines access-settings for the resource configuration, and associates the resource configuration with service networks.



Features

The following are the core features that VPC Lattice provides.

Service discovery

All clients and services in VPCs associated with the service network can communicate with other services within the same service network. DNS directs client-to-service and service-to-service traffic through the VPC Lattice endpoint. When a client wants to send a request to a service, it uses the service's DNS name. The Route 53 Resolver sends the traffic to VPC Lattice, which then identifies the destination service.

Connectivity

Client-to-service and client-to-resource connectivity is established within the AWS network infrastructure. When you associate a VPC with the service network, any client within the VPC can connect with services and resources (through resource configurations) in the service network, if they have the required access. VPC Lattice supports overlapping CIDR technology.

On premise access

You can enable connectivity to a service network from a VPC using a VPC endpoint (powered by AWS PrivateLink). A VPC endpoint of type *service network* lets you enable access to services and resources in the service network from on premises networks over Direct Connect and VPN. Traffic that traverses VPC peering or AWS Transit Gateway can also access resources and services over a VPC endpoint.

Observability

VPC Lattice generates metrics and logs for each request and response traversing the service network, to help you monitor and troubleshoot applications. By default, metrics are published

to the service owner account. Service owners and resource owners have the option to turn on logging, and receive logs for all client access/requests to their services and resources. Service network owners can also turn on logging on the service network, to log all access/requests to the services and resources from clients in VPCs that are connected to the service network.

VPC Lattice works with the following tools to help you monitor and troubleshoot your services: Amazon CloudWatch log groups, Firehose delivery streams, and Amazon S3 buckets.

Security

VPC Lattice provides a framework that you can use to implement a defense strategy at multiple layers of the network. The first layer is the combination of service, resource configuration, VPC association, and VPC endpoint of type service network. Without a VPC and a service association or a VPC endpoint of type service network, clients cannot access services. Similarly, without a VPC and a resource configuration and a service association or a VPC endpoint of type service network, clients cannot access resources.

The second layer enables users to attach security groups to the association between the VPC and the service network. The third and fourth layers are auth policies that can be applied individually at the service network level and the service level.

Accessing VPC Lattice

You can create, access, and manage VPC Lattice using any of the following interfaces:

- **AWS Management Console** – Provides a web interface that you can use to access VPC Lattice.
- **AWS Command Line Interface (AWS CLI)** – Provides commands for a broad set of AWS services, including VPC Lattice. The AWS CLI is supported on Windows, MacOS, and Linux. For more information about the CLI, see AWS Command Line Interface. For more information about the APIs, see Amazon VPC Lattice API Reference.
- **VPC Lattice Controller for Kubernetes** – Manages VPC Lattice resources for a Kubernetes cluster. For more information about using VPC Lattice with Kubernetes, see the AWS Gateway API Controller User Guide.
- **AWS CloudFormation** – Helps you to model and set up your AWS resources. For more information, see the Amazon VPC Lattice resource type reference.

VPC Lattice service endpoints

An endpoint is a URL that serves as an entry point for an AWS web service. VPC Lattice supports the following endpoint types:

- the section called "IPv4 endpoints"
- Dualstack endpoints (support both IPv4 and IPv6)

When you make a request, you can specify the endpoint to use. If you do not specify an endpoint, the IPv4 endpoint is used by default. To use a different endpoint type, you must specify it in your request. For examples of how to do this, see the section called "Specifying endpoints". For a table of available endpoints, see Amazon VPC Lattice endpoints.

IPv4 endpoints

IPv4 endpoints support IPv4 traffic only. IPv4 endpoints are available for all Regions.

If you specify the general endpoint, `vpc-lattice.amazonaws.com`, we use the endpoint for `us-east-1`. To use a different Region, specify its associated endpoint. For example, if you specify `vpc-lattice.us-east-2.amazonaws.com` as the endpoint, we direct your request to the `us-east-2` endpoint.

IPv4 endpoint names use the following naming convention:

- `vpc-lattice.`*`region`*`.amazonaws.com`

For example, the IPv4 endpoint name for the `eu-west-1` Region is `vpc-lattice.eu-west-1.amazonaws.com`.

Dualstack (IPv4 and IPv6) endpoints

Dualstack endpoints support both IPv4 and IPv6 traffic. Dualstack endpoints are available for all Regions. When you make a request to a dualstack endpoint, the endpoint URL resolves to an IPv6 or an IPv4 address, depending on the protocol used by your network and client.

Dual-stack endpoint names use the following naming convention:

- `vpc-lattice.`*`region`*`.api.aws`

For example, the dual-stack endpoint name for the `eu-west-1` Region is `vpc-lattice.eu-west-1.api.aws`.

Specifying endpoints

The following examples show how to specify an endpoint for the `us-east-2` Region using the AWS CLI for `vpc-lattice`.

- **IPv4**

```
aws vpc-lattice get-service --service-identifier svc-0285b53b2eEXAMPLE --region us-east-2 --endpoint-url https://vpc-lattice.us-east-2.amazonaws.com
```

- **Dualstack**

```
aws vpc-lattice get-service --service-identifier svc-0285b53b2eEXAMPLE --region us-east-2 --endpoint-url https://vpc-lattice.us-east-2.api.aws
```

Pricing

With VPC Lattice you pay for the time that a service is provisioned, the amount of data transferred through each service, and the number of requests. As a resource owner, you pay for the data transferred to and from each resource. As a service network owner, you pay hourly for resource configurations associated to your service network. As a consumer who has a VPC associated to a service network, you pay for data transferred to and from resources in the service network from your VPC. For more information, see Amazon VPC Lattice Pricing.

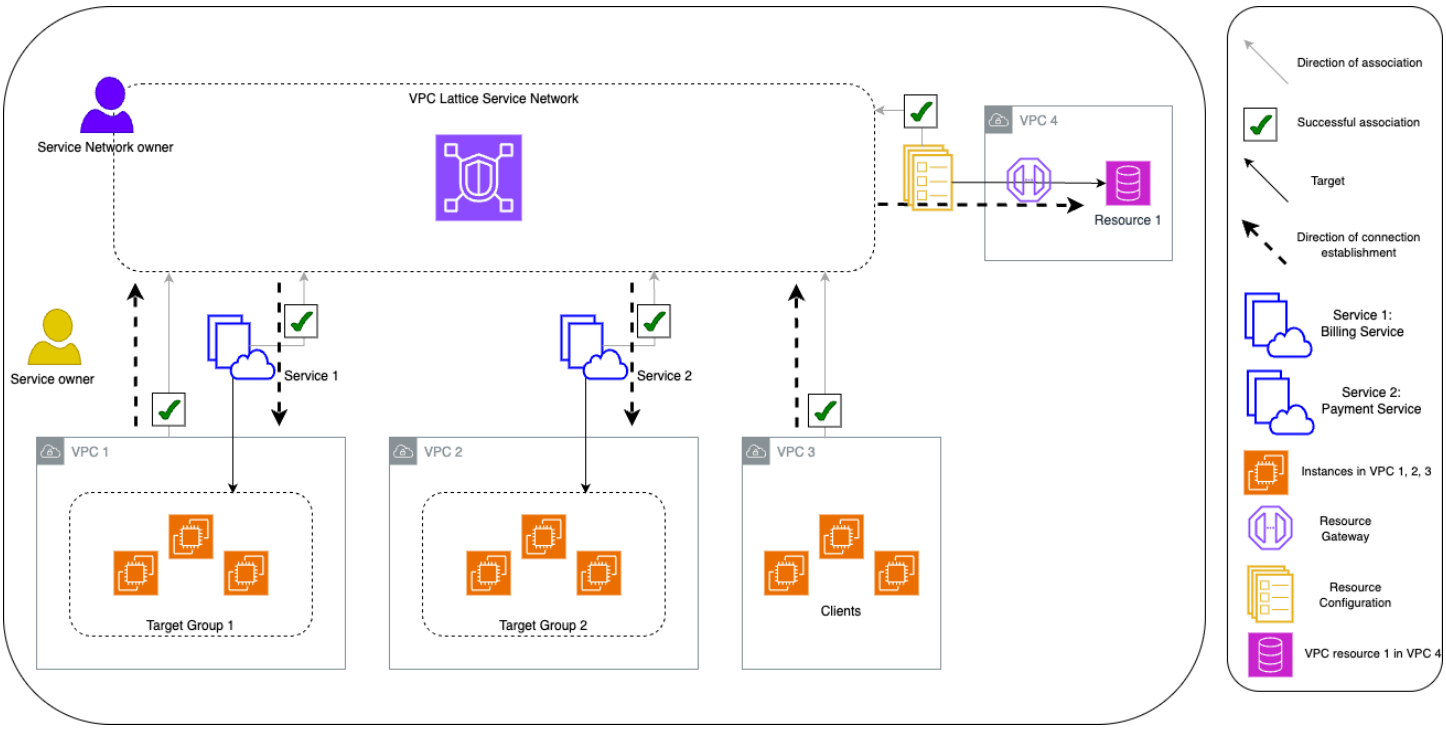
How VPC Lattice works

VPC Lattice is designed to help you easily and effectively discover, secure, connect, and monitor all of the services and resources within it. Each component within VPC Lattice communicates unidirectionally or bi-directionally within the service network based on its association with the service network and its access settings. Access settings are comprised of authentication and authorization policies required for this communication.

The following summary describes communication between components within VPC Lattice:

- There are two ways a VPC can be connected to a service network - through a VPC association and through a VPC endpoint of type service network.
- Services and resources that are associated with the service network can receive requests from clients whose VPCs are also connected to the service network.
- A client can send requests to services and resources associated with a service network only if it's in a VPC that's connected to the same service network. Client traffic that traverses a VPC peering connection, a transit gateway, Direct Connect, or VPN can reach resources and services only if the VPC is connected to the service network through a VPC endpoint.
- Targets of services in VPCs that are associated with the service network are also clients and can send requests to other services and resources associated with the service network.
- Targets of services in VPCs that aren't associated with the service network aren't clients and can't send requests to other services and resources associated with the service network.
- Clients in VPCs that have resources but where the VPC isn’t associated with the service network aren't clients and can't send requests to other services and resources associated with the service network.

The following flow diagram uses an example scenario to explain the flow of information and direction of communication between the components within VPC Lattice. There are two services associated with a service network. Both services and all VPCs were created in the same account as the service network. Both services are configured to allow traffic from the service network.



Service 1 is a billing application running on a group of instances registered with target group 1 in VPC 1. Service 2 is a payment application running on a group of instances registered with target group 2 in VPC 2. VPC 3 is in the same account, and it has clients but no services. Resource 1 is a database that has customer data in VPC 4.

The following list describes, in order, the typical workflow of tasks for VPC Lattice.

1. **Create a service network**

   The service network owner creates the service network.

2. **Create a service**

   The service owners create their respective services, service 1 and service 2. During creation, the service owner adds listeners and defines rules for routing requests to the target group for each service.

3. **Define routing**

   The service owners create the target group for each service (target group 1 and target group 2). They do this by specifying the target instances on which the services run. They also specify the VPCs in which these targets reside.

In the preceding diagram, the solid arrows represent services routing traffic to target groups, and resource configurations routing to resources.

4. **Associate services with the service network**

   The service network owner or the service owner associates the services with the service network. The associations are shown as arrows with check marks pointing to the service network from the service. When you associate a service with a service network, that service becomes discoverable to other services associated with the service network and clients in VPCs connected to the service network.

   The dashed arrows between the service network and target groups show the direction of connection establishment. Return traffic flows back to clients using the service network. The arrows representing the returning traffic aren't included in this diagram.

5. **Create a resource gateway**

   The resource owner creates a resource gateway in VPC 4 in order to be able to enable connectivity from clients to resource 1.

6. **Create a resource configuration**

   The resource owner creates a resource configuration to represent resource 1 and specifies the resource gateway for resource 1.

7. **Associate resource configurations with the service network**

   The service network owner or the resource owner associates the resource configuration with the service network. The association is shown as an arrow with a check mark pointing to the service network from the resource configuration. When you associate a resource configuration with a service network, that resource configuration becomes discoverable to other services associated with the service network and clients in the VPCs connected to the service network.

   The dashed arrows from the service network to the resource represent the resource receiving requests from clients. Return traffic flows back to the client using the service network. The arrows representing the returning traffic aren't included in this diagram.

8. **Connect VPCs with the service network**

   VPCs can be connected with the service network in two ways - by associating the VPC to service network, or by creating a VPC endpoint. Here, the service network owner associates VPC 1 and VPC 3 with the service network. The associations are shown using arrows with check marks

pointed to the service network. With these associations, any resources in the VPC can act as clients, and can make requests to services within the service network. The dashed arrows between VPC 1 and the service network show the direction of connection establishment. The service network only initiates connections towards resources targeted by service 1 target groups. Any resource in VPC 1 can act as a client and initiate connections to the service network services and resources.

VPC 2 does not have an arrow or a check mark that represents an association. This means that the service network owner or the service owner hasn't associated VPC 2 with the service network. This is because service 2, in this example, only needs to receive requests and send responses using the same request. In other words, the targets for service 2 aren't clients and don't need to make requests to other services in the service network.

Similarly, VPC 4 does not have an arrow or a check mark that represents an association. This means that the service network owner or the resource owner hasn't associated VPC 4 with the service network. This is because resource 1 only receives requests and send responses using the same request. It cannot make requests to other services and resources in the service network.

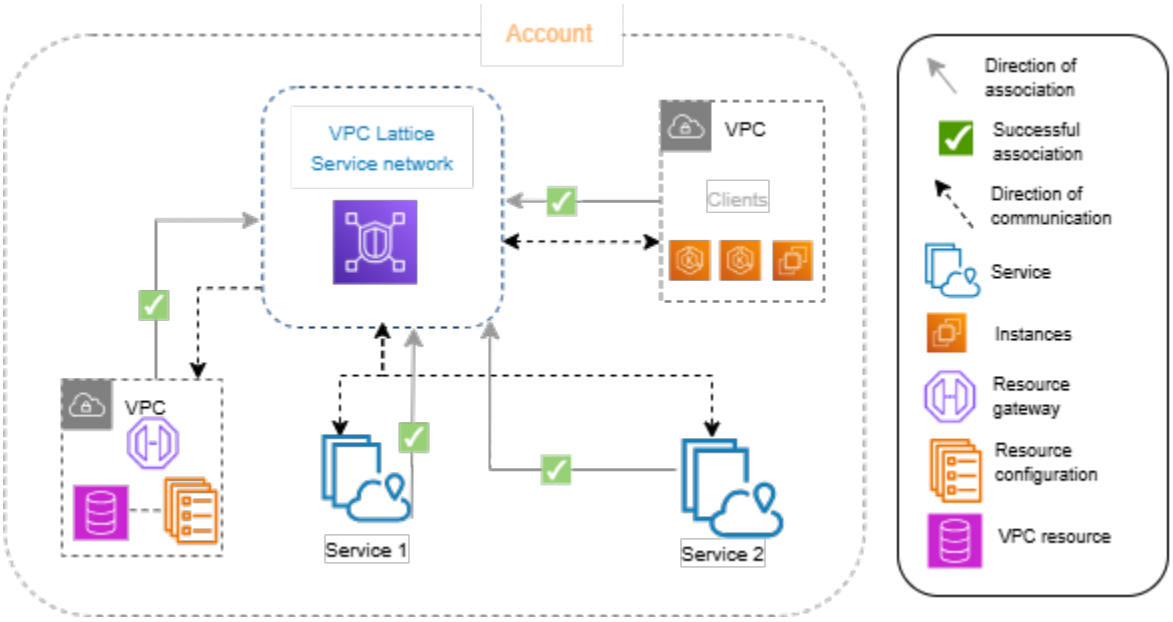
In summary, the proceeding diagram showed the following scenarios:

- VPCs with ingress only connections from VPC Lattice to their resources. VPC 2 and VPC 4 represent these scenarios.
- A VPC with egress only connections from their resources to VPC Lattice. VPC 3 represents this scenario.
- A VPC with ingress connections from VPC Lattice to their resources and with egress connections from their resources to VPC Lattice. VPC 1 represents this scenario.

Service networks in VPC Lattice

A *service network* is a logical boundary for a collection of services and resource configurations. Services and resource configurations associated with the network can be authorized for discovery, connectivity, accessibility, and observability. To make requests to services and resource configurations in the network, your service or client must be in a VPC that is connected to the service network either through an association or through a VPC endpoint.

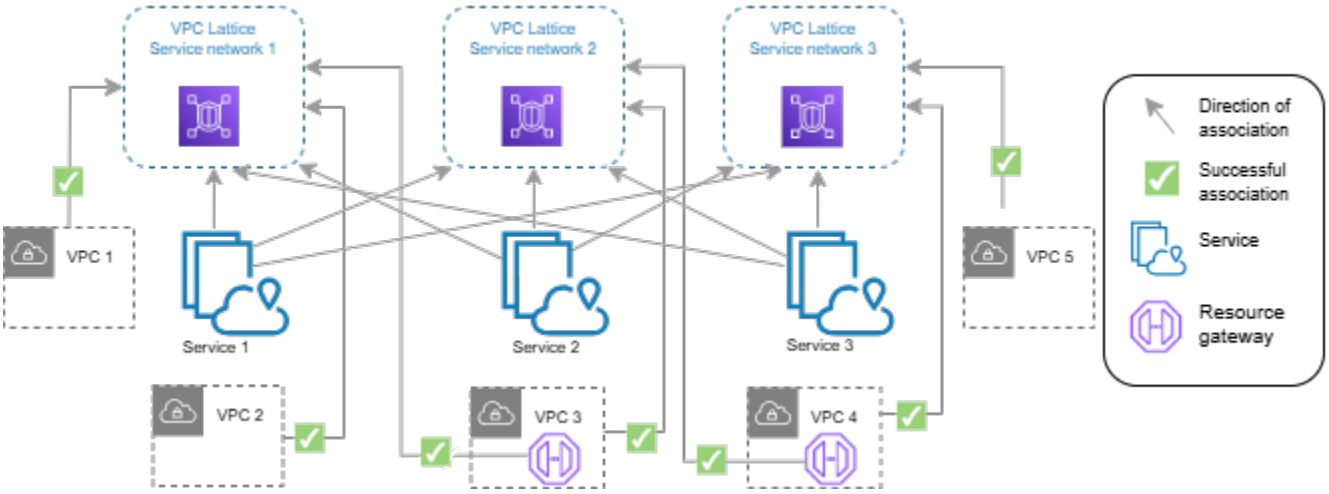
The following diagram shows the key components of a typical service network within Amazon VPC Lattice. Check marks on the arrows indicate that the services and the VPC are associated with the service network. Clients in the VPC associated with the service network can communicate with both services through the service network.



You can associate one or more services and resource configurations with multiple service networks. You can also connect multiple VPCs with one service network. You can connect a VPC to only one service network through an association. To connect a VPC to multiple service networks, you can use VPC endpoints of type service network. For more information on VPC endpoints of type service network, see the *AWS PrivateLink user guide*.

In the following diagram, the arrows represent the associations between services and service networks, as well as associations between the VPCs and service networks. You can see that multiple services are associated to multiple service networks, and multiple VPCs are associated to each service network. Each VPC has exactly one association to a service network. VPC 3 and VPC 4

however connect to two service-networks. VPC 3 connects to service-network 1 through a VPC endpoint. Similarly, VPC 4 connects to service-network 2 through a VPC endpoint.



For more information, see Quotas for Amazon VPC Lattice.

Contents

Create a VPC Lattice service network

Use the console to create a service network and optionally configure it with services, associations, access settings, and access logs.

To create a service network using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.
3. Choose **Create service network**.

4. For **Identifiers**, enter a name, an optional description, and optional tags. The name must be between 3 and 63 characters. You can use lowercase letters, numbers, and hyphens. The name must begin and end with a letter or number. Do not use consecutive hyphens. The description can have up to 256 characters. To add a tag, choose **Add new tag** and specify a tag key and tag value.
5. (Optional) To associate a service, choose the service from **Service associations**, **Services**. The list includes services that are in your account and any services that are shared with you from a different account. If there aren't any services in the list, you can create a service by choosing **Create an VPC Lattice service**.

   Alternatively, to associate a service after you've created the service network, see the section called “Manage service associations”.
6. (Optional) To associate a resource configuration, choose the resource configuration service from **Resource Configuration associations**, **Resource configuration**. The list includes resource configurations that are in your account and any resource configurations that are shared with you from a different account. If there aren't any resource configurations in the list, you can create a resource configuration by choosing **Create an Amazon VPC Lattice resource configuration**.

   Alternatively, to associate a resource configuration after you've created the service network, see the section called “Manage resource configuration associations”.
7. (Optional) To associate a VPC, choose **Add VPC association**. Select the VPC to associate from **VPC**, and select up to five security groups from **Security groups**. To create a security group, choose **Create new security group**.

   Alternatively, you can skip this step and connect a VPC to the service network using a VPC endpoint (powered by AWS PrivateLink). For more information, see Access service networks in the *AWS PrivateLink user guide*.
8. When creating a service network, you have to decide if you intend sharing the service network with other accounts or not. Your selection is immutable and cannot be changed after you create the service network. If you choose to allow sharing, the service network can be shared with other accounts through AWS Resource Access Manager.

   To share your service network with other accounts, choose the AWS RAM resource shares from **Resource shares**.

   To create a resource share, go to the AWS RAM console and choose **Create a resource share**.

9. For **Network access**, you can leave the default auth type, **None**, if you want the clients in the associated VPCs to access the services in this service network. To apply an auth policy to control access to your services, choose **AWS IAM** and do one of the following for **Auth policy**:
   - Enter a policy in the input field. For example policies that you can copy and paste, choose **Policy examples**.
   - Choose **Apply policy template** and select the **Allow authenticated and unauthenticated access** template. This template allows a client from another account to access the service either by signing the request (meaning authenticated) or anonymously (meaning unauthenticated).
   - Choose **Apply policy template** and select the **Allow only authenticated access** template. This template allows a client from another account to access the service only by signing the request (meaning authenticated).
10. (Optional) To turn on access logs, select the **Access logs** toggle switch and specify a destination for your access logs as follows:
    - Select **CloudWatch Log group** and choose a CloudWatch Log group. To create a log group, choose **Create a log group in CloudWatch**.
    - Select **S3 bucket** and enter the S3 bucket path, including any prefix. To search your S3 buckets, choose **Browse S3**.
    - Select **Kinesis Data Firehose delivery stream** and choose a delivery stream. To create a delivery stream, choose **Create a delivery stream in Kinesis**.
11. (Optional) To share your service network with other accounts, choose the AWS RAM resource shares from **Resource shares**. To create a resource share, choose **Create a resource share in RAM console**.
12. Review your configuration in the **Summary** section, and then choose **Create service network**.

To create a service network using the AWS CLI

Use the create-service-network command. This command creates only the basic service network. To create a fully functional service network, you must also use the commands that create service associations, VPC associations, and access settings.

Manage the associations for a VPC Lattice service network

When you associate a service or a resource configuration with the service network, it enables clients in VPCs connected to the service network, to make requests to the service and resource

configuration. When you connect a VPC with the service network, it enables all the targets within that VPC to be clients and communicate with other services and resource configurations in the service network.

Contents

Manage service associations

You can associate services that reside in your account or services that are shared with you from different accounts. This is an optional step while creating a service network. However, a service network is not fully functional until you associate a service. Service owners can associate their services to a service network if their account has the required access. For more information, see Identity-based policy examples for VPC Lattice.

When you delete a service association, the service can no longer connect to other services in the service network.

To manage service associations using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.
3. Select the name of the service network to open its details page.
4. Choose the **Service associations** tab.
5. To create an association, do the following:
    a. Choose **Create associations**.
    b. Select a service from **Services**. To create a service, choose **Create an Amazon VPC Lattice service**.
    c. (Optional) To add a tag, expand **Service association tags**, choose **Add new tag**, and enter a tag key and tag value.
    d. Choose **Save changes**.

6. To delete an association, select the check box for the association and then choose **Actions**, **Delete service associations**. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To create a service association using the AWS CLI

Use the create-service-network-service-association command.

To delete a service association using the AWS CLI

Use the delete-service-network-service-association command.

Manage resource configuration associations

A resource configuration is a logical object that represents either a single resource or a group of resources. You can associate resource configurations that reside in your account or resource configurations that are shared with you from different accounts. This is an optional step while creating a service network. Resource configuration owners can associate their resource configurations to a service network if their account has the required access. For more information, see Identity-based policy examples for VPC Lattice.

Manage associations between service networks and resource configurations

You can create or delete the association between the service network and resource configuration.

To manage resource configuration associations using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **PrivateLink and Lattice**, choose **Service networks**.
3. Select the name of the service network to open its details page.
4. Choose the **Resource configuration associations** tab.
5. To create an association, do the following:

    a. Choose **Create associations**.

    b. Select a resource configuration from **Resource configurations**. Choose **Create an Amazon VPC Lattice resource configuration.**.

    c. (Optional) To add a tag, expand **Service association tags**, choose **Add new tag**, and enter a tag key and tag value.

d. Choose **Save changes**.

6. To delete an association, select the check box for the association and then choose **Actions**, **Delete**. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To create a resource configuration association using the AWS CLI

Use the create-service-network-resource-association command.

To delete a resource configuration association using the AWS CLI

Use the delete-service-network-resource-association command.

Manage VPC associations

Clients can send requests to services and resources specified in resource configurations associated with a service network if the client is in VPCs associated with the service network. Client traffic that traverses a VPC peering connection or a transit gateway is only allowed through a service network using a VPC endpoint of type service network.

Associating a VPC is an optional step when you create a service network. Network owners can associate VPCs to a service network if their account has the required access. For more information, see Identity-based policy examples for VPC Lattice.

When you a delete a VPC association, clients in the VPCs can no longer connect to services in the service network.

To manage VPC associations using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.
3. Select the name of the service network to open its details page.
4. Choose the **VPC associations** tab.
5. To create a VPC association, do the following:

   a. Choose **Create VPC associations**.

   b. Choose **Add VPC association**.

   c. Select a VPC from **VPC** and select up to five security groups from **Security groups**. To create a security group, choose **Create new security group**.

d. (Optional) To add a tag, expand **VPC association tags**, choose **Add new tag**, and enter a tag key and tag value.

e. Choose **Save changes**.

6. To edit the security groups for an association, select the check box for the association and then chose **Actions**, **Edit security groups**. Add and remove security groups as needed.
7. To delete an association, select the check box for the association and then choose **Actions**, **Delete VPC associations**. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To create a VPC association using the AWS CLI

Use the create-service-network-vpc-association command.

To update the security groups for a VPC association using the AWS CLI

Use the update-service-network-vpc-association command.

To delete a VPC association using the AWS CLI

Use the delete-service-network-vpc-association command.

Manage VPC endpoint associations

Clients can send requests to services and resources specified in resource configurations over a VPC endpoint (powered by AWS PrivateLink) in their VPC. A VPC endpoint of type *service network* connects a VPC to a service network. Client traffic that comes from outside the VPC over a VPC peering connection, Transit Gateway, Direct Connect, or VPN can use the VPC endpoint to reach services and resource configurations. With VPC endpoints, you can connect a VPC to multiple service networks. When you create a VPC endpoint in a VPC, IP addresses from the VPC (and not IP addresses from the managed prefix list) are used to establish connectivity to the service network.

To manage VPC endpoint associations using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.
3. Select the name of the service network to open its details page.
4. Choose the **Endpoint associations** tab to view the VPC endpoints connected to your service network.

5. Select the Endpoint ID of the VPC endpoint to open its details page. Then modify or delete the VPC endpoint association.

To create a new VPC endpoint association using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Endpoints**.
3. Choose **Create endpoints**.
4. For **Type**, choose **Service networks**.
5. Select the service network you want to connect to your VPC.
6. Select the VPC, subnets and security groups.
7. (Optional) To add a tag, expand **VPC association tags**, choose **Add new tag**, and enter a tag key and tag value.
8. Choose **Create endpoint**.

To learn more about VPC endpoint to how to connect to service networks, see Access service networks in the *AWS PrivateLink user guide*.

Edit access settings for a VPC Lattice service network

Access settings enable you to configure and manage client access to a service network. Access settings include *auth type* and *auth policies*. Auth policies help you authenticate and authorize traffic flowing to services within VPC Lattice. Access settings of the service network do not apply to the resource configurations associated to the service network.

You can apply auth policies at the service network level, the service level, or both. Typically, auth policies are applied by the network owners or cloud administrators. They can implement coarse-grained authorization, for example, allowing authenticated calls from within the organization, or allowing anonymous GET requests that match a certain condition. At the service level, service owners can apply fine-grained controls, which can be more restrictive. For more information, see Control access to VPC Lattice services using auth policies.

To add or update access policies using the console

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

2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.
3. Select the name of the service network to open its details page.
4. Choose the **Access** tab to check the current access settings.
5. To update the access settings, choose **Edit access settings**.
6. If you want the clients in the associated VPCs to access the services in this service network, choose **None** for **Auth type**.
7. To apply a resource policy to the service network, choose **AWS IAM** for **Auth type** and do one the following for **Auth policy**:
   - Enter a policy in the input field. For example policies that you can copy and paste, choose **Policy examples**.
   - Choose **Apply policy template** and select the **Allow authenticated and unauthenticated access** template. This template allows a client from another account to access the service either by signing the request (meaning authenticated) or anonymously (meaning unauthenticated).
   - Choose **Apply policy template** and select the **Allow only authenticated access** template. This template allows a client from another account to access the service only by signing the request (meaning authenticated).
8. Choose **Save changes**.

To add or update an access policy using the AWS CLI

Use the put-auth-policy command.

Edit monitoring details for a VPC Lattice service network

VPC Lattice generates metrics and logs for every request and response, making it more efficient to monitor and troubleshoot applications.

You can enable access logs and specify the destination resource for your logs. VPC Lattice can send logs to the following resources: CloudWatch Log groups, Firehose delivery streams, and S3 buckets.

To enable access logs or update a log destination using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.

3. Select the name of the service network to open its details page.
4. Choose the **Monitoring** tab. Check **Access logs** to see whether access logs are enabled.
5. To enable or disable access logs, choose **Edit access logs**, and then turn the **Access logs** toggle switch on or off.
6. When you enable access logs, you must select the type of delivery destination, and then create or choose the destination for the access logs. You can also change the delivery destination at any time. For example:
   - Select **CloudWatch Log group** and choose a CloudWatch Log group. To create a log group, choose **Create a log group in CloudWatch**.
   - Select **S3 bucket** and enter the S3 bucket path, including any prefix. To search your S3 buckets, choose **Browse S3**.
   - Select **Kinesis Data Firehose delivery stream** and choose a delivery stream. To create a delivery stream, choose **Create a delivery stream in Kinesis**.
7. Choose **Save changes**.

To enable access logs using the AWS CLI

Use the create-access-log-subscription command.

To update the log destination using the AWS CLI

Use the update-access-log-subscription command.

To disable access logs using the AWS CLI

Use the delete-access-log-subscription command.

Manage tags for a VPC Lattice service network

Tags help you to categorize your service network in different ways, for example, by purpose, owner, or environment.

You can add multiple tags to each service network. Tag keys must be unique for each service network. If you add a tag with a key that is already associated with the service network, it updates the value of that tag. You can use characters such as letters, spaces, numbers (in UTF-8), and the following special characters: + - = . _ : / @. Do not use leading or trailing spaces. Tag values are case sensitive.

To add or delete tags using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.
3. Select the name of the service network to open its details page.
4. Choose the **Tags** tab.
5. To add a tag, choose **Add tags** and enter the tag key and tag value. To add another tag, choose **Add new tag**. When you are finished adding tags, choose **Save changes**.
6. To delete a tag, select the check box for the tag and choose **Delete**. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To add or delete tags using the AWS CLI

Use the tag-resource and untag-resource commands.

Delete a VPC Lattice service network

Before you can delete a service network, you must first delete all associations that the service network might have with any service, resource configuration, VPC, or VPC endpoint. When you delete a service network, we also delete all resources related to the service network, such as the resource policy, auth policy, and access log subscriptions.

To delete a service network using the console

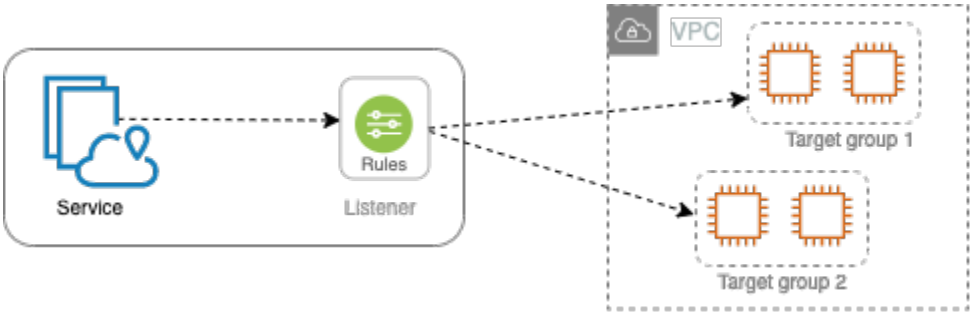
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.
3. Select the check box for the service network, and then choose **Actions**, **Delete service network**.
4. When prompted for confirmation, enter **`confirm`**, and then choose **Delete**.

To delete a service network using the AWS CLI

Use the delete-service-network command.

Services in VPC Lattice

A service within VPC Lattice is an independently deployable unit of software that delivers a specific task or function. A service can run on instances, containers, or as serverless functions within an account or a virtual private cloud (VPC). A service has a listener that uses rules, called listener rules, that you can configure to help route traffic to your targets. The supported target types include EC2 instances, IP addresses, Lambda functions, Application Load Balancers, Amazon ECS tasks, and Kubernetes Pods. For more information, see Target groups in VPC Lattice. You can associate a service with multiple service networks. The following diagram shows the key components of a typical service within VPC Lattice.



You can create a service by giving it a name and description. However, to control and monitor traffic to your service, it is important that you include access settings and monitoring details. To send traffic from your service to your targets you must set up a listener and configure rules. To allow traffic to flow from the service network to your service, you must associate your service with the service network.

There is an idle timeout and overall connection timeout for connections to targets. The idle connection timeout is 1 minute, after which we close the connection. The maximum duration is 10 minutes, after which we do not allow new streams over the connection and we begin the process of closing the existing streams.

Tasks

- Step 1: Create a VPC Lattice service
- Step 2: Define routing
- Step 3: Create network associations
- Step 4: Review and create
- Manage associations for a VPC Lattice service
- Edit access settings for a VPC Lattice service

- Edit monitoring details for a VPC Lattice service
- Manage tags for a VPC Lattice service
- Configure a custom domain name for your VPC Lattice service
- Bring Your Own Certificate (BYOC) for VPC Lattice
- Delete a VPC Lattice service

Step 1: Create a VPC Lattice service

Create a basic VPC Lattice service with access settings and monitoring details. However, the service is not fully functional until you define its routing configuration and associate it with a service network.

To create a basic service using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Choose **Create service**.
4. For **Identifiers**, do the following:

   a. Enter a name for the service. The name must be between 3-63 characters and use lowercase letters, numbers, and hyphens. It must begin and end with a letter or number. Do not use double hyphens.

   b. (Optional) Enter a description for the service network. You can set or change the description during or after creation. The description can have up to 256 characters.
5. To specify a custom domain name for your service, select **Specify a custom domain configuration** and enter the custom domain name.

   For HTTPS listeners, you can select the certificate that VPC Lattice will use to perform TLS termination. If you do not select a certificate now, you can select it when you create an HTTPS listener for the service.

   For TCP listeners, you must specify a custom domain name for your service. If you specify a certificate, it is not used. Instead, you perform TLS termination in your application.
6. For **Service access**, choose **None** if you want clients in the VPCs associated with the service network to access your service. To apply an auth policy to control access to the service, choose **AWS IAM**. To apply a resource policy to the service, do one of the following for **Auth policy**:

- Enter a policy in the input field. For example policies that you can copy and paste, choose **Policy examples**.
- Choose **Apply policy template** and select the **Allow authenticated and unauthenticated access** template. This template allows a client from another account to access the service either by signing the request (meaning authenticated) or anonymously (meaning unauthenticated).
- Choose **Apply policy template** and select the **Allow only authenticated access** template. This template allows a client from another account to access the service only by signing the request (meaning authenticated).

7. (Optional) To enable access logs, turn on the **Access logs** toggle switch and specify a destination for your access logs as follows:
   - Select **CloudWatch Log group** and choose a CloudWatch Log group. To create a log group, choose **Create a log group in CloudWatch**.
   - Select **S3 bucket** and enter the S3 bucket path, including any prefix. To search your S3 buckets, choose **Browse S3**.
   - Select **Kinesis Data Firehose delivery stream** and choose a delivery stream. To create a delivery stream, choose **Create a delivery stream in Kinesis**.
8. (Optional) To share your service with other accounts, choose an AWS RAM resource share from **Resource shares**. To create a resource share, choose **Create a resource share in RAM console**.
9. To review your configuration and create the service, choose **Skip to review and create**. Otherwise, choose **Next** to define the routing configuration for your service.

Step 2: Define routing

Define your routing configuration using listeners so your service can send traffic to the targets that you specify.

Prerequisite

Before you can add a listener, you must create a VPC Lattice target group. For more information, see the section called “Create a target group”.

To define routing for your service using the console

1. Choose **Add listener**.

2. For **Listener name**, you can either provide a custom listener name or use the protocol and port of your listener as the listener name. A custom name that you specify can have up to 63 characters, and it must be unique for every service in your account. The valid characters are a-z, 0-9, and hyphens (-). You can't use a hyphen as the first or last character, or immediately after another hyphen. You cannot change the name of a listener after you create it.
3. Choose a protocol and then enter a port number.
4. For **Default action**, choose the VPC Lattice target group to receive traffic and choose the weight to assign to this target group. You can optionally add another target group for the default action. Choose **Add action** and then choose another target group and specify its weight.
5. (Optional) To add another rule, choose **Add rule** and then enter a name, a priority, a condition, and an action for the rule.

   You can give each rule a priority number between 1 and 100. A listener can't have multiple rules with the same priority. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last.

   For **Condition**, enter a path pattern for the path match condition. The maximum size of each string is 200 characters. The comparison is not case sensitive.
6. (Optional) To add tags, expand **Listener tags**, choose **Add new tag**, and enter a tag key and a tag value.
7. To review your configuration and create the service, choose **Skip to review and create**. Otherwise, choose **Next** to associate your service to a service network.

Step 3: Create network associations

Associate your service with a service network so that clients can communicate with it.

To associate a service to a service network using the console

1. For **VPC Lattice service networks**, select the service network. To create a service network, choose **Create a VPC Lattice network**. You can associate your service with multiple service networks.
2. (Optional) To add a tag, expand **Service network association tags**, choose **Add new tag**, and enter a tag key and tag value.
3. Choose **Next**.

Step 4: Review and create

To review the configuration and create the service using the console

1. Review the configuration for your service.
2. Choose **Edit** if you need to modify any portion of the service configuration.
3. When you have finished reviewing or editing your configuration, choose **Create VPC Lattice service**.
4. If you specified a custom domain name for the service, you must configure DNS routing after the service is created. For more information, see the section called "Configure a custom domain name".

Manage associations for a VPC Lattice service

When you associate a service with the service network, it enables clients (resources in a VPC associated with the service network), to make requests to this service. You can associate services that are in your account or services that are shared with you from different accounts. This step is optional when creating the service. However, after creation, the service can't communicate with other services until you associate it with a service network. Service owners can associate their services to the service network if their account has the required access. For more information, see How VPC Lattice works.

To manage service network associations using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. Choose the **Service network associations** tab.
5. To create an association, do the following:
   a. Choose **Create associations**.
   b. Select a service network from **VPC Lattice service networks**. To create a service network, choose **Create a VPC Lattice network**.
   c. (Optional) To add a tag, expand **Service association tags**, choose **Add new tag**, and enter a tag key and tag value.

d. Choose **Save changes**.

6. To delete an association, select the check box for the association and then choose **Actions**, **Delete network associations**. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To create a service network association using the AWS CLI

Use the create-service-network-service-association command.

To delete a service network association using the AWS CLI

Use the delete-service-network-service-association command.

Edit access settings for a VPC Lattice service

Access settings enable you to configure and manage client access to a service. Access settings include *auth type* and *auth policies*. Auth policies help you authenticate and authorize traffic flowing to services within VPC Lattice.

You can apply auth policies at the service network level, the service level, or both. At the service level, service owners can apply fine-grained controls, which can be more restrictive. Typically, auth policies are applied by the network owners or cloud administrators. They can implement course-grained authorization, for example, allowing authenticated calls from within the organization, or allowing anonymous GET requests that match a certain condition. For more information, see Control access to VPC Lattice services using auth policies.

To add or update access policies using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. Choose the **Access** tab to check the current access settings.
5. To update the access settings, choose **Edit access settings**.
6. If you want the clients in VPCs in the associated service network to access your service, choose **None** for **Auth type**.
7. To apply a resource policy to control access to the service, choose **AWS IAM** for **Auth type** and do one the following for **Auth policy**:

- Enter a policy in the input field. For example policies that you can copy and paste, choose **Policy examples**.
- Choose **Apply policy template** and select the **Allow authenticated and unauthenticated access** template. This template allows a client from another account to access the service either by signing the request (meaning authenticated) or anonymously (meaning unauthenticated).
- Choose **Apply policy template** and select the **Allow only authenticated access** template. This template allows a client from another account to access the service only by signing the request (meaning authenticated).

8. Choose **Save changes**.

To add or update an access policy using the AWS CLI

Use the put-auth-policy command.

Edit monitoring details for a VPC Lattice service

VPC Lattice generates metrics and logs for every request and response, making it more efficient to monitor and troubleshoot applications.

You can enable access logs and specify the destination resource for your logs. VPC Lattice can send logs to the following resources: CloudWatch Log groups, Firehose delivery streams, and S3 buckets.

To enable access logs or update a log destination using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. Choose the **Monitoring** tab and then choose **Logs**. Check **Access logs** to see whether access logs are enabled.
5. To enable or disable access logs, choose **Edit access logs**, and then turn the **Access logs** toggle switch on or off.
6. When you enable access logs, you must select the type of delivery destination, and then create or choose the destination for the access logs. You can also change the delivery destination at any time. For example:

- Select **CloudWatch Log group** and choose a CloudWatch Log group. To create a log group, choose **Create a log group in CloudWatch**.
- Select **S3 bucket** and enter the S3 bucket path, including any prefix. To search your S3 buckets, choose **Browse S3**.
- Select **Kinesis Data Firehose delivery stream** and choose a delivery stream. To create a delivery stream, choose **Create a delivery stream in Kinesis**.

7. Choose **Save changes**.

To enable access logs using the AWS CLI

Use the create-access-log-subscription command.

To update the log destination using the AWS CLI

Use the update-access-log-subscription command.

To disable access logs using the AWS CLI

Use the delete-access-log-subscription command.

Manage tags for a VPC Lattice service

Tags help you to categorize your service in different ways, for example, by purpose, owner, or environment.

You can add multiple tags to each service. Tag keys must be unique for each service. If you add a tag with a key that is already associated with the service, it updates the value of that tag. You can use characters such as letters, spaces, numbers (in UTF-8), and the following special characters: + - = . _ : / @. Do not use leading or trailing spaces. Tag values are case sensitive.

To add or delete tags using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. Choose the **Tags** tab.
5. To add a tag, choose **Add tags** and enter the tag key and tag value. To add another tag, choose **Add new tag**. When you are finished adding tags, choose **Save changes**.

6. To delete a tag, select the check box for the tag and choose **Delete**. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To add or delete tags using the AWS CLI

Use the tag-resource and untag-resource commands.

Configure a custom domain name for your VPC Lattice service

When you create a new service, VPC Lattice generates a unique Fully Qualified Domain Name (FQDN) for the service with the following syntax.

```
service_name-service_id.partition_id.vpc-lattice-svcs.region.on.aws
```

However, the domain names that VPC Lattice provides are not easy for your users to remember. Custom domain names are simpler and more intuitive URLs that you can provide to your users. If you'd prefer to use a custom domain name for your service, such as `www.parking.example.com` instead of the VPC Lattice generated DNS name, you can configure it when you create a VPC Lattice service. When a client makes a request using your custom domain name, the DNS server resolves it to the VPC Lattice generated domain name.

Prerequisites

- You must have a registered domain name for your service. If you don't already have a registered domain name, you can register one through Amazon Route 53 or any other commercial registrar.
- To receive HTTPS requests, you must provide your own certificate in AWS Certificate Manager. VPC Lattice doesn't support a default certificate as a fallback. Therefore, if you don't provide an SSL/TLS certificate corresponding to your custom domain name, all HTTPS connections to your custom domain name will fail. For more information, see Bring Your Own Certificate (BYOC) for VPC Lattice.

Limitations and considerations

- You can’t have more than one custom domain name for a service.
- You can’t modify the custom domain name after you've created the service.

- The custom domain name must be unique for a service network. This means that a service can't be created with a custom domain name that already exists (for another service) in the same service network.

The following procedure shows how to configure a custom domain name for your service.

AWS Management Console

To configure a custom domain name for your service

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service**.
3. Choose **Create Service**. You are navigated to **Step 1: Create a service**.
4. In the **Custom domain configuration** section, choose **Specify a custom domain configuration**.
5. Enter your custom domain name.
6. To serve HTTPS requests, select the SSL/TLS certificate matching your custom domain name in **Custom SSL/TLS certificate**. If you don't have a certificate yet, or don't want to add one now, you can add a certificate when you create your HTTPS listener. However, without a certificate, your custom domain name won't be able to serve HTTPS requests. For more information, see Add an HTTPS listener.
7. When you have finished adding all other information for creating the service, choose **Create**.

AWS CLI

To configure a custom domain name for your service

Use the create-service command.

```
aws vpc-lattice create-service --name service_name --custom-domain-
name your_custom_domain_name --type https --certificate-arn arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

In the above command, for `--name`, enter a name for your service. For `--custom-domain-name`, enter your service's domain name such as, `parking.example.com`. For `--`

`certificate-arn` enter the ARN of your certificate in ACM. The certificate ARN is available in your account in AWS Certificate Manager.

Associate a custom domain name with your service

First, if you haven't already done so, register your custom domain name. The Internet Corporation for Assigned Names and Numbers (ICANN) manages domain names on the internet. You register a domain name using a *domain name registrar*, an ICANN-accredited organization that manages the registry of domain names. The website for your registrar will provide detailed instructions and pricing information for registering your domain name. For more information, see the following resources:

- To use Amazon Route 53 to register a domain name, see Registering domain names using Route 53 in the *Amazon Route 53 Developer Guide*.
- For a list of accredited registrars, see the Accredited Registrar Directory.

Next, use your DNS service, such as your domain registrar, to create a record to route queries to your service. For more information, see the documentation for your DNS service. Alternatively, you can use Route 53 as your DNS service.

If you're using Route 53, you can use an alias record or a CNAME record to route queries to your service. We recommend that you use an alias record as you can create an alias record at the top node of a DNS namespace, also known as the zone apex.

If you're using Route 53, you must first create a *hosted zone*, which contains information about how to route traffic on the internet for your domain. After you create the private or public hosted zone, create a record such that your custom domain name, for example `parking.example.com`, is mapped to the VPC Lattice auto-generated domain name, for example, `my-service-02031c045478f6ddf1.7d67968.vpc-lattice-svcs.us-west-2.on.aws`. Without this mapping, your custom domain name won't work in VPC Lattice.

The following procedures show how to create a private or public hosted zone using Route 53

AWS Management Console

To create an alias record to route queries to your service using Route 53, see Routing traffic to Amazon VPC Lattice service domain endpoint.

Use the VPC Lattice generated domain name for your service, for instance `my-service-02031c045478f6ddf1.7d67968.vpc-lattice-svcs.us-west-2.on.aws` for the **Value**. You can find this auto-generated domain name in the VPC Lattice console on your service page.

AWS CLI

To create an alias record in your hosted zone

1. Obtain the VPC Lattice generated domain name for your service (for example, `my-service-02031c045478f6ddf1.7d67968.vpc-lattice-svcs.us-west-2.on.aws`) and the hosted zone ID by running the `get-service` command.
2. To set the alias, use following command.

```
aws route53 change-resource-record-sets --hosted-zone-id hosted-zone-id-for-your-service-domain --change-batch file://~/Desktop/change-set.json
```

For the `change-set.json` file, create a JSON file with the content in the following JSON example, and save it on your local machine. Replace *file://~/Desktop/change-set.json* in the above command with the path of the JSON file saved in your local machine. Note that "Type" in the following JSON can be an A or AAAA record type.

```
{
    "Comment": "my-service-domain.com alias",
    "Changes": [
        {
            "Action": "CREATE",
            "ResourceRecordSet": {
                "Name": "my-custom-domain-name.com",
                "Type": "alias-record-type",
                "AliasTarget": {
                    "HostedZoneId": "hosted-zone-id-for-your-service-domain",
                    "DNSName": "lattice-generated-domain-name",
                    "EvaluateTargetHealth": true
                }
            }
        }
    ]
}
```

Bring Your Own Certificate (BYOC) for VPC Lattice

To serve HTTPS requests, you must have your own SSL/TLS certificate ready in AWS Certificate Manager (ACM) before you set up a custom domain name. These certificates must have a Subject Alternate Name (SAN) or Common Name (CN) that matches the custom domain name for your service. If the SAN is present, we check for a match only in the SAN list. If the SAN is absent, we check for a match in the CN.

VPC Lattice serves HTTPS requests using Server Name Indication (SNI). DNS routes the HTTPS request to your VPC Lattice service based on the custom domain name and the certificate that matches this domain name. To request an SSL/TLS certificate for a domain name in ACM or import one into ACM, see Issuing and Managing Certificates and Importing certificates in the *AWS Certificate Manager User Guide*. If you can't request or import your own certificate in ACM, use the domain name and certificate generated by VPC Lattice.

VPC Lattice accepts only one custom certificate per service. However, you can use a custom certificate for multiple custom domains. This means that you can use the same certificate for all VPC Lattice services that you create with a custom domain name.

To view your certificate using the ACM console, open **Certificates**, and select your certificate ID. You should see the VPC Lattice service that is associated with that certificate under **Associated resource**.

Limitations and considerations

- VPC Lattice allows wildcard matches that are one level deep in the Subject Alternate Name (SAN) or Common Name (CN) of the associated certificate. For example, if you create a service with the custom domain name `parking.example.com` and associate your own certificate with the SAN `*.example.com`. When a request comes in for `parking.example.com`, VPC Lattice matches the SAN to any domain name with the apex domain `example.com`. However, if you have the custom domain `parking.different.example.com` and your certificate has the SAN `*.example.com`, the request fails.
- VPC Lattice supports one level of wildcard domain match. This means that a wildcard can be used only as a first-level subdomain, and that it only secures one subdomain level. For example, if your certificate's SAN is `*.example.com`, then `parking.*.example.com` is not supported.
- VPC Lattice supports one wildcard per domain name. This means that `*.*.example.com` is not valid. For more information, see Request a public certificate in the *AWS Certificate Manager User Guide*.

- VPC Lattice only supports certificates with 2048-bit RSA keys.
- The SSL/TLS certificate in ACM must be in the same Region as the VPC Lattice service you're associating it with.

Securing your certificate's private key

When you request an SSL/TLS certificate using ACM, ACM generates a public/private key pair. When you import a certificate, you generate the key pair. The public key becomes part of the certificate. To safely store the private key, ACM creates another key using AWS KMS, called the KMS key, with the alias **aws/acm**. AWS KMS uses this key to encrypt your certificate's private key. For more information, see Data protection in AWS Certificate Manager in the *AWS Certificate Manager User Guide*.

VPC Lattice uses AWS TLS Connection Manager, a service that is only accessible only to AWS services, to secure and use your certificate's private keys. When you use your ACM certificate to create a VPC Lattice service, VPC Lattice associates your certificate with AWS TLS Connection Manager. We do this by creating a grant in AWS KMS against your AWS managed key. This grant allows TLS Connection Manager to use AWS KMS to decrypt your certificate's private key. TLS Connection Manager uses the certificate and the decrypted (plaintext) private key to establish a secure connection (SSL/TLS session) with clients of VPC Lattice services. When the certificate is disassociated from a VPC Lattice service, the grant is retired. For more information, see Grants in the *AWS Key Management Service Developer Guide*.

For more information, see Encryption at rest.

Delete a VPC Lattice service

To delete a VPC Lattice service, you must first delete all associations that the service might have with any service network. If you delete a service, all resources related to the service, such as the resource policy, auth policy, listeners, listener rules, and access log subscriptions, are also deleted.

To delete a service using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service**.
3. On the **Services** page, select the service that you want to delete, and then choose **Actions**, **Delete service**.

4. When prompted for confirmation, choose **Delete**.

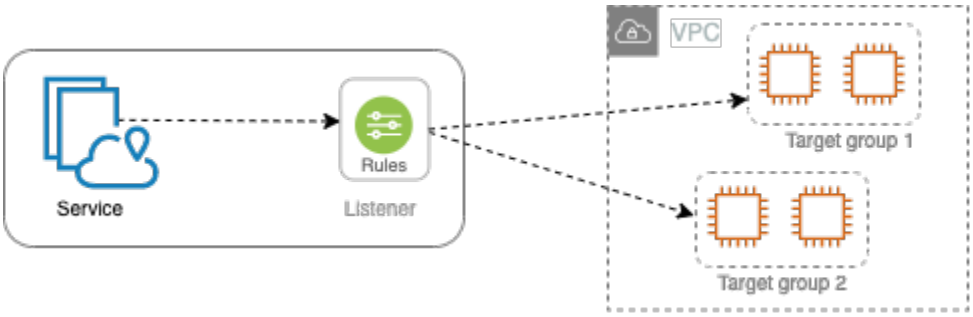
To delete a service using the AWS CLI

Use the delete-service command.

Target groups in VPC Lattice

A VPC Lattice target group is a collection of targets, or compute resources, that run your application or service. The supported target types include EC2 instances, IP addresses, Lambda functions, Application Load Balancers, Amazon ECS tasks, and Kubernetes Pods. You can also attach existing services to your target groups. For more information about using Kubernetes with VPC Lattice, see the AWS Gateway API Controller User Guide.

Each *target group* is used to route requests to one or more registered targets. When you create a listener rule, you specify a target group and conditions. When a rule condition is met, traffic is forwarded to the corresponding target group. You can create different target groups for different types of requests. For example, create one target group for general requests and other target groups for requests that include specific rule conditions, such as a path or header value.



You define health check settings for your service on a per target group basis. Each target group uses the default health check settings, unless you override them when you create the target group or modify them later on. After you specify a target group in a rule for a listener, the service continually monitors the health of all targets registered with the target group. The service routes requests to the registered targets that are healthy.

To specify a target group in a rule for a service listener, the target group must be in the same account as the service.

VPC Lattice target groups are similar to the target groups provided by Elastic Load Balancing, but they are not interchangeable.

Contents

- Routing configuration
- Routing algorithm
- Target type
- IP address type
- HTTP targets in VPC Lattice
- Lambda functions as targets in VPC Lattice
- Application Load Balancers as targets in VPC Lattice
- Protocol version
- Tags for your VPC Lattice target group
- Delete a VPC Lattice target group

Create a VPC Lattice target group

You register your targets with a target group. By default, the VPC Lattice service sends requests to registered targets using the port and protocol that you specified for the target group. You can override this port when you register each target with the target group.

To route traffic to the targets in a target group, specify the target group in an action when you create a listener or create a rule for your listener. For more information, see Listener rules for your VPC Lattice service. You can specify the same target group in multiple listeners, but these listeners must belong to the same service. To use a target group with a service, you must verify that the target group is not in use by a listener for any other service.

You can add or remove targets from your target group at any time. For more information, see Register targets with a VPC Lattice target group. You can also modify the health check settings for your target group. For more information, see Health checks for your VPC Lattice target groups.

Create a target group

You can create a target group and optionally register targets as follows.

To create a target group using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.

3. Choose **Create target group**.
4. For **Choose a target type**, do one of the following:
   - Choose **Instances** to register targets by instance ID.
   - Choose **IP addresses** to register targets by IP address.
   - Choose **Lambda function** to register a Lambda function as a target.
   - Choose **Application Load Balancer** to register an Application Load Balancer as a target.
5. For **Target group name**, enter a name for the target group. This name must be unique for your account in each AWS Region, can have a maximum of 32 characters, must contain only alphanumeric characters or hyphens, and must not begin or end with a hyphen.
6. For **Protocol** and **Port**, you can modify the default values as needed. The default protocol is **HTTPS** and the default port is **443**.

   If the target type is **Lambda function**, you can't specify a protocol or a port.
7. For **IP address type**, choose **IPv4** to register targets with IPv4 addresses or choose **IPv6** to register targets with IPv6 addresses. You can't change this setting after the target group is created.

   This option is available only if the target type is **IP addresses**.
8. For **VPC**, select a virtual private cloud (VPC).

   This option is not available if the target type is **Lambda function**.
9. For **Protocol version**, modify the default value as needed. The default is **HTTP1**.

   This option is not available if the target type is **Lambda function**.
10. For **Health checks**, modify the default settings as needed. For more information, see Health checks for your VPC Lattice target groups.

    Health checks are not available if the target type is **Lambda function**.
11. For **Lambda event structure version**, choose a version. For more information, see the section called “Receive events from the VPC Lattice service”.

    This option is available only if the target type is **Lambda function**
12. (Optional) To add tags, expand **Tags**, choose **Add new tag**, and enter the tag key and tag value.
13. Choose **Next**.

14. For **Register targets**, you can either skip this step or add targets as follows:
    - If the target type is **Instances**, select the instances, enter the ports, and then choose **Include as pending below**.
    - If the target type is **IP addresses**, do the following:
        a. For **Choose a network**, keep the VPC that you selected for the target group or choose **Other private IP address**.
        b. For **Specify IPs and define ports**, enter the IP address and enter the ports. The default port is the target group port.
        c. Choose **Include as pending below**.
    - If the target type is a **Lambda function**, choose a Lambda function. To create a Lambda function, choose **Create a new Lambda function**.
    - If the target type is a **Application Load Balancer**, choose an Application Load Balancer. To create an Application Load Balancer, choose **create an Application Load Balancer**.
15. Choose **Create target group**.

    It might take a few minutes for VPC Lattice to register the targets. For more information see, Why is it taking so long for my DNS changes to propagate in Route 53 and public resolvers?

To create a target group using the AWS CLI

Use the create-target-group command to create the target group and the register-targets command to add targets.

Shared subnets

Participants can create VPC Lattice target groups in a shared VPC. The following rules apply to shared subnets:

- All parts of a VPC Lattice service, such as listeners, target groups, and targets, must be created by the same account. They can be created in subnets owned by or shared with the owner of the VPC Lattice service.
- The targets registered with a target group must be created by the same account as the target group.
- Only the owner of a VPC can associate the VPC with a service network. Participant resources in a shared VPC that is associated with a service network can send requests to services that

are associated with the service network. However, the administrator can prevent this by using security groups, network ACLs, or auth policies.

For more information about the shareable resources for VPC Lattice, see *Share VPC Lattice entities*.

Register targets with a VPC Lattice target group

Your service serves as a single point of contact for clients and distributes incoming traffic across its healthy registered targets. You can register each target with one or more target groups.

If demand on your application increases, you can register additional targets with one or more target groups to handle the demand. The service starts routing requests to a newly registered target as soon as the registration process completes and the target passes the initial health checks.

If demand on your application decreases, or you need to service your targets, you can deregister targets from your target groups. Deregistering a target removes it from your target group, but does not affect the target otherwise. The service stops routing requests to a target as soon as it is deregistered. The target enters the `DRAINING` state until in-flight requests have completed. You can register the target with the target group again when you are ready for it to resume receiving requests.

The target type of your target group determines how you register targets with that target group. For more information, see Target type.

Use the following console procedures to register or deregister targets. Alternatively, use the register-targets and deregister-targets commands from the AWS CLI.

Contents

Register or deregister targets by instance ID

The target instances must be in the virtual private cloud (VPC) that you specified for the target group. The instance must also be in the `running` state when you register it.

When you register targets by instance ID, you can use your service with an Auto Scaling group. After you attach a target group to an Auto Scaling group and the group scales out, the instances that the Auto Scaling group launches are automatically registered with the target group. If you detach the target group from the Auto Scaling group, the instances are automatically deregistered from the target group. For more information, see Routing traffic to your Auto Scaling group with a VPC Lattice target group in the *Amazon EC2 Auto Scaling User Guide*.

To register or deregister targets by instance ID using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.
3. Choose the name of the target group to open its details page.
4. Choose the **Targets** tab.
5. To register instances, choose **Register targets**. Select the instances, enter the instance port, and then choose **Include as pending below**. When you are finished adding instances, choose **Register targets**.
6. To deregister instances, select the instances, and then choose **Deregister**.

Register or deregister targets by IP address

The target IP addresses must be from the subnets of the VPC that you specified for the target group. You can't register the IP addresses of another service in the same VPC. You can't register VPC endpoints or publicly routable IP addresses.

To register or deregister targets by IP address using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.
3. Choose the name of the target group to open its details page.
4. Choose the **Targets** tab.
5. To register IP addresses, choose **Register targets**. For each IP address, select the network, enter the IP address and port, and choose **Include as pending below**. When you are finished specifying addresses, choose **Register targets**.
6. To deregister IP addresses, select the IP addresses and then choose **Deregister**.

Register or deregister a Lambda function

You can register a single Lambda function with the target group. If you no longer need to send traffic to your Lambda function, you can deregister it. After you deregister a Lambda function, in-flight requests fail with HTTP 5XX errors. It is better to create a new target group instead of replacing the Lambda function for a target group.

To register or deregister a Lambda function using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.
3. Choose the name of the target group to open its details page.
4. Choose the **Targets** tab.
5. If there is no Lambda function registered, choose **Register target**. Select the Lambda function and choose **Register target**.
6. To deregister a Lambda function, choose **Deregister**. When prompted for confirmation, enter **`confirm`** and then choose **Deregister**.

Register or deregister an Application Load Balancer

You can register a single Application Load Balancer with each target group. If you no longer need to send traffic to your load balancer, you can deregister it. After you deregister a load balancer, in-flight requests fail with HTTP 5XX errors. It is better to create a new target group instead of replacing the Application Load Balancer for a target group.

To register or deregister an Application Load Balancer using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.
3. Choose the name of the target group to open its details page.
4. Choose the **Targets** tab.
5. If there is no Application Load Balancer registered, choose **Register target**. Select the Application Load Balancer and choose **Register target**.
6. To deregister an Application Load Balancer, choose **Deregister**. When prompted for confirmation, enter **`confirm`** and then choose **Deregister**.

Health checks for your VPC Lattice target groups

Your service periodically sends requests to its registered targets to test their status. These tests are called *health checks*.

Each VPC Lattice service routes requests only to the healthy targets. Each service checks the health of each target, using the health check settings for the target groups with which the target is registered. After your target is registered, it must pass one health check to be considered healthy. After each health check is completed, the service closes the connection that was established for the health check.

Limitations and considerations

- When the target group protocol version is HTTP1, health checks are enabled by default.
- When the target group protocol version is HTTP2, health checks are not enabled by default. However, you can enable health checks, and manually set the protocol version to HTTP1 or HTTP2.
- Health checks do not support gRPC target group protocol versions. However, if you enable health checks, you must specify the health check protocol version as HTTP1 or HTTP2.
- Health checks do not support Lambda target groups.
- Health checks do not support Application Load Balancer target groups. However, you can enable health checks for the targets of your Application Load Balancer using Elastic Load Balancing. For more information, see Target group health checks in the *User Guide for Application Load Balancers*.

Health check settings

You configure health checks for the targets in a target group as described in the following table. The setting names used in the table are the names used in the API. The service sends a health check request to each registered target every **HealthCheckIntervalSeconds** seconds, using the specified port, protocol, and ping path. Each health check request is independent and the result lasts for the entire interval. The time that it takes for the target to respond does not affect the interval for the next health check request. If the health checks exceed **UnhealthyThresholdCount** consecutive failures, the service takes the target out of service. When the health checks exceed **HealthyThresholdCount** consecutive successes, the service puts the target back in service.

Setting	Description
HealthCheckProtocol	The protocol the service uses when performin g health checks on targets. The possible protocols are HTTP and HTTPS. The default is the HTTP protocol.
HealthCheckPort	The port the service uses when performing health checks on targets. The default is to use the port on which each target receives traffic from the service.
HealthCheckPath	The destination for health checks on the targets. If the protocol version is HTTP1 or HTTP2, specify a valid URI (/*path*?*query*). The default is /.
HealthCheckTimeoutSeconds	The amount of time, in seconds, during which no response from a target means a failed health check. The range is 1–120 seconds. The default is 5 seconds if the target type is `INSTANCE` or `IP`. Specify 0 to reset this setting to its default value.
HealthCheckIntervalSeconds	The approximate amount of time, in seconds, between health checks of an individual target. The range is 5–300 seconds. The default is 30 seconds if the target type is `INSTANCE` or `IP`. Specify 0 to reset this setting to its default value.
HealthyThresholdCount	The number of consecutive successful health checks required before an unhealthy target is considered healthy. The range is 2–10. The default is 5. Specify 0 to reset this setting to its default value.

Setting	Description
UnhealthyThresholdCount	The number of consecutive health check failures required before considering a target unhealthy. The range is 2–10. The default is 2. Specify 0 to reset this setting to its default value.
Matcher	The codes to use when checking for a successful response from a target. These are called **Success codes** in the console. If the protocol version is HTTP1 or HTTP2, the possible values are from 200 to 499. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-299"). The default value is 200. Health check protocol version for gRPC is not currently supported. However, if your target group protocol version is gRPC, you can specify HTTP1 or HTTP2 protocol versions in your health check configuration.

Check the health of your targets

You can check the health status of the targets registered with your target groups.

To check the health of your targets using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.
3. Choose the name of the target group to open its details page.
4. On the **Targets** tab, the **Health status** column indicates the status of each target. If the status is any value other than `Healthy`, the **Health status details** column contains more information.

To check the health of your targets using the AWS CLI

Use the list-targets command. The output of this command contains the target health state. If the status is any value other than `Healthy`, the output also includes a reason code.

To receive email notifications about unhealthy targets

Use CloudWatch alarms to initiate a Lambda function to send details about unhealthy targets.

Modify the health check settings

You can modify the health check settings for your target group at any time.

To modify the health check settings using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.
3. Choose the name of the target group to open its details page.
4. On the **Health checks** tab, in the **Health check settings** section, choose **Edit**.
5. Modify the health check settings as needed.
6. Choose **Save changes**.

To modify the health check settings using the AWS CLI

Use the update-target-group command.

Routing configuration

By default, a service routes requests to its targets using the protocol and port number that you specified when you created the target group. Alternatively, you can override the port used for routing traffic to a target when you register it with the target group.

Target groups support the following protocols and ports:

- **Protocols**: HTTP, HTTPS, TCP
- **Ports**: 1-65535

If a target group is configured with the HTTPS protocol or uses HTTPS health checks, the TLS connections to the targets use the security policy from the listener. VPC Lattice establishes TLS

connections with the targets using certificates that you install on the targets. VPC Lattice does not validate these certificates. Therefore, you can use self-signed certificates or certificates that have expired. The traffic between VPC Lattice and the targets is authenticated at the packet level, so it is not at risk of man-in-the-middle attacks or spoofing even if the certificates on the targets are not valid.

TCP target groups are supported only with TLS listeners.

Routing algorithm

By default, the round robin routing algorithm is used to route requests to healthy targets.

When the VPC Lattice service receives a request, it uses the following process:

1. Evaluates the listener rules in priority order to determine which rule to apply.
2. Selects a target from the target group for the rule action, using the default round robin algorithm. Routing is performed independently for each target group, even when a target is registered with multiple target groups.

If a target group contains only unhealthy targets, the requests are routed to all targets, regardless of their health status. This means that if all targets fail health checks at the same time, the VPC Lattice service fails open. The effect of the fail-open is to allow traffic to all targets, regardless of their health status, based on the round robin algorithm.

Target type

When you create a target group, you specify its target type, which determines the type of target you specify when registering targets with this target group. After you create a target group, you can't change its target type.

The following are the possible target types:

`INSTANCE`

The targets are specified by instance ID.

`IP`

The targets are IP addresses.

`LAMBDA`

The target is a Lambda function.

`ALB`

The target is an Application Load Balancer.

Considerations

- When the target type is `IP`, you must specify IP addresses from the subnets of the VPC for the target group. If you need to register IP addresses from outside this VPC, create a target group of type `ALB` and register the IP addresses with the Application Load Balancer.
- When the target type is `IP`, you can't register VPC endpoints or publicly routable IP addresses.
- When the target type is `LAMBDA`, you can register a single Lambda function. When the service receives a request for the Lambda function, it invokes the Lambda function. If you would like to register multiple lambda functions to a service, you need to use multiple target groups.
- When the target type is `ALB`, you can register a single internal Application Load Balancer as the target of up to two VPC Lattice services. To do this, register the Application Load Balancer with two separate target groups, used by two different VPC Lattice services. Additionally, the targeted Application Load Balancer must have at least one listener whose port matches the target group port.
- You can automatically register your ECS tasks with a VPC Lattice target group at launch. The target group must have a target type of `IP`. For more information, see Use VPC Lattice with your Amazon ECS services in the *Amazon Elastic Container Service Developer Guide*.

  Alternatively, register the Application Load Balancer for your Amazon ECS service with a VPC Lattice target group of type `ALB`. For more information, see Use load balancing to distribute Amazon ECS service traffic in the *Amazon Elastic Container Service Developer Guide*.
- To register an EKS pod as a target, use the AWS Gateway API Controller, which gets the IP addresses from the Kubernetes service.
- If the target group protocol is TCP, the only supported target types are `INSTANCE` and `IP`.

IP address type

When you create a target group with a target type of `IP`, you can specify an IP address type for the target group. This specifies what type of addresses the load balancer uses to send requests and health checks to targets. The possible values are `IPv4` and `IPv6`. The default is `IPV4`.

Considerations

- If you create a target group with an IP address type of `IPv6`, the VPC that you specify for the target group must have an IPv6 address range.
- The IP addresses that you register with a target group must match the IP address type of the target group. For example, you can't register an IPv6 address with a target group if its IP address type is `IPv4`.
- The IP addresses that you register with a target group must be within the IP address range of the VPC that you specified for the target group.

HTTP targets in VPC Lattice

HTTP requests and HTTP responses use header fields to send information about the HTTP messages. HTTP headers are added automatically. Header fields are colon-separated name-value pairs that are separated by a carriage return (CR) and a line feed (LF). A standard set of HTTP header fields is defined in RFC 2616, Message Headers. There are also non-standard HTTP headers available that are automatically added and widely used by the applications. For example, there are non-standard HTTP headers with the `x-forwarded` prefix.

x-forwarded headers

Amazon VPC Lattice adds the following `x-forwarded` headers:

`x-forwarded-for`

The source IP address.

`x-forwarded-for-port`

The destination port.

`x-forwarded-for-proto`

The connection protocol (`http` | `https`).

Caller identity headers

Amazon VPC Lattice adds the following caller identity headers:

`x-amzn-lattice-identity`

The identity information. The following fields are present if AWS authentication is successful.

- `Principal` – The authenticated principal.
- `PrincipalOrgID` – The ID of the organization for the authenticated principal.
- `SessionName` – The name of the authenticated session.

The following fields are present if Roles Anywhere credentials are used and authentication is successful.

- `X509Issuer/OU` – The issuer (OU).
- `X509SAN/DNS` – The subject alternative name (DNS).
- `X509SAN/NameCN` – The issuer alternative name (Name/CN).
- `X509SAN/URI` – The subject alternative name (URI).
- `X509Subject/CN` – The subject name (CN).

`x-amzn-lattice-network`

The VPC. The format is as follows.

```
SourceVpcArn=arn:aws:ec2:region:account:vpc/id
```

`x-amzn-lattice-target`

The target. The format is as follows.

```
ServiceArn=arn;ServiceNetworkArn=arn;TargetGroupArn=arn
```

For information about the resource ARNs for VPC Lattice, see Resource types defined by Amazon VPC Lattice.

The caller identity headers can't be spoofed. VPC Lattice strips these headers from any incoming requests.

Lambda functions as targets in VPC Lattice

You can register your Lambda functions as targets with a VPC Lattice target group, and configure a listener rule to forward requests to the target group for your Lambda function. When the service forwards the request to a target group with a Lambda function as a target, it invokes your Lambda function and passes the content of the request to the Lambda function, in JSON format.

Limitations

- The Lambda function and target group must be in the same account and in the same Region.
- The maximum size of the request body that you can send to a Lambda function is 6 MB.
- The maximum size of the response JSON that the Lambda function can send is 6 MB.
- The protocol must be HTTP or HTTPS.

Prepare the Lambda function

The following recommendations apply if you are using your Lambda function with a VPC Lattice service.

Permissions to invoke the Lambda function

When you create the target group and register the Lambda function using the AWS Management Console or the AWS CLI, VPC Lattice adds the required permissions to your Lambda function policy on your behalf.

You can also add permissions on your own using the following API call:

```
aws lambda add-permission \
  --function-name lambda-function-arn-with-alias-name \
  --statement-id vpc-lattice \
  --principal vpc-lattice.amazonaws.com \
  --action lambda:InvokeFunction \
  --source-arn target-group-arn
```

Lambda function versioning

You can register one Lambda function per target group. To ensure that you can change your Lambda function and that the VPC Lattice service always invokes the current version of the

Lambda function, create a function alias and include the alias in the function ARN when you register the Lambda function with the VPC Lattice service. For more information, see Lambda function versions and Create an alias for a Lambda function in the *AWS Lambda Developer Guide*.

Create a target group for the Lambda function

Create a target group, which is used in request routing. If the request content matches a listener rule with an action to forward it to this target group, the VPC Lattice service invokes the registered Lambda function.

To create a target group and register the Lambda function using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.
3. Choose **Create target group**.
4. For **Choose a target type**, select **Lambda function**.
5. For **Target group name**, enter a name for the target group.
6. For **Lambda event structure version**, choose a version. For more information, see the section called "Receive events from the VPC Lattice service".
7. (Optional) To add tags, expand **Tags**, choose **Add new tag**, and enter the tag key and tag value.
8. Choose **Next**.
9. For **Lambda function**, do one of the following:
   - Select an existing Lambda function.
   - Create a new Lambda function and select it.
   - Register the Lambda function later.
10. Choose **Create target group**.

To create a target group and register the Lambda function using the AWS CLI

Use the create-target-group and register-targets commands.

Receive events from the VPC Lattice service

The VPC Lattice service supports Lambda invocation for requests over both HTTP and HTTPS. The service sends an event in JSON format, and adds the `X-Forwarded-For` header to every request.

Base64 encoding

The service Base64 encodes the body if the `content-encoding` header is present and the content type is not one of the following:

- `text/*`
- `application/json`
- `application/xml`
- `application/javascript`

If the `content-encoding` header is not present, Base64 encoding depends on the content type. For the content types above, the service sends the body as is, without Base64 encoding.

Event structure format

When you create or update a target group of type `LAMBDA`, you can specify the version of the event structure that your Lambda function receives. The possible versions are `V1` and `V2`.

Example Example event: V2

```
{
    "version": "2.0",
    "path": "/",
    "method": "GET|POST|HEAD|...",
    "headers": {
        "header-key": ["header-value", ...],
        ...
    },
    "queryStringParameters": {
        "key": ["value", ...]
    },
    "body": "request-body",
    "isBase64Encoded": true|false,
    "requestContext": {
        "serviceNetworkArn": "arn:aws:vpc-
lattice:region:123456789012:servicenetwork/sn-0bf3f2882e9cc805a",
        "serviceArn": "arn:aws:vpc-
lattice:region:123456789012:service/svc-0a40eebed65f8d69c",
        "targetGroupArn": "arn:aws:vpc-
lattice:region:123456789012:targetgroup/tg-6d0ecf831eec9f09",
        "identity": {
```

```
            "sourceVpcArn":
 "arn:aws:ec2:region:123456789012:vpc/vpc-0b8276c84697e7339",
            "type": "AWS_IAM",
            "principal": "arn:aws:iam::123456789012:assumed-role/my-role/my-session",
            "principalOrgID": "o-50dc6c495c0c9188",
            "sessionName": "i-0c7de02a688bde9f7",
            "x509IssuerOu": "string",
            "x509SanDns": "string",
            "x509SanNameCn": "string",
            "x509SanUri": "string",
            "x509SubjectCn": "string"
        },
        "region": "region",
        "timeEpoch": "1690497599177430"
    }
}
```

body

The body of the request. Present only if the protocol is HTTP, HTTPS, or gRPC.

`headers`

The HTTP headers of the request. Present only if the protocol is HTTP, HTTPS, or gRPC.

`identity`

The identity information. The following are possible fields.

- `principal` – The authenticated principal. Present only if AWS authentication is successful.
- `principalOrgID` – The ID of the organization for the authenticated principal. Present only if AWS authentication is successful.
- `sessionName` – The name of the authenticated session. Present only if AWS authentication is successful.
- `sourceVpcArn` – The ARN of the VPC where the request originated. Present only if the source VPC can be identified.
- `type` – The value is `AWS_IAM` if an auth policy is used and AWS authentication is successful.

If Roles Anywhere credentials are used and authentication is successful, the following are possible fields.

- `x509IssuerOu` – The issuer (OU).
- `x509SanDns` – The subject alternative name (DNS).

- `x509SanNameCn` – The issuer alternative name (Name/CN).
- `x509SanUri` – The subject alternative name (URI).
- `x509SubjectCn` – The subject name (CN).

`isBase64Encoded`

Indicates whether the body was base64 encoded. Present only if the protocol is HTTP, HTTPS, or gRPC and the request body is not already a string.

`method`

The HTTP method of the request. Present only if the protocol is HTTP, HTTPS, or gRPC.

`path`

The path of the request. Present only if the protocol is HTTP, HTTPS, or gRPC.

`queryStringParameters`

The HTTP query string parameters. Present only if the protocol is HTTP, HTTPS, or gRPC.

`serviceArn`

The ARN of the service that receives the request.

`serviceNetworkArn`

The ARN of the service network that delivers the request.

`targetGroupArn`

The ARN of the target group that receives the request.

`timeEpoch`

The time, in microseconds.

Example Example event: V1

```
{
    "raw_path": "/path/to/resource",
    "method": "GET|POST|HEAD|...",
    "headers": {"header-key": "header-value", ... },
    "query_string_parameters": {"key": "value", ...},
```

```
    "body": "request-body",
    "is_base64_encoded": true|false
}
```

Respond to the VPC Lattice service

The response from your Lambda function must include the Base64 encoding status, status code, and headers. You can omit the body.

To include a binary content in the body of the response, you must Base64 encode the content and set `isBase64Encoded` to `true`. The service decodes the content to retrieve the binary content and sends it to the client in the body of the HTTP response.

The VPC Lattice service does not honor hop-by-hop headers, such as `Connection` or `Transfer-Encoding`. You can omit the `Content-Length` header because the service computes it before sending responses to clients.

The following is an example response from a Lambda function:

```
{
    "isBase64Encoded": false,
    "statusCode": 200,
    "statusDescription": "200 OK",
    "headers": {
        "Set-cookie": "cookies",
        "Content-Type": "application/json"
    },
    "body": "Hello from Lambda (optional)"
}
```

Multi-value headers

VPC Lattice supports requests from a client or responses from a Lambda function that contain headers with multiple values or contain the same header multiple times. VPC Lattice passes all values to the targets.

In the following example, there are two headers named header1 with different values.

```
header1 = value1
header1 = value2
```

With a V2 event structure, VPC Lattice sends the values in a list. For example:

```
"header1": ["value1", "value2"]
```

With a V1 event structure, VPC Lattice combines the values into a single string. For example:

```
"header1": "value1, value2"
```

Multi-value query string parameters

VPC Lattice supports query parameters with multiple values for the same key.

In the following example, there are two parameters named QS1 with different values.

```
http://www.example.com?&QS1=value1&QS1=value2
```

With a V2 event structure, VPC Lattice sends the values in a list. For example:

```
"QS1": ["value1", "value2"]
```

With a V1 event structure, VPC Lattice uses the last value passed. For example:

```
"QS1": "value2"
```

Deregister the Lambda function

If you no longer need to send traffic to your Lambda function, you can deregister it. After you deregister a Lambda function, in-flight requests fail with HTTP 5XX errors.

To replace a Lambda function, we recommend that you create a new target group, register the new function with the new target group, and update the listener rules to use the new target group instead of the existing one.

To deregister a Lambda function using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.

3. Choose the name of the target group to open its details page.
4. On the **Targets** tab, choose **Deregister**.
5. When prompted for confirmation, enter **`confirm`** and then choose **Deregister**.

To deregister the Lambda function using the AWS CLI

Use the deregister-targets command.

Application Load Balancers as targets in VPC Lattice

You can create a VPC Lattice target group, register a single internal Application Load Balancer as the target, and configure your VPC Lattice service to forward traffic to this target group. In this scenario, the Application Load Balancer takes over the routing decision as soon as traffic reaches it. This configuration allows you to use the layer 7 request-based routing feature of the Application Load Balancer in combination with features that VPC Lattice supports, such as IAM authentication and authorization, and connectivity across VPCs and accounts.

Limitations

- You can register a single internal Application Load Balancer as the target in a VPC Lattice target group of type ALB.
- You can register an Application Load Balancer as a target of up to two VPC Lattice target groups, used by two different VPC Lattice services.
- VPC Lattice does not provide health checks for an ALB type target group. However, you can configure health checks independently at the load balancer level for the targets in Elastic Load Balancing. For more information, see Target group health checks in the *User Guide for Application Load Balancers*

Prerequisites

Create an Application Load Balancer to register as a target with your VPC Lattice target group. The load balancer must meet the following criteria:

- The load balancer scheme is **Internal**.
- The Application Load Balancer must be in the same account as the VPC Lattice target group, and must be in the **Active** state.

- The Application Load Balancer must be in the same VPC as the VPC Lattice target group.
- You can use HTTPS listeners on the Application Load Balancer to terminate TLS, but only if the VPC Lattice service uses the same SSL/TLS certificate as the load balancer.
- To preserve the client IP of the VPC Lattice service in the `X-Forwarded-For` request header, you must set the attribute for the Application Load Balancer `routing.http.xff_header_processing.mode` to `Preserve`. If the value is `Preserve`, the load balancer preserves the `X-Forwarded-For` header in the HTTP request, and sends it to targets without any change.

For more information, see Create an Application Load Balancer in the *User Guide for Application Load Balancers*.

Step 1: Create a target group of type ALB

Use the following procedure to create the target group. Note that VPC Lattice does not support health checks for ALB target groups. However, you can configure health checks for the target groups for your Application Load Balancer. For more information, see Target group health checks in the *User Guide for Application Load Balancers*.

To create the target group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, under **VPC Lattice**, choose **Target groups**.
3. Choose **Create target group**.
4. On the **Specify target group details** page, under **Basic configuration**, choose **Application Load Balancer** as the target type.
5. For **Target group name**, enter a name for the target group.
6. For **Protocol**, choose **HTTP** or **HTTPS**. The target group protocol must match the protocol of the listener for your internal Application Load Balancer.
7. For **Port**, specify the port for your target group. This port must match the port of the listener for your internal Application Load Balancer. You can alternatively add a listener port on the internal Application Load Balancer to match the target group port that you specify here.
8. For **VPC**, select the same virtual private cloud (VPC) that you selected when you created the internal Application Load Balancer. This should be the VPC that contains your VPC Lattice resources.

9. For **Protocol version**, choose the protocol version that your Application Load Balancer supports.
10. (Optional) Add any required tags.
11. Choose **Next**.

Step 2: Register the Application Load Balancer as a target

You can either register the load balancer as a target now or later on.

To register an Application Load Balancer as a target

1. Choose **Register now**.
2. For **Application Load Balancer**, choose your internal Application Load Balancer.
3. For **Port**, keep the default or specify a different port as needed. This port must match an existing listener port on your Application Load Balancer. If you continue without a matching port, traffic won't reach your Application Load Balancer.
4. Choose **Create target group**.

Protocol version

By default, services send requests to targets using HTTP/1.1. You can use the protocol version to send requests to targets using HTTP/2 or gRPC.

The following table summarizes the result for the combinations of request protocol and target group protocol version.

Request protocol	Protocol version	Result
HTTP/1.1	HTTP/1.1	Success
HTTP/2	HTTP/1.1	Success
gRPC	HTTP/1.1	Error
HTTP/1.1	HTTP/2	Error
HTTP/2	HTTP/2	Success

Request protocol	Protocol version	Result
gRPC	HTTP/2	Success if targets support gRPC
HTTP/1.1	gRPC	Error
HTTP/2	gRPC	Success if a POST request
gRPC	gRPC	Success

Considerations for the gRPC protocol version

- The only supported listener protocol is HTTPS.
- The only supported target types are `INSTANCE` and `IP`.
- The service parses gRPC requests and routes the gRPC calls to the appropriate target groups based on the package, service, and method.
- You can't use Lambda functions as targets.

Considerations for the HTTP/2 protocol version

- The only supported listener protocol is HTTPS. You can choose either HTTP or HTTPS for the target group protocol.
- The only supported listener rules are forward and fixed response.
- The only supported target types are `INSTANCE` and `IP`.
- The service supports streaming from clients. The service does not support streaming to the targets.

Tags for your VPC Lattice target group

Tags help you to categorize your target groups in different ways, for example, by purpose, owner, or environment.

You can add multiple tags to each target group. Tag keys must be unique for each target group. If you add a tag with a key that is already associated with the target group, it updates the value of that tag.

When you are finished with a tag, you can remove it.

Restrictions

- Maximum number of tags per resource—50
- Maximum key length—127 Unicode characters
- Maximum value length—255 Unicode characters
- Tag keys and values are case sensitive. Allowed characters are letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @. Do not use leading or trailing spaces.
- Do not use the `aws:` prefix in your tag names or values because it is reserved for AWS use. You can't edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.

To update the tags for a target group using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Target groups**.
3. Select the name of the target group to open its details page.
4. Choose the **Tags** tab.
5. To add a tag, choose **Add tags** and enter the tag key and tag value. To add another tag, choose **Add new tag**. When you are finished adding tags, choose **Save changes**.
6. To delete a tag, select the check box for the tag and choose **Delete**. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To update the tags for a target group using the AWS CLI

Use the tag-resource and untag-resource commands.

Delete a VPC Lattice target group

You can delete a target group if it is not referenced by the forward actions of any listener rules. Deleting a target group does not affect the targets registered with the target group. If you no longer need a registered EC2 instance, you can stop or terminate it.

To delete a target group using the console

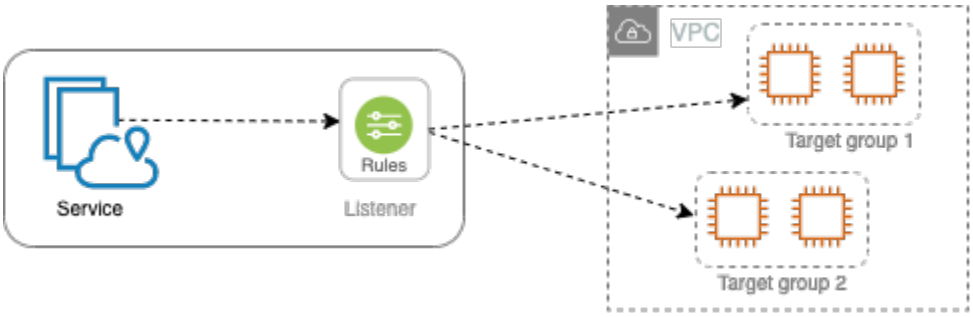
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the navigation pane, choose **Target groups**.
3. Select the check box for the target group and then choose **Actions**, **Delete**.
4. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To delete a target group using the AWS CLI

Use the delete-target-group command.

Listeners for your VPC Lattice service

Before you start using your VPC Lattice service, you must add a *listener*. A listener is a process that checks for connection requests, using the protocol and port that you configure. The rules that you define for a listener determine how the service routes requests to its registered targets.



Contents

Listener configuration

Listeners support the following protocols and ports:

- **Protocols**: HTTP, HTTPS, TLS
- **Ports**: 1-65535

If the listener protocol is HTTPS, VPC Lattice will provision and manage a TLS certificate that is associated with the VPC Lattice generated FQDN. VPC Lattice supports TLS on HTTP/1.1 and HTTP/2. When you configure a service with an HTTPS listener, VPC Lattice will automatically determine the HTTP protocol using Application-Layer Protocol Negotiation (ALPN). If ALPN is absent, VPC Lattice defaults to HTTP/1.1. For more information, see HTTPS listeners.

VPC Lattice can listen on HTTP, HTTPS, HTTP/1.1, and HTTP/2 and communicate to targets in any of these protocols and versions. We do not require that the listener and target group protocols match. VPC Lattice manages the entire process of upgrading and downgrading between protocols and versions. For more information, see Protocol version.

You can create a TLS listener to ensure that your application decrypts the encrypted traffic instead of VPC Lattice. For more information, see TLS listeners.

VPC Lattice does not support WebSockets.

HTTP listeners for VPC Lattice services

A listener is a process that checks for connection requests. You can define a listener when you create your VPC Lattice service. You can add listeners to your service at any time.

The information on this page helps you create an HTTP listener for your service. For information about creating listeners that use other protocols, see HTTPS listeners and TLS listeners.

Prerequisites

- To add a forward action to the default listener rule, you must specify an available VPC Lattice target group. For more information, see Create a VPC Lattice target group.
- You can specify the same target group in multiple listeners, but these listeners must belong to the same service. To use a target group with a VPC Lattice service, you must verify that it is not used by a listener for any other VPC Lattice service.

Add an HTTP listener

You can add listeners and rules to your service at any time. You configure a listener with a protocol and a port for connections from clients to the service, and a VPC Lattice target group for the default listener rule. For more information, see Listener configuration.

To add an HTTP listener using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. On the **Routing** tab, choose **Add listener**.

5. For **Listener name**, you can either provide a custom listener name, or use the protocol and port of your listener as the listener name. A custom name that you specify can have up to 63 characters, and it must be unique for every service in your account. The valid characters are a-z, 0-9, and hyphens (-). You can't use a hyphen as the first or last character, or immediately after another hyphen. You cannot change the name after you create it.
6. For **Protocol : port**, choose **HTTP** and enter a port number.
7. For **Default action**, choose the VPC Lattice target group to receive traffic and choose the weight to assign to this target group. The weight that you assign to a target group sets its priority to receive traffic. For example, if two target groups have the same weight, each target group receives half of the traffic. If you've specified only one target group, then 100 percent of the traffic is sent to the one target group.

   You can optionally add another target group for the default action. Choose **Add action** and then choose a target group and specify its weight.
8. (Optional) To add another rule, choose **Add rule** and then enter a name, a priority, a condition, and an action for the rule.

   You can give each rule a priority number between 1 and 100. A listener can't have multiple rules with the same priority. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last. For more information, see Listener rules.
9. (Optional) To add tags, expand **Listener tags**, choose **Add new tag**, and enter a tag key and tag value.
10. Review your configuration, and then choose **Add**.

To add an HTTP listener using the AWS CLI

Use the create-listener command to create a listener with a default rule, and the create-rule command to create additional listener rules.

HTTPS listeners for VPC Lattice services

A listener is a process that checks for connection requests. You define a listener when you create your service. You can add listeners to your service in VPC Lattice at any time.

You can create an HTTPS listener, which uses TLS version 1.2 or TLS version 1.3 to terminate HTTPS connections with VPC Lattice directly. VPC Lattice will provision and manage a TLS certificate that is associated with the VPC Lattice generated Fully Qualified Domain Name (FQDN).

VPC Lattice supports TLS on HTTP/1.1 and HTTP/2. When you configure a service with an HTTPS listener, VPC Lattice will automatically determine the HTTP protocol via Application-Layer Protocol Negotiation (ALPN). If ALPN is absent, VPC Lattice defaults to HTTP/1.1.

VPC Lattice uses a multi-tenancy architecture, meaning that it can host multiple services on the same endpoint. VPC Lattice uses TLS with Server Name Indication (SNI) for every client request. Encrypted Client Hello (ECH) and Encrypted Server Name Indication (ESNI) aren't supported.

VPC Lattice can listen on HTTP, HTTPS, HTTP/1.1, and HTTP/2 and communicate to targets in any of these protocols and versions. These listener and target group configurations do not need to match. VPC Lattice manages the entire process of upgrading and downgrading between protocols and versions. For more information, see Protocol version.

To ensure that your application decrypts the traffic, create a TLS listener instead. With TLS passthrough, VPC Lattice does not terminate TLS. For more information, see TLS listeners.

Contents

Security policy

VPC Lattice uses a security policy that is a combination a TLSv1.2 protocol and a list of SSL/TLS ciphers. The protocol establishes a secure connection between a client and a server and helps to ensure that all data passed between the client and your service in VPC Lattice is private. A cipher is an encryption algorithm that uses encryption keys to create a coded message. Protocols use several ciphers to encrypt data. During the connection negotiation process, the client and VPC Lattice present a list of ciphers and protocols that they each support, in order of preference. By default, the first cipher on the server's list that matches any one of the client's ciphers is selected for the secure connection.

VPC Lattice uses the following TLS 1.2 SSL/TLS ciphers in this order of preference:

- `ECDHE-RSA-AES128-GCM-SHA256`
- `ECDHE-RSA-AES128-SHA`
- `ECDHE-RSA-AES256-GCM-SHA384`
- `ECDHE-RSA-AES256-SHA`

- AES128-GCM-SHA256
- AES128-SHA
- AES256-GCM-SHA384
- AES256-SHA

VPC Lattice also uses the following TLS 1.3 SSL/TLS ciphers in this order of preference:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

ALPN policy

Application-Layer Protocol Negotiation (ALPN) is a TLS extension that is sent on the initial TLS handshake hello messages. ALPN enables the application layer to negotiate which protocols should be used over a secure connection, such as HTTP/1 and HTTP/2.

When the client initiates an ALPN connection, the VPC Lattice service compares the client ALPN preference list with its ALPN policy. If the client supports a protocol from the ALPN policy, the VPC Lattice service establishes the connection based on the preference list of the ALPN policy. Otherwise, the service does not use ALPN.

VPC Lattice supports the following ALPN policy:

HTTP2Preferred

Prefer HTTP/2 over HTTP/1.1. The ALPN preference list is h2, http/1.1.

Add an HTTPS listener

You configure a listener with a protocol and a port for connections from clients to the service, and a target group for the default listener rule. For more information, see Listener configuration.

Prerequisites

- To add a forward action to the default listener rule, you must specify an available VPC Lattice target group. For more information, see Create a VPC Lattice target group.

- You can specify the same target group in multiple listeners, but these listeners must belong to the same VPC Lattice service. To use a target group with a VPC Lattice service, you must verify that it is not used by a listener for any other VPC Lattice service.
- You can use the certificate provided by VPC Lattice or import your own certificate to AWS Certificate Manager. For more information, see the section called “BYOC”.

To add an HTTPS listener using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. On the **Routing** tab, choose **Add listener**.
5. For **Listener name**, you can either provide a custom listener name or use the protocol and port of your listener as the listener name. A custom name that you specify can have up to 63 characters, and it must be unique for every service in your account. The valid characters are a-z, 0-9, and hyphens (-). You can't use a hyphen as the first or last character, or immediately after another hyphen. You cannot change the name of a listener after you create it.
6. For **Protocol : port**, choose **HTTPS** and enter a port number.
7. For **Default action**, choose the VPC Lattice target group to receive traffic and choose the weight to assign to this target group. The weight that you assign to a target group sets its priority to receive traffic. For example, if two target groups have the same weight, each target group receives half of the traffic. If you've specified only one target group, then 100 percent of the traffic is sent to the one target group.

   You can optionally add another target group for the default action. Choose **Add action** and then choose a target group and specify its weight.
8. (Optional) To add another rule, choose **Add rule** and then enter a name, a priority, a condition, and an action for the rule.

   You can give each rule a priority number between 1 and 100. A listener can't have multiple rules with the same priority. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last. For more information, see Listener rules.
9. (Optional) To add tags, expand **Listener tags**, choose Add new tag, and enter a tag key and tag value.

10. For **HTTPS listener certificate settings**, if you did not specify a custom domain name when you created the service, VPC Lattice automatically generates a TLS certificate to secure the traffic flowing though the listener.

    If you created the service with a custom domain name, but didn't specify a matching certificate, you can do so now by choosing the certificate from **Custom SSL/TLS certificate**. Otherwise, the certificate that you specified when you created the service is already chosen.

11. Review your configuration, and then choose **Add**.

To add an HTTPS listener using the AWS CLI

Use the create-listener command to create a listener with a default rule, and the create-rule command to create additional listener rules.

TLS listeners for VPC Lattice services

A listener is a process that checks for connection requests. You can define a listener when you create your VPC Lattice service. You can add listeners to your service at any time.

You can create a TLS listener so that VPC Lattice passes encrypted traffic through to your applications without decrypting it.

If you prefer that VPC Lattice decrypts encrypted traffic and sends unencrypted traffic to your applications, create an HTTPS listener instead. For more information, see HTTPS listeners.

Considerations

The following considerations apply to TLS listeners:

- The VPC Lattice service must have a custom domain name. The service custom domain name is used as a Service Name Indication (SNI) match. If you specified a certificate when you created the service, it is not used.
- The only rule allowed for a TLS listener is the default rule.
- The default action for a TLS listener must be a forward action to a TCP target group.
- By default, health checks are disabled for TCP target groups. If you enable health checks for a TCP target group, you must specify a protocol and protocol version.

- TLS listeners route requests using the SNI field of the client-hello message. You can use wildcard and SAN certificates on your targets if the matching condition is an exact match to the client-hello.
- Because all traffic remains encrypted from the client to the target, VPC Lattice can't read the HTTP headers and can't insert or remove HTTP headers. Therefore, with a TLS listener, the following limitations exist:
  - Connection duration is limited to 10 minutes
  - Auth policies are limited to anonymous principals
  - Lambda targets are not supported
- Encrypted Client Hello (ECH) isn't supported.
- Encrypted Server Name Indication (ESNI) isn't supported.

Add a TLS listener

You configure a listener with a protocol and a port for connections from clients to the service, and a target group for the default listener rule. For more information, see Listener configuration.

To add a TLS listener using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. On the **Routing** tab, choose **Add listener**.
5. For **Listener name**, you can either provide a custom listener name or use the protocol and port of your listener as the listener name. A custom name that you specify can have up to 63 characters, and it must be unique for every service in your account. The valid characters are a-z, 0-9, and hyphens (-). You can't use a hyphen as the first or last character, or immediately after another hyphen. You cannot change the name of a listener after you create it.
6. For **Protocol**, choose **TLS**. For **Port**, enter a port number.
7. For **Forward to target group**, choose a VPC Lattice target group that uses the TCP protocol to receive the traffic, and choose the weight to assign to this target group. You can optionally add another target group. Choose **Add target group** and then choose a target group and enter its weight.

8. (Optional) To add tags, expand **Listener tags**, choose Add new tag, and enter a tag key and tag value.
9. Review your configuration, and then choose **Add**.

To add a TLS listener using the AWS CLI

Use the create-listener command to create a listener with a default rule. Specify the TLS_PASSTHROUGH protocol.

Listener rules for your VPC Lattice service

Each listener has a default rule and additional rules that you can define. Each rule consists of a priority, one or more actions, and one or more conditions. You can add or edit rules at any time.

Contents

Default rules

When you create a listener, you define actions for the default rule. Default rules can't have conditions. If the conditions for none of a listener's rules are met, then the action for the default rule is performed.

Rule priority

Each rule has a priority. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last. You can change the priority of a non-default rule at any time. You cannot change the priority of the default rule.

Rule action

Listeners for VPC Lattice services support forward actions and fixed response actions.

Forward actions

You can use `forward` actions to route requests to one or more VPC Lattice target groups. If you specify multiple target groups for a `forward` action, you must specify a weight for each target group. Each target group weight is a value from 0 to 999. Requests that match a listener rule with weighted target groups are distributed to these target groups based on their weights. For example, if you specify two target groups, each with a weight of 10, each target group receives half the requests. If you specify two target groups, one with a weight of 10 and the other with a weight of 20, the target group with a weight of 20 receives twice as many requests as the other target group.

Fixed-response actions

You can use `fixed-response` actions to drop client requests and return a custom HTTP response. You can use this action to return a 404 or a 500 response code.

Example Example fixed response action for the AWS CLI

You can specify an action when you create or update a rule. The following action sends a fixed response with the specified status code.

```
"action": {
    "fixedResponse": {
        "statusCode": 404
},
```

Rule conditions

Each rule condition has a type and configuration information. When the conditions for a rule are met, then its actions are performed.

The following are the supported matching criteria for a rule:

Header match

Routing is based on the HTTP headers for each request. You can use HTTP header conditions to configure rules that route requests based on the HTTP headers for the request. You can

specify the names of standard or custom HTTP header fields. The header name and the match evaluation are not case sensitive. You can change this setting by turning on case-sensitivity. Wildcard characters are not supported in the header name. Prefix, exact, and contains matching are supported on header match.

Method match

Routing is based on the HTTP request method of each request.

You can use HTTP request method conditions to configure rules that route requests based on the HTTP request method of the request. You can specify standard or custom HTTP methods. The method match is case sensitive. The method name must be an exact match. Wildcard characters are not supported.

Path match

Routing is based on matching the path patterns in the request URLs.

You can use path conditions to define rules that route requests based on the URL in the request. Wildcard characters are not supported. Prefix and exact matching on path are supported.

Add a rule

You can add a listener rule at any time.

To add a listener rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. On the **Routing** tab, choose **Edit listener**.
5. Expand **Listener rules** and choose **Add rule**.
6. For **Rule name**, enter a name for the rule.
7. For **Priority**, enter a priority between 1 and 100. Rules are evaluated in priority order, from the lowest value to the highest value. The default rule is evaluated last.
8. For **Condition**, enter a path pattern for the path match condition. The maximum size of each string is 200 characters. The comparison is not case sensitive. Wildcard characters are not supported.

To add a header match or method match rule condition, use the AWS CLI or an AWS SDK.

9. For **Action**, choose a VPC Lattice target group.
10. Choose **Save changes**.

To add a rule using the AWS CLI

Use the create-rule command.

Update a rule

You can update a listener rule at any time. You can modify its priority, condition, target group, and the weight of each target group. You can't modify the name of the rule.

To update a listener rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. On the **Routing** tab, choose **Edit listener**.
5. Modify the rule priorities, conditions, and actions as needed.
6. Review your updates and choose **Save changes**.

To update a rule using the AWS CLI

Use the update-rule command.

Delete a rule

You can delete the non-default rules for a listener at any time. You cannot delete the default rule for a listener. When you delete a listener, all of its rules are deleted.

To delete a listener rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.

4. On the **Routing** tab, choose **Edit listener**.
5. Find the rule and choose **Remove**.
6. Choose **Save changes**.

To delete a rule using the AWS CLI

Use the delete-rule command.

Delete a listener for your VPC Lattice service

You can delete a listener at any time. When you delete a listener, all its rules are automatically deleted.

To delete a listener using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**.
3. Select the name of the service to open its details page.
4. On the **Routing** tab, choose **Delete listener**.
5. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To delete a listener using the AWS CLI

Use the delete-listener command.

VPC resources in Amazon VPC Lattice

You can share VPC resources with other teams in your organization or with external independent software vendor (ISV) partners. A VPC resource can be an AWS-native resource such as an Amazon RDS database, a domain name, or an IP address. The resource can be in your VPC or on-premises network and does not need to be load-balanced. You use AWS RAM to specify the principals who can access the resource. You create a resource gateway through which your resource can be accessed. You also create a resource configuration that represents the resource or a group of resources that you want to share.

The principals that you share the resource with can access these resources privately using VPC endpoints. They can use a resource VPC endpoint to access one resource or pool multiple resources in an VPC Lattice service network, and access the service network using a service-network VPC endpoint.

The following sections explain how to create and manage VPC resources in VPC Lattice:

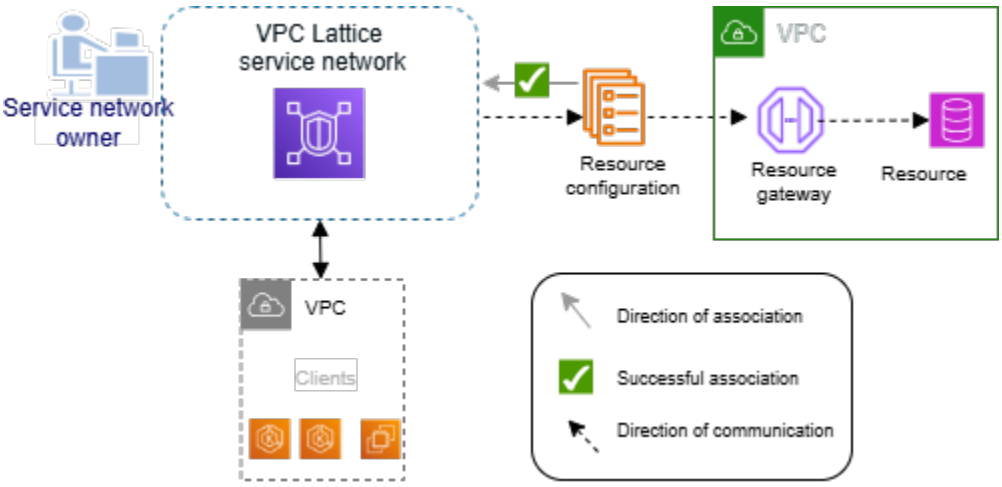
Topics

Resource gateways in VPC Lattice

A *resource gateway* is the point that receives traffic into the VPC where a resource resides. It spans multiple Availability Zones.

A VPC must have a resource gateway if you plan on making resources inside the VPC accessible from other VPCs or accounts. Every resource you share is associated with a resource gateway. When clients in other VPCs or accounts access a resource in your VPC, the resource sees traffic coming locally from the resource gateway in that VPC. The source IP address of the traffic is the IP address of the resource gateway in an Availability Zone. Multiple resource configurations, each having multiple resources, can be attached to a resource gateway.

The following diagram shows how a client accesses a resource through the resource gateway:



Contents

Considerations

The following considerations apply to resource gateways:

- For your resource to be accessible from all Availability Zones, you should create your resource gateways to span as many Availability Zones as possible.
- At least one Availability Zone of the VPC endpoint and the resource gateway have to overlap.
- A VPC can have a maximum of 100 resource gateways. For more information, see Quotas for VPC Lattice.
- You can't create a resource gateway in a shared subnet.

Security groups

You can attach security groups to a resource gateway. Security group rules for resource gateways control outbound traffic from the resource gateway to resources.

Recommended outbound rules for traffic flowing from a resource gateway to a database resource

For traffic to flow from a resource gateway to a resource, you must create outbound rules for the resource's accepted listener protocols and port ranges.

Destination	Protocol	Port range	Comment
CIDR range for resource	TCP	3306	Allows traffic from resource gateway to databases.

IP address types

A resource gateway can have IPv4, IPv6 or dual-stack addresses. The IP address type of a resource gateway must be compatible with the subnets of the resource gateway and the IP address type of the resource, as described here:

- **IPv4** – Assign IPv4 addresses to your resource gateway network interfaces. This option is supported only if all selected subnets have IPv4 address ranges, and the resource also has an IPv4 address.
- **IPv6** – Assign IPv6 addresses to your resource gateway network interfaces. This option is supported only if all selected subnets are IPv6 only subnets, and the resource also has an IPv6 address.
- **Dualstack** – Assign both IPv4 and IPv6 addresses to your resource gateway network interfaces. This option is supported only if all selected subnets have both IPv4 and IPv6 address ranges, and the resource either has an IPv4 or IPv6 address.

The IP address type of the resource gateway is independent of the IP address type of the client or the VPC endpoint through which the resource is accessed.

Create a resource gateway in VPC Lattice

Use the console to create a resource gateway.

Prerequisite

To create a resource gateway, you must have a /28 block available in a subnet.

To create a resource gateway using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **PrivateLink and Lattice**, choose **Resource gateways**.
3. Choose **Create resource gateway**.
4. Enter a name that is unique within your AWS account.
5. Choose the type of IP for the resource gateway.
6. Choose the VPC that the resource is in.
7. Choose up to five security groups to control inbound traffic from the VPC to the service network.
8. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
9. Choose **Create resource gateway**.

To create a resource gateway using the AWS CLI

Use the create-resource-gateway command.

Delete a resource gateway in VPC Lattice

Use the console to delete a resource gateway.

To delete a resource gateway using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **PrivateLink and Lattice**, choose **Resource gateways**.
3. Select the check box for the resource gateway that you want to delete and choose **Actions**, **Delete**. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To delete a resource gateway using the AWS CLI

Use the delete-resource-gateway command.

Resource configurations for VPC resources

A resource configuration represents a resource or a group of resources that you want to make accessible to clients in other VPCs and accounts. By defining a resource configuration, you can

allow private, secure, unidirectional network connectivity to resources in your VPC from clients in other VPCs and accounts. A resource configuration is associated with a resource gateway through which it receives traffic. For a resource to be accessed from another VPC, it needs to have a resource configuration.

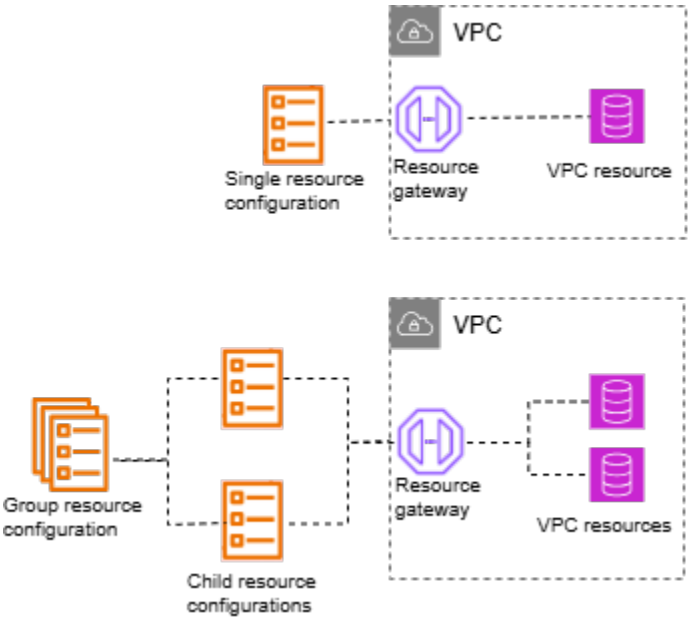
Contents

Types of resource configurations

A resource configuration can be of several types. The different types help represent different kinds of resources. The types are:

- **Single resource configuration**: Represents an IP address or a domain name. It can be shared independently.
- **Group resource configuration**: It is collection of child resource configurations. It can be used to represent a group of DNS and IP address endpoints.
- **Child resource configuration**: It is a member of a group resource configuration. It represents an IP address or a domain name. It can't be shared independently; it can only be shared as part of a group. It can be added and removed from a group. When added, its automatically accessible to those who can access the group.
- **ARN resource configuration**: Represents a supported resource-type that is provisioned by an AWS service. Any group-child relationship is automatically taken care of.

The following image shows a single, child, and group resource configuration:



Resource gateway

A resource configuration is associated with a resource gateway. A resource gateway is a set of ENIs that serve as a point of ingress into the VPC in which the resource is in. Multiple resource configurations can be associated with the same resource gateway. When clients in other VPCs or accounts access a resource in your VPC, the resource sees traffic coming locally from the resource gateway's IP addresses in that VPC.

Resource definition

In the resource configuration, identify the resource in one of the following ways:

- By an **Amazon Resource Name (ARN)**: Supported resource-types that are provisioned by AWS services, can be identified by their ARN. Only Amazon RDS databases are supported. You can't create a resource configuration for a publicly accessible cluster.
- By a **domain-name target**: You can use any domain name that is publicly resolvable. If your domain name points to an IP that's outside of your VPC, you must have a NAT gateway in your VPC.
- By an **IP-address**: For IPv4, specify a private IP from the following ranges: 10.0.0.0/8, 100.64.0.0/10, 172.16.0.0/12, 192.168.0.0/16. For IPv6, specify an IP from the VPC. Public IPs aren't supported.

Protocol

When you create a resource configuration you can define the protocols that the resource will support. Currently, only the TCP protocol is supported.

Port ranges

When you create a resource configuration you can define the ports it will accept requests on. Client access on other ports will not be allowed.

Accessing resources

Consumers can access resource configurations directly from their VPC using a VPC endpoint or through a service network. As a consumer, you can enable access from your VPC to a resource configuration that is in your account or that has been shared with you from another account through AWS RAM.

- *Accessing a resource configuration directly*

  You can create a AWS PrivateLink VPC endpoint of type resource (resource endpoint) in your VPC to access a resource configuration privately from your VPC. For more information on how to create a resource endpoint, see Accessing VPC resources in the *AWS PrivateLinkuser guide*.

- *Accessing a resource configuration through a service network*

  You can associate a resource configuration to a service network, and connect your VPC to the service network. You can connect your VPC to the service network either through an association or using a AWS PrivateLink service-network VPC endpoint.

  For more information on service network associations, see Manage the associations for a VPC Lattice service network.

  For more information on service network VPC endpoints, see Access service networks in the *AWS PrivateLink user guide*.

When private DNS is enabled for your VPC, you can't create a resource endpoint and service network endpoint for the same resource configuration.

Association with service network type

When you share a resource configuration with a consumer account, for example, Account-B, through AWS RAM, Account-B can access the resource configuration either directly through a resource VPC endpoint, or through a service network.

To access a resource configuration through a service network, Account-B would have to associate the resource configuration with a service network. Service networks are shareable between accounts. So, Account-B can share their service network (that the resource configuration is associated to) with Account-C, making your resource accessible from Account-C.

In order to prevent such transitive sharing, you can specify that your resource configuration cannot be added to service networks that are shareable between accounts. If you specify this, then Account-B won't be able to add your resource configuration to service networks that are shared or can be shared with another account in the future.

Types of service networks

When you share a resource configuration with another account, for example Account-B, through AWS RAM, Account-B can access the resources specified in the resource configuration in one of three ways:

- Using a VPC endpoint of type *resource* (resource VPC endpoint).
- Using a VPC endpoint of type *service network* (service network VPC endpoint).
- Using a service network VPC association.

  When you use a service-network association, each resource is assigned an IP per subnet from the 129.224.0.0/17 block, which is AWS owned and non-routable. This is in addition to the managed prefix list that VPC Lattice uses to route traffic to services over the VPC Lattice network. Both of these IPs are updated to your VPC route table.

For service network VPC endpoint and service network VPC association, the resource configuration would have to be associated with a service network in Account-B. Service networks are shareable between accounts. So, Account-B can share their service network (that contains the resource configuration) with Account-C, making your resource accessible from Account-C. In order to prevent such transitive sharing, you can disallow your resource configuration from being added to service networks that are shareable between accounts. If you disallow this, then Account-B won't be

able to add your resource configuration to a service network that is shared or can be shared with another account.

Sharing resource configurations through AWS RAM

Resource configurations are integrated with AWS Resource Access Manager. You can share your resource configuration with another account through AWS RAM. When you share a resource configuration with an AWS account, clients in that account can privately access the resource. You can share a resource configuration using a resource share in AWS RAM.

Use the AWS RAM console, to view the resource shares to which you have been added, the shared resources that you can access, and the AWS accounts that have shared resources with you. For more information, see Resources shared with you in the *AWS RAM User Guide*.

To access a resource from another VPC in the same account as the resource configuration, you don't need to share the resource configuration through AWS RAM.

Monitoring

You can enable monitoring logs on your resource configuration. You can choose a destination to send the logs to.

Create a resource configuration in VPC Lattice

Use the console to create a resource configuration.

To create a resource configuration using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **PrivateLink and Lattice**, choose **Resource configurations**.
3. Choose **Create resource configuration**.
4. Enter a name that is unique within your AWS account. You can't change this name after the resource configuration is created.
5. For **Configuration type**, choose **Resource** for a single or child resource or **Resource group** for a group of child resources.
6. Choose a resource gateway that you previously created or create a one now.
7. Choose the identifier for the resource that you want this resource configuration to represent.
8. Choose the port ranges through which you want to share the resource.

9. For **Association settings**, specify whether this resource configuration can be associated with shareable service networks.
10. For **Share resource configuration**, choose the resource shares that identify the principals who can access this resource.
11. (Optional) For **Monitoring**, enable **Resource access logs** and the delivery destination if you want to monitor requests and responses to and from the resource configuration.
12. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
13. Choose **Create resource configuration**.

To create a resource configuration using the AWS CLI

Use the create-resource-configuration command.

Manage associations for a VPC Lattice resource configuration

Consumer accounts with which you share a resource configuration with and clients in your account can access the resource configuration either directly using a VPC endpoint of type resource or through a VPC endpoint of type service-network. As a result your resource configuration will have endpoint associations and service network associations.

Manage service network associations

Create or delete a service network association.

> **Note**
>
> If you receive an access-denied message while creating the association between the service network and resource configuration, check your AWS RAM policy version and ensure that it is version 2. For more information, see the AWS RAM user guide.

To manage a service-network association using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **PrivateLink and Lattice**, choose **Resource configurations**.
3. Select the name of the resource configuration to open its details page.
4. Select **Service network associations** tab.

5. Choose **Create associations**.
6. Select a service network from **VPC Lattice service networks**. To create a service network, choose **Create a VPC Lattice network**.
7. (Optional) To add a tag, expand **Service association tags**, choose **Add new tag**, and enter a tag key and tag value.
8. Choose **Save changes**.
9. To delete an association, select the check box for the association and then choose **Actions**, **Delete**. When prompted for confirmation, enter **`confirm`** and then choose **Delete**.

To create a service network association using the AWS CLI

Use the create-service-network-resource-association command.

To delete a service network association using the AWS CLI

Use the delete-service-network-resource-association command.

Manage VPC endpoint associations

Manage a VPC endpoint association.

To manage a VPC endpoint association using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **PrivateLink and Lattice**, choose **Resource configurations**.
3. Select the name of the resource configuration to open its details page.
4. Choose the **Endpoint associations** tab.
5. Select the association ID to open its details page. From here, you can modify or delete the association.
6. To create a new endpoint association, go to **PrivateLink and Lattice** in the left navigation pane and choose **Endpoints**.
7. Choose **Create endpoints**.
8. Select the resource configuration you want to connect to your VPC.
9. Select the VPC, subnets and security groups.
10. (Optional) To tag you VPC endpoint, choose **Add new tag**, and enter a tag key and tag value.
11. Choose **Create endpoint**.

To create a VPC endpoint association using the AWS CLI

Use the create-vpc-endpoint command.

To delete a VPC endpoint association using the AWS CLI

Use the delete-vpc-endpoint command.

Share your VPC Lattice entities

Amazon VPC Lattice integrates with AWS Resource Access Manager (AWS RAM) to enable sharing of services, resource configurations, and service networks. AWS RAM is a service that enables you to share some VPC Lattice entities with other AWS accounts or through AWS Organizations. With AWS RAM, you share entities that you own by creating a *resource share*. A resource share specifies the entities to share, and the consumers with whom to share them. Consumers can include:

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

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

Contents

Prerequisites for sharing VPC Lattice entities

- To share an entity, you must own it in your AWS account. This means that the entity must be allocated or provisioned in your account. You can't share a entity that has been shared with you.
- To share an entity with your organization or an organizational unit in AWS Organizations, you must enable sharing with AWS Organizations. For more information, see Enable resource sharing within AWS Organizations in the *AWS RAM User Guide*.

Share VPC Lattice entities

To share an entity, start by creating a resource share using AWS Resource Access Manager. A resource share specifies the entities to share, the consumers with whom they are shared, and what actions principals can perform.

When you share a VPC Lattice entity that you own with other AWS accounts, you enable those accounts to associate their entities with entities in your account. When you create an association against a shared entity, we generate an Amazon Resource Name (ARN) in the entity owner account and in the account that created the association. Therefore, both the entity owner and the account that created the association can delete the association.

If you are part of an organization in AWS Organizations and sharing within your organization is enabled, consumers in your organization are automatically granted access to the shared entity. Otherwise, consumers receive an invitation to join the resource share and are granted access to the shared entity after accepting the invitation.

Considerations

- You can share three types of VPC Lattice entities: service networks, services, and resource configurations.
- You can share your VPC Lattice entities with any AWS account.
- You can't share your VPC Lattice entities with individual IAM users and roles.
- VPC Lattice supports customer-managed permissions for services, resource configurations, and service networks.

To share an entity that you own using the VPC Lattice console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**, **Service networks**, or **Resource configurations**.
3. Choose the name of the entity to open its details page, and then choose **Share service**, **Share service network**, or **Share resource configuration** from the **Sharing** tab.
4. Choose the AWS RAM resource shares from **Resource shares**. To create a resource share, choose **Create a resource share in RAM console**.
5. Choose **Share service**, **Share service network**, or **Share resource configuration**.

To share an entity that you own using the AWS RAM console

Use the procedure described in Create a resource share in the *AWS RAM User Guide*.

To share an entity that you own using the AWS CLI

Use the associate-resource-share command.

Stop sharing VPC Lattice entities

To stop sharing a VPC Lattice entity that you own, you must remove it from the resource share. Existing associations persist after you stop sharing your entity. New associations to a previously shared entity are not allowed. When either the entity owner or the association owner deletes an association, it is deleted from both accounts. If an account owner wants to leave a resource share, they must ask the owner of the resource share to remove their account from the list of accounts this resource was shared with.

To stop sharing an entity that you own using the VPC Lattice console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Services**, **Service networks**, or Resource configurations.
3. Choose the name of the entity to open its details page.
4. On the **Sharing** tab, select the check box for the resource share and then choose **Remove**.

To stop sharing an entity that you own using the AWS RAM console

See Update a resource share in the *AWS RAM User Guide*.

To stop sharing an entity that you own using the AWS CLI

Use the disassociate-resource-share command.

Responsibilities and permissions

The following responsibilities and permissions apply when using shared VPC Lattice entities.

Entity owners

- The service network owner can't modify a service created by a consumer.
- The service network owner can't delete a service created by a consumer.
- The service network owner can describe all service associations for the service network.
- The service network owner can disassociate any service associated with the service network, regardless of who created the association.

- The service network owner can describe all VPC associations for the service network.
- The service network owner can disassociate any VPC that a consumer associated with the service network.
- The service network owner can describe all resource configuration associations for the service network.
- The service network owner can disassociate any resource configuration associated with the service network, regardless of who created the association.
- The service network owner can describe all endpoint associations for the service network.
- The service network owner can disassociate any endpoints associated with the service network, regardless of who created the association.
- The service owner can describe all service network associations with the service.
- The service owner can disassociate a service from any service network that it is associated with.
- The resource configuration owner can describe all network associations with the resource configuration.
- The resource configuration owner can disassociate a resource configuration from any service network that it is associated with.
- The VPC endpoint owner can describe the service network that it is associated with.
- The VPC endpoint owner can dissociate an endpoint from the service network.
- Only the account that created an association can update the association between the service network and the VPC.

Entity consumers

- The consumer can't delete a service or resource configuration that they didn’t create.
- The consumer can disassociate only the services or resource configurations that they associated with a service network.
- The consumer and network owner can describe all associations between a service network and a service or resource configuration.
- The consumer can't retrieve service information of a service or resource configuration information of a resource configuration that they don't own.
- The consumer can describe all service associations and resource configurations associations with a shared service network.
- The consumer can associate a service or a resource configuration with a shared service network.

- The consumer can see all VPC associations with a shared service network.
- The consumer can associate a VPC with a shared service network.
- The consumer can disassociate only the VPCs that they associated with a service network.
- The consumer can create a service network VPC endpoint to connect their VPC to a shared service network.
- The consumer can delete only the service network VPC endpoint they created to connect their VPC to a shared service network.
- The consumer of a shared service can't associate a service with a service network that they don't own.
- The consumer of a shared service network can't associate a VPC or service that they don't own.
- The consumer of a shared resource configuration can't associate a resource configuration with a service network that they don't own.
- The consumer of a shared service network can't associate a VPC or service or resource configuration that they don't own.
- The consumer can describe a service, service network, or resource configuration that is shared with them.
- The consumer can't associate two entities if both are shared with them.

Cross-account events

When entity owners and consumers perform actions on a shared entity, those actions are recorded as cross-account events in AWS CloudTrail.

`CreateServiceNetworkResourceAssociationBySharee`

Sent to the entity owner when a entity consumer calls CreateServiceNetworkResourceAssociation with a shared entity. If the caller owns the resource configuration, the event is sent to the owner of the service network. If the caller owns the service network, the event is sent to the owner of the resource configuration.

`CreateServiceNetworkServiceAssociationBySharee`

Sent to the entity owner when a entity consumer calls CreateServiceNetworkServiceAssociation with a shared entity. If the caller owns the service, the event is sent to the owner of the service network. If the caller owns the service network, the event is sent to the owner of the service.

`CreateServiceNetworkVpcAssociationBySharee`

Sent to the entity owner when a entity consumer calls CreateServiceNetworkVpcAssociation with a shared service network.

`DeleteServiceNetworkResourceAssociationByOwner`

Sent to the association owner when the entity owner calls DeleteServiceNetworkResourceAssociation with a shared entity. If the caller owns the resource configuration, the event is sent to the owner of the service network association. If the caller owns the service network, the event is sent to the owner of the resource association.

`DeleteServiceNetworkResourceAssociationBySharee`

Sent to the entity owner when a entity consumer calls DeleteServiceNetworkResourceAssociation with a shared entity. If the caller owns the resource configuration, the event is sent to the owner of the service network. If the caller owns the service network, the event is sent to the owner of the resource configuration.

`DeleteServiceNetworkServiceAssociationByOwner`

Sent to the association owner when the entity owner calls DeleteServiceNetworkServiceAssociation with a shared entity. If the caller owns the service, the event is sent to the owner of the service network association. If the caller owns the service network, the event is sent to the owner of the service association.

`DeleteServiceNetworkServiceAssociationBySharee`

Sent to the entity owner when a entity consumer calls DeleteServiceNetworkServiceAssociation with a shared entity. If the caller owns the service, the event is sent to the owner of the service network. If the caller owns the service network, the event is sent to the owner of the service.

`DeleteServiceNetworkVpcAssociationByOwner`

Sent to the association owner when the entity owner calls DeleteServiceNetworkVpcAssociation with a shared service network.

`DeleteServiceNetworkVpcAssociationBySharee`

Sent to the entity owner when a entity consumer calls DeleteServiceNetworkVpcAssociation with a shared service network.

`GetServiceBySharee`

Sent to the entity owner when a entity consumer calls GetService with a shared service.

`GetServiceNetworkBySharee`

Sent to the entity owner when a entity consumer calls GetServiceNetwork with a shared service network.

`GetServiceNetworkResourceAssociationBySharee`

Sent to the entity owner when a entity consumer calls GetServiceNetworkResourceAssociation with a shared entity. If the caller owns the resource configuration, the event is sent to the owner of the service network. If the caller owns the service network, the event is sent to the owner of the resource configuration.

`GetServiceNetworkServiceAssociationBySharee`

Sent to the entity owner when a entity consumer calls GetServiceNetworkServiceAssociation with a shared entity. If the caller owns the service, the event is sent to the owner of the service network. If the caller owns the service network, the event is sent to the owner of the service.

`GetServiceNetworkVpcAssociationBySharee`

Sent to the entity owner when a entity consumer calls GetServiceNetworkVpcAssociation with a shared service network.

The following is an example entry for the `CreateServiceNetworkServiceAssociationBySharee` event.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-04-27T17:12:46Z",
    "eventSource": "vpc-lattice.amazonaws.com",
    "eventName": "CreateServiceNetworkServiceAssociationBySharee",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "vpc-lattice.amazonaws.com",
    "userAgent": "ec2.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
        "callerAccountId": "111122223333"
    },
    "requestID": "ddabb0a7-70c6-4f70-a6c9-00cbe8a6a18b",
```

```
    "eventID": "bd03cdca-7edd-4d50-b9c9-eaa89f4a47cd",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::VpcLattice::ServiceNetworkServiceAssociation",
            "ARN": "arn:aws:vpc-
lattice:region:123456789012:servicenetworkserviceassociation/snsa-0d5ea7bc72EXAMPLE"
        }
    ],
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

VPC Lattice for Oracle Database@AWS

VPC Lattice powers AWS managed service integrations for Oracle Database@AWS (ODB) and provides you with simplified connectivity between the ODB network, AWS VPCs and on premise. To support this connectivity, VPC Lattice provisions the following entities on your behalf:

Default service network

The default service network uses the naming convention `default-odb-network-randomHash`

Default service-network endpoint

There is no name for this AWS resource.

Resource gateway

The resource gateway uses the naming convention `default-odb-network-randomHash`

VPC Lattice supports AWS managed service integrations, referred to as *managed integrations* to your ODB network. By default, Oracle Cloud Infrastructure (OCI) Managed Backup to Amazon S3 is enabled. You can choose to enable self-managed access to Amazon S3 and Zero-ETL.

Once you create your ODB network, you can view the provisioned resources using the AWS Management Console or AWS CLI. The following example command lists the ODB network's default managed integrations and any other resources you might have for this service network:

```
aws vpc-lattice list-service-network-resource-associations \
        --service-network-identifier default-odb-network-randomHash
```

Considerations

The following considerations apply to VPC Lattice for Oracle Database@AWS:

- You can't delete the default service network, service-network endpoint, resource gateway, or any ODB managed integrations provisioned by VPC Lattice. To delete these entities, delete your ODB network or disable the managed integrations.
- Clients can only access the managed integrations in the ODB network. Clients outside the ODB network, such as in your VPCs, cannot use these managed integrations to access S3 or Zero-ETL.

- You can't connect to any of the managed integrations outside of the ODB network provisioned by VPC Lattice.
- All traffic to Amazon S3 goes through the default service-network endpoint and standard processing charges for accessing resources apply. All Zero-ETL traffic goes over the resource gateway and standard data processing charges for resources that you share apply. For more information, see VPC Lattice pricing.
- There are no hourly charges for Oracle Database@AWS managed integrations.
- You can manage the resources provisioned by VPC Lattice just like any other service network. You can share the default service network with other AWS accounts or organizations, and add new endpoints, VPC associations, VPC Lattice services and resources to the default network.
- The following permissions are required for VPC Lattice to provision Oracle Database@AWS resources:

  JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "AllowODBEC2andLatticeActions",
         "Action": [
             "ec2:DescribeVpcs",
             "ec2:CreateTags",
             "ec2:DescribeAvailabilityZones",
             "ec2:CreateOdbNetworkPeering",
             "ec2:DeleteOdbNetworkPeering",
             "ec2:ModifyOdbNetworkPeering",
             "ec2:DescribeVpcEndpointAssociations",
             "ec2:CreateVpcEndpoint",
             "ec2:DeleteVpcEndpoints",
             "ec2:DescribeVpcEndpoints",
             "vpc-lattice:CreateServiceNetwork",
             "vpc-lattice:DeleteServiceNetwork",
             "vpc-lattice:GetServiceNetwork",
             "vpc-lattice:CreateServiceNetworkResourceAssociation",
             "vpc-lattice:DeleteServiceNetworkResourceAssociation",
             "vpc-lattice:GetServiceNetworkResourceAssociation",
             "vpc-lattice:CreateResourceGateway",
             "vpc-lattice:DeleteResourceGateway",
             "vpc-lattice:GetResourceGateway",
             "vpc-lattice:CreateServiceNetworkVpcEndpointAssociation"
```

```
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
    "Sid": "AllowSLRActionsForLattice",
    "Effect": "Allow",
    "Action": [
     "iam:CreateServiceLinkedRole"
    ],
    "Resource": "*",
     "Condition": {
      "StringEquals": {
       "iam:AWSServiceName": [
        "vpc-lattice.amazonaws.com"
       ]
      }
     }
   }
    ]
}
```

To use VPC Lattice for Oracle Database@AWS, we recommend that you are familiar with service networks, service-network associations, and resource gateways in VPC Lattice.

Topics

- the section called “Oracle Cloud Infrastructure (OCI) Managed Backup to Amazon S3”
- the section called “Amazon S3 access”
- the section called “Zero-ETL for Amazon Redshift”
- the section called “Access and share VPC Lattice entities”

Oracle Cloud Infrastructure (OCI) Managed Backup to Amazon S3

When you create an Oracle Database@AWS database, VPC Lattice creates a resource configuration called `odb-managed-s3-backup-access`. This resource configuration represents an OCI managed backup of your databases to Amazon S3 and only enables connectivity to Amazon

S3 buckets owned by OCI. Traffic between the ODB Network and S3 never leaves the Amazon network.

Amazon S3 access

In addition to the OCI Managed Backup to Amazon S3, you can create a managed integration that enables access to Amazon S3 from the ODB network. When you modify the Oracle Database@AWS network to enable the Amazon S3 Access managed integration, VPC Lattice provisions a resource configuration called `odb-s3-access` in the default service network. You can use this integration to access Amazon S3 for your own needs including self-managed backups or restores. You can establish perimeter control by providing an auth policy.

Considerations

The following are considerations for the Amazon S3 Access managed integration:

- You can create only one Amazon S3 Access managed integration for the ODB network.
- This managed integration enables access to Amazon S3 from the ODB network only, and not from other VPC associations or service-network endpoints in the default service network.
- You can't access S3 buckets in different AWS Regions.

Enable the Amazon S3 Access managed integration

Use the following command to enable the Amazon S3 Access managed integration:

```
aws odb update-odb-network \
  --odb-network-id odb-network-id \
  --s3-access ENABLED
```

Secure access with an auth policy

You can secure access to S3 buckets by defining an auth policy using the ODB API. The following example policy grants access to specific S3 buckets owned by a specific organization.

JSON

```
{
```

```
  "Version": "2012-10-17",
  "Id": "Policy1515115909152",
  "Statement": [
    {
      "Sid": "GrantAccessToMyOrgS3",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3:::awsexamplebucket1",
        "arn:aws:s3:::awsexamplebucket1/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceOrgID": "o-abcd1234"
        }
      }
    }
  ]
}
```

Note

The `aws:SourceVpc`, `aws:SourceVpce`, and `aws:VpcSourceIp` condition keys aren't supported for S3 bucket policies when using ODB managed integrations.

Zero-ETL for Amazon Redshift

You can use the service network provisioned by VPC Lattice to enable Zero-ETL. This managed integration connects your ODB network databases to Amazon Redshift to help analyze data across different databases. You can initiate the Zero-ETL setup using AWS Glue integration APIs and use the ODB APIs to turn on the managed integration and setup the network path. For more information, see Zero-ETL integration with Amazon Redshift.

Considerations

The following are considerations for the managed Zero-ETL integration:

- If you enable the managed Zero-ETL integration, you can only use Zero-ETL to access instances in your ODB network. Other services and resources associated with your service network are isolated from Zero-ETL.

Access and share VPC Lattice entities

You can also connect your ODB network to services, resources, and other clients in VPCs using VPC Lattice. These connectivity options are powered through the default service network, resource gateway, and service-network endpoint provisioned by VPC Lattice.

Access VPC Lattice services and resources

To access other entities, associate services or resources that you own, or are shared with you, to the default service network. Clients in the ODB network can access the services or resources through the default service-network endpoint.

Considerations

The following are considerations for connecting to other VPC Lattice entities:

- You can add new service-network endpoints, VPC associations, VPC Lattice resources and services to the service network, but you can't modify the resources provisioned by VPC Lattice on behalf of the ODB network. These must be managed through the Oracle Database@AWS APIs.

Share your ODB network through VPC Lattice

You can share your ODB network resources with clients in other VPCs, accounts or on premises. To get started, create a resource configuration for the resources that you want to share. The resource configurations must use the default resource gateway for your ODB network. You can then associate the resources with your default service network.

Clients in other VPCs or AWS accounts that you've shared your service network with can access these resources through their own service network endpoints or VPC associations. For more information, see the section called “Manage associations”.

Considerations

The following are considerations for sharing your ODB network:

- We recommend only sharing ODB network instances as IP-based resources.
- VPC Lattice doesn't support OCI's Single Client Access Name (SCAN) listener DNS.

Security in Amazon VPC Lattice

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

You are responsible for maintaining control over your content that is hosted on this infrastructure. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

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

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

Contents

Manage access to VPC Lattice services

VPC Lattice is secure by default because you must be explicit about which services and resource configurations to provide access to and with which VPCs. You can access services through a VPC association or a VPC endpoint of type service network. For multi-account scenarios, you can use AWS Resource Access Manager to share services, resource configurations, and service networks across account boundaries.

VPC Lattice provides a framework that lets you implement a defense-in-depth strategy at multiple layers of the network.

- **First layer** – The service, resource, VPC, and VPC endpoint association with a service network. A VPC may be connected to a service network either though an association or through a VPC endpoint. If a VPC is not connected to a service network, clients in the VPC cannot access the service and resource configurations that are associated with the service network.
- **Second layer** – Optional network-level security protections for the service network, such as security groups and network ACLs. By using these, you can allow access to specific groups of clients in a VPC instead of all clients in the VPC.
- **Third layer** – Optional VPC Lattice auth policy. You can apply an auth policy to service networks and individual services. Typically, the auth policy on the service network is operated by the network or cloud administrator, and they implement coarse-grained authorization. For example, allowing only authenticated requests from a specific organization in AWS Organizations. For an auth policy at the service level, typically the service owner sets fine-grained controls, which might be more restrictive than the coarse-grained authorization applied at the service network level.

> **Note**
>
> The auth policy on the service network doesn't apply to resource configurations in the service network.

Methods of access control

- Auth policies
- Security groups
- Network ACLs

Control access to VPC Lattice services using auth policies

VPC Lattice auth policies are IAM policy documents that you attach to service networks or services to control whether a specified principal has access to a group of services or specific service. You can attach one auth policy to each service network or service that you want to control access to.

> **Note**
>
> The auth policy on the service network doesn't apply to resource configurations in the service network.

Auth policies are different from IAM identity-based policies. IAM identity-based policies are attached to IAM users, groups, or roles and define what actions those identities can do on which resources. Auth policies are attached to services and service networks. For authorization to succeed, both auth policies and identity-based policies need to have explicit allow statements. For more information, see How authorization works.

You can use the AWS CLI and console to view, add, update, or remove auth policies on services and service networks. When you add, update, or remove an auth policy, it might take a few minutes to be ready. When using the AWS CLI, make sure you are in the correct Region. You can either change the default Region for your profile, or use the `--region` parameter with the command.

Contents

To get started with auth policies, follow the procedure to create an auth policy that applies to a service network. For more restrictive permissions that you don't want applied to other services, you can optionally set auth policies on individual services.

Manage access to a service network with auth policies

The following AWS CLI tasks show you how to manage access to a service network using auth policies. For instructions that use the console, see Service networks in VPC Lattice.

Tasks

- Add an auth policy to a service network
- Change a service network's auth type
- Remove an auth policy from a service network

Add an auth policy to a service network

Follow the steps in this section to use the AWS CLI to:

- Enable access control on a service network using IAM.
- Add an auth policy to the service network. If you do not add an auth policy, all traffic will get an access denied error.

To enable access control and add an auth policy to a new service network

1. To enable access control on a service network so that it can use an auth policy, use the **create-service-network** command with the `--auth-type` option and a value of `AWS_IAM`.

   ```
   aws vpc-lattice create-service-network --name Name --auth-type AWS_IAM [--
   tags TagSpecification]
   ```

   If successful, the command returns output similar to the following.

   ```
   {
      "arn": "arn",
      "authType": "AWS_IAM",
      "id": "sn-0123456789abcdef0",
      "name": "Name"
   }
   ```

2. Use the **put-auth-policy** command, specifying the ID of the service network where you want to add the auth policy and the auth policy you want to add.

For example, use the following command to create an auth policy for the service network with the ID *sn-0123456789abcdef0*.

```
aws vpc-lattice put-auth-policy --resource-identifier sn-0123456789abcdef0 --
policy file://policy.json
```

Use JSON to create a policy definition. For more information, see Common elements in an auth policy.

If successful, the command returns output similar to the following.

```
{
   "policy": "policy",
   "state": "Active"
}
```

To enable access control and add an auth policy to an existing service network

1. To enable access control on a service network so that it can use an auth policy, use the **update-service-network** command with the `--auth-type` option and a value of `AWS_IAM`.

```
aws vpc-lattice update-service-network --service-network-
identifier sn-0123456789abcdef0 --auth-type AWS_IAM
```

If successful, the command returns output similar to the following.

```
{
   "arn": "arn",
   "authType": "AWS_IAM",
   "id": "sn-0123456789abcdef0",
   "name": "Name"
}
```

2. Use the **put-auth-policy** command, specifying the ID of the service network where you want to add the auth policy and the auth policy you want to add.

```
aws vpc-lattice put-auth-policy --resource-identifier sn-0123456789abcdef0 --
policy file://policy.json
```

Use JSON to create a policy definition. For more information, see Common elements in an auth policy.

If successful, the command returns output similar to the following.

```
{
   "policy": "policy",
   "state": "Active"
}
```

Change a service network's auth type

To disable the auth policy for a service network

Use the **update-service-network** command with the `--auth-type` option and a value of `NONE`.

```
aws vpc-lattice update-service-network --service-network-
identifier sn-0123456789abcdef0 --auth-type NONE
```

If you need to enable the auth policy again later, run this command with `AWS_IAM` specified for the `--auth-type` option.

Remove an auth policy from a service network

To remove an auth policy from a service network

Use the **delete-auth-policy** command.

```
aws vpc-lattice delete-auth-policy --resource-identifier sn-0123456789abcdef0
```

The request fails if you remove an auth policy before changing the auth type of a service network to `NONE`.

Manage access to a service with auth policies

The following AWS CLI tasks show you how to manage access to a service using auth policies. For instructions that use the console, see Services in VPC Lattice.

Tasks

- Add an auth policy to a service
- Change a service's auth type
- Remove an auth policy from a service

Add an auth policy to a service

Follow these steps to use the AWS CLI to:

- Enable access control on a service using IAM.
- Add an auth policy to the service. If you do not add an auth policy, all traffic will get an access denied error.

To enable access control and add an auth policy to a new service

1. To enable access control on a service so that it can use an auth policy, use the **create-service** command with the `--auth-type` option and a value of `AWS_IAM`.

   ```
   aws vpc-lattice create-service --name Name --auth-type AWS_IAM [--
   tags TagSpecification]
   ```

   If successful, the command returns output similar to the following.

   ```
   {
      "arn": "arn",
      "authType": "AWS_IAM",
      "dnsEntry": {
         ...
      },
      "id": "svc-0123456789abcdef0",
      "name": "Name",
      "status": "CREATE_IN_PROGRESS"
   }
   ```

2. Use the **put-auth-policy** command, specifying the ID of the service where you want to add the auth policy and the auth policy you want to add.

   For example, use the following command to create an auth policy for the service with the ID *svc-0123456789abcdef0*.

```
aws vpc-lattice put-auth-policy --resource-identifier svc-0123456789abcdef0 --
policy file://policy.json
```

Use JSON to create a policy definition. For more information, see Common elements in an auth policy.

If successful, the command returns output similar to the following.

```
{
   "policy": "policy",
   "state": "Active"
}
```

To enable access control and add an auth policy to an existing service

1. To enable access control on a service so that it can use an auth policy, use the **update-service** command with the `--auth-type` option and a value of `AWS_IAM`.

```
aws vpc-lattice update-service --service-identifier svc-0123456789abcdef0 --auth-
type AWS_IAM
```

If successful, the command returns output similar to the following.

```
{
   "arn": "arn",
   "authType": "AWS_IAM",
   "id": "svc-0123456789abcdef0",
   "name": "Name"
}
```

2. Use the **put-auth-policy** command, specifying the ID of the service where you want to add the auth policy and the auth policy you want to add.

```
aws vpc-lattice put-auth-policy --resource-identifier svc-0123456789abcdef0 --
policy file://policy.json
```

Use JSON to create a policy definition. For more information, see Common elements in an auth policy.

If successful, the command returns output similar to the following.

```
{
   "policy": "policy",
   "state": "Active"
}
```

Change a service's auth type

To disable the auth policy for a service

Use the **update-service** command with the `--auth-type` option and a value of `NONE`.

```
aws vpc-lattice update-service --service-identifier svc-0123456789abcdef0 --auth-type
 NONE
```

If you need to enable the auth policy again later, run this command with `AWS_IAM` specified for the `--auth-type` option.

Remove an auth policy from a service

To remove an auth policy from a service

Use the **delete-auth-policy** command.

```
aws vpc-lattice delete-auth-policy --resource-identifier svc-0123456789abcdef0
```

The request fails if you remove an auth policy before changing the auth type of the service to `NONE`.

If you enable auth policies that require authenticated requests to a service, any requests to that service must contain a valid request signature that is computed using Signature Version 4 (SigV4). For more information, see SIGv4 authenticated requests for Amazon VPC Lattice.

Common elements in an auth policy

VPC Lattice auth policies are specified using the same syntax as IAM policies. For more information, see Identity-based policies and resource-based policies in the *IAM User Guide*.

An auth policy contains the following elements:

- **Principal** – The person or application who is allowed access to the actions and resources in the statement. In an auth policy, the principal is the IAM entity who is the recipient of this permission. The principal is authenticated as an IAM entity to make requests to a specific resource, or group of resources as in the case of services in a service network.

  You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services. For more information, see AWS JSON policy elements: Principal in the *IAM User Guide*.
- **Effect** – The effect when the specified principal requests the specific action. This can be either `Allow` or `Deny`. By default, when you enable access control on a service or service network using IAM, principals have no permissions to make requests to the service or service network.
- **Actions** – The specific API action for which you are granting or denying permission. VPC Lattice supports actions that use the `vpc-lattice-svcs` prefix. For more information, see Actions defined by Amazon VPC Lattice Services in the *Service Authorization Reference*.
- **Resources** – The services that are affected by the action.
- **Condition** – Conditions are optional. You can use them to control when your policy is in effect. For more information, see Condition keys for Amazon VPC Lattice Services in the *Service Authorization Reference*.

As you create and manage auth policies, you might want to use the IAM Policy Generator.

Requirement

The policy in JSON must not contain newlines or blank lines.

Resource format for auth policies

You can restrict access to specific resources by creating an auth policy that uses a matching schema with a `<serviceARN>/<path>` pattern and code the `Resource` element as shown in the following examples.

Protocol	Examples
HTTP	• `"Resource": "arn:aws:vpc-lattice:us-west-2:1234567890:service/svc-0123456789abcdef0/rates"`

Protocol	Examples
	• "Resource": "*/rates" • "Resource": "*/*"
gRPC	• "Resource": "arn:aws:vpc-lattice:us-west-2:1234567890:service/svc-0123456789abcdef0/api.parking/GetRates" • "Resource": "arn:aws:vpc-lattice:us-west-2:1234567890:service/svc-0123456789abcdef0/api.parking/*" • "Resource": "arn:aws:vpc-lattice:us-west-2:1234567890:service/svc-0123456789abcdef0/*"

Use the following Amazon Resource Name (ARN) resource format for <serviceARN>:

```
arn:aws:vpc-lattice:region:account-id:service/service-id
```

For example:

```
"Resource": "arn:aws:vpc-lattice:us-west-2:123456789012:service/svc-0123456789abcdef0"
```

Condition keys that can be used in auth policies

Access can be further controlled by condition keys in the **Condition** element of auth policies. These condition keys are present for evaluation depending on the protocol and whether the request is signed with Signature Version 4 (SigV4) or anonymous. Condition keys are case sensitive.

AWS provides global condition keys that you can use to control access, such as aws:PrincipalOrgID and aws:SourceIp. To see a list of the AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

The following tale lists the VPC Lattice condition keys. For more information, see Condition keys for Amazon VPC Lattice Services in the *Service Authorization Reference*.

Condition keys	Description	Example	Available for anonymous (unauthenticated) caller?	Available for gRPC?
`vpc-lattice-svcs:Port`	Filters access by the service port the request is made to	80	Yes	Yes
`vpc-lattice-svcs:RequestMethod`	Filters access by the method of the request	GET	Yes	Always POST
`vpc-lattice-svcs:RequestHeader/` *`header-name`* `:` *`value`*	Filters access by a header name-value pair in the request headers	content-type: application/json	Yes	Yes
`vpc-lattice-svcs:RequestQueryString/` *`key-name`*`:` *`value`*	Filters access by the query string key-value pairs in the request URL	quux: [corge, grault]	Yes	No
`vpc-lattice-svcs:ServiceNetworkArn`	Filters access by the ARN of the service network of the service that is receiving the request	arn:aws:vpc-lattice:us-west-2:123456789012:servicenetwork/sn-0123456789abcdef0	Yes	Yes
`vpc-lattice-svcs:ServiceArn`	Filters access by the ARN of the service that is receiving the request	arn:aws:vpc-lattice:us-west-2:123456	Yes	Yes

Condition keys	Description	Example	Available for anonymous (unauthenticated) caller?	Available for gRPC?
		789012:service/svc-0123456789abcdef0		
`vpc-lattice-svcs:SourceVpc`	Filters access by the VPC the request is made from	vpc-1a2b3c4d	Yes	Yes
`vpc-lattice-svcs:SourceVpcOwnerAccount`	Filters access by the owning account of the VPC the request is made from	123456789012	Yes	Yes

Anonymous (unauthenticated) principals

Anonymous principals are callers that don't sign their AWS requests with Signature Version 4 (SigV4), and are within a VPC that is connected to the service network. Anonymous principals can make unauthenticated requests to services in the service network if an auth policy allows it.

Example auth policies

The following are example auth policies that require requests to be made by authenticated principals.

All examples use the `us-west-2` Region and contain fictitious account IDs.

Example 1: Restrict access to services by a specific AWS organization

The following auth policy example grants permissions to any authenticated request to access any services in the service network to which the policy applies. However, the request must originate from principals that belong to the AWS organization specified in the condition.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "vpc-lattice-svcs:Invoke",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "aws:PrincipalOrgID": [
                  "o-123456example"
               ]
            }
         }
      }
   ]
}
```

Example 2: Restrict access to a service by a specific IAM role

The following auth policy example grants permissions to any authenticated request that uses the IAM role `rates-client` to make HTTP GET requests on the service specified in the `Resource` element. The resource in the `Resource` element is the same as the service that the policy is attached to.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": [
               "arn:aws:iam::123456789012:role/rates-client"
            ]
         },
```

```
         "Action": "vpc-lattice-svcs:Invoke",
         "Resource": [
            "arn:aws:vpc-lattice:us-
west-2:123456789012:service/svc-0123456789abcdef0/*"
         ],
         "Condition": {
            "StringEquals": {
               "vpc-lattice-svcs:RequestMethod": "GET"
            }
         }
      }
   ]
}
```

Example 3: Restrict access to services by authenticated principals in a specific VPC

The following auth policy example only allows authenticated requests from principals in the VPC whose VPC ID is *vpc-1a2b3c4d*.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "vpc-lattice-svcs:Invoke",
         "Resource": "*",
         "Condition": {
            "StringNotEquals": {
               "aws:PrincipalType": "Anonymous"
            },
            "StringEquals": {
               "vpc-lattice-svcs:SourceVpc": "vpc-1a2b3c4d"
            }
         }
      }
   ]
}
```

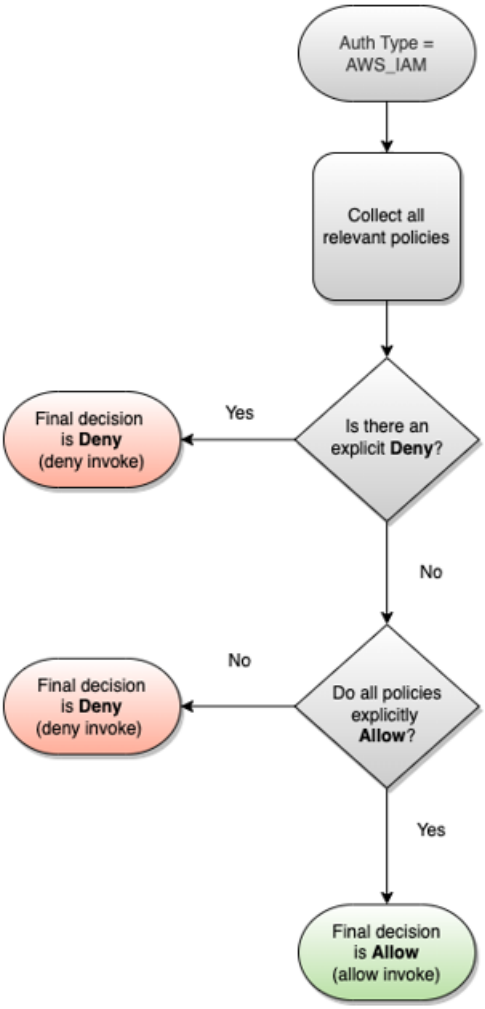
How authorization works

When a VPC Lattice service receives a request, the AWS enforcement code evaluates all relevant permissions policies together to determine whether to authorize or deny the request. It evaluates all the IAM identity-based policies and auth policies that are applicable in the request context during authorization. By default, all requests are implicitly denied when the auth type is `AWS_IAM`. An explicit allow from all relevant policies overrides the default.

Authorization includes:

- Collecting all the relevant IAM identity-based policies and auth policies.
- Evaluating the resulting set of policies:
  - Verifying that the requester (such as an IAM user or role) has permissions to perform the operation from the account to which the requester belongs. If there is no explicit allow statement, AWS does not authorize the request.
  - Verifying that the request is allowed by the auth policy for the service network. If an auth policy is enabled, but there is no explicit allow statement, AWS does not authorize the request. If there is an explicit allow statement, or the auth type is `NONE`, the code continues.
  - Verifying that the request is allowed by the auth policy for the service. If an auth policy is enabled, but there is no explicit allow statement, AWS does not authorize the request. If there is an explicit allow statement,or the auth type is `NONE`, then the enforcement code returns a final decision of **Allow**.
  - An explicit deny in any policy overrides any allows.

The diagram shows the authorization workflow. When a request is made, the relevant policies allow or deny the request access to a given service.



Control traffic in VPC Lattice using security groups

AWS security groups act as virtual firewalls, controlling the network traffic to and from the entities that they are associated with. With VPC Lattice, you can create security groups and assign them to the VPC association that connects a VPC to a service network to enforce additional network-level security protections for your service network. If you connect a VPC to a service network using a VPC endpoint, you can assign security groups to the VPC endpoint too. Similarly you can assign security groups to resource gateways that you create to enable access to resources in your VPC.

Contents

Managed prefix list

VPC Lattice provides managed prefix lists that includes the IP addresses used to route traffic over the VPC Lattice network when you use a service-network association to connect your VPC to a service network using a VPC association. These IPs are either private link-local IPs or non-routeable public IPs.

You can reference the VPC Lattice managed prefix lists in your security group rules. This allows traffic to flow from clients, through the VPC Lattice service network, and to the VPC Lattice service targets.

For example, suppose that you have an EC2 instance registered as a target in the US West (Oregon) Region (`us-west-2`). You can add a rule to the instance security group that allows inbound HTTPS access from the VPC Lattice managed prefix list, so that the VPC Lattice traffic in this Region can reach the instance. If you remove all other inbound rules from the security group, you can prevent any traffic other than VPC Lattice traffic from reaching the instance.

The names of the managed prefix lists for VPC Lattice are as follows:

- com.amazonaws.*region*.vpc-lattice
- com.amazonaws.*region*.ipv6.vpc-lattice

For more information, see AWS-managed prefix lists in the *Amazon VPC User Guide*.

Windows and macOS clients

The addresses in the VPC Lattice prefix lists are link-local addresses and non-routable public addresses. If you connect to VPC Lattice from these clients, you must update their configurations so that it forwards the IP addresses in the managed prefix list to the primary IP address for the client. The following is an example command that updates the configuration of the Windows client, where 169.254.171.0 is one of the addresses in the managed prefix list.

```
C:\> route add 169.254.171.0 mask 255.255.255.0 primary-ip-address
```

The following is an example command that updates the configuration of the macOS client, where 169.254.171.0 is one of the addresses in the managed prefix list.

```
sudo route -n add -net 169.254.171.0 primary-ip-address 255.255.255.0
```

To avoid creating a static route, we recommend that you use a service network endpoint in a VPC to establish connectivity. For more information, see the section called “Manage VPC endpoint associations”.

Security group rules

Using VPC Lattice with or without security groups will not impact your existing VPC security group configuration. However, you can add your own security groups at any time.

Key considerations

- Security group rules for clients control outbound traffic to VPC Lattice.
- Security group rules for targets control inbound traffic from VPC Lattice to the targets, including health check traffic.
- Security group rules for the association between the service network and VPC control which clients can access the VPC Lattice service network.
- Security group rules for resource gateway control outbound traffic from the resource gateway to resources.

Recommended outbound rules for traffic flowing from resource gateway to a database resource

For traffic to flow from resource gateway to resources, you must create outbound rules for the open ports and accepted listener protocols for the resources.

Destination	Protocol	Port range	Comment
CIDR range for resource	*TCP*	*3306*	Allow traffic from resource gateway to databases

Recommended inbound rules for service network and VPC associations

For traffic to flow from client VPCs to the services associated with the service network, you must create inbound rules for the listener ports and listener protocols for the services.

Source	Protocol	Port range	Comment
VPC CIDR	*listener*	*listener*	Allow traffic from clients to VPC Lattice

Recommended outbound rules for traffic flowing from client instances to VPC Lattice

By default, security groups allow all outbound traffic. However, if you have custom outbound rules, you must allow outbound traffic to VPC Lattice prefix for listener ports and protocols so that client instances can connect to all services associated with the VPC Lattice service network. You can allow this traffic by referencing the ID of the prefix list for VPC Lattice.

Destination	Protocol	Port range	Comment
ID of the VPC Lattice prefix list	*listener*	*listener*	Allow traffic from clients to VPC Lattice

Recommended inbound rules for traffic flowing from VPC Lattice to target instances

You can't use the client security group as a source for your target's security groups, because traffic flows from VPC Lattice. You can reference the ID of the prefix list for VPC Lattice.

Source	Protocol	Port range	Comment
ID of the VPC Lattice prefix list	*target*	*target*	Allow traffic from VPC Lattice to targets
ID of the VPC Lattice prefix list	*health check*	*health check*	Allow health check traffic from VPC Lattice to targets

Manage security groups for a VPC association

You can use the AWS CLI to view, add, or update security groups on the VPC to service network association. When using the AWS CLI, remember that your commands run in the AWS Region configured for your profile. If you want to run the commands in a different Region, either change the default Region for your profile, or use the `--region` parameter with the command.

Before you begin, confirm that you have created the security group in the same VPC as the VPC you want to add to the service network. For more information, see Control traffic to your resources using security groups in the *Amazon VPC User Guide*

To add a security group when you create a VPC association using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.
3. Select the name of the service network to open its details page.
4. On the **VPC associations** tab, choose **Create VPC associations** and then choose **Add VPC association**.
5. Select a VPC and up to five security groups.
6. Choose **Save changes**.

To add or update security groups for an existing VPC association using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, under **VPC Lattice**, choose **Service networks**.
3. Select the name of the service network to open its details page.
4. On the **VPC associations** tab, select the check box for the association and then choose **Actions**, **Edit security groups**.
5. Add and remove security groups as needed.
6. Choose **Save changes**.

To add a security group when you create a VPC association using the AWS CLI

Use the create-service-network-vpc-association command, specifying the ID of the VPC for the VPC association and the ID of the security groups to add.

```
aws vpc-lattice create-service-network-vpc-association \
    --service-network-identifier sn-0123456789abcdef0 \
    --vpc-identifier vpc-1a2b3c4d \
    --security-group-ids sg-7c2270198example
```

If successful, the command returns output similar to the following.

```
{
  "arn": "arn",
  "createdBy": "464296918874",
  "id": "snva-0123456789abcdef0",
  "status": "CREATE_IN_PROGRESS",
  "securityGroupIds": ["sg-7c2270198example"]
}
```

To add or update security groups for an existing VPC association using the AWS CLI

Use the update-service-network-vpc-association command, specifying the ID of the service network and the IDs of the security groups. These security groups override any previously associated security groups. Define at least one security group when updating the list.

```
aws vpc-lattice update-service-network-vpc-association
    --service-network-vpc-association-identifier sn-903004f88example \
    --security-group-ids sg-7c2270198example sg-903004f88example
```

> **Warning**
>
> You can't remove all security groups. Instead, you must first delete the VPC association, and then re-create the VPC association without any security groups. Be cautious when deleting the VPC association. This prevents traffic from reaching services that are in that service network.

Control traffic to VPC Lattice using network ACLs

A network access control list (ACL) allows or denies specific inbound or outbound traffic at the subnet level. The default network ACL allows all inbound and outbound traffic. You can create custom network ACLs for your subnets to provide an additional layer of security. For more information, see Network ACLs in the *Amazon VPC User Guide*.

Contents

Network ACLs for your client subnets

The network ACLs for client subnets must allow traffic between clients and VPC Lattice. You can get the IP address ranges to allow from the managed prefix list for VPC Lattice.

The following is an example inbound rule.

Source	Protocol	Port range	Comment
vpc_lattice_cidr_block	TCP	1025-65535	Allow traffic from VPC Lattice to clients

The following is an example outbound rule.

Destination	Protocol	Port range	Comment
vpc_lattice_cidr_block	*listener*	*listener*	Allow traffic from clients to VPC Lattice

Network ACLs for your target subnets

The network ACLs for target subnets must allow traffic between targets and VPC Lattice on both the target port and the health check port. You can get the IP address ranges to allow from the managed prefix list for VPC Lattice.

The following is an example inbound rule.

Source	Protocol	Port range	Comment
vpc_lattice_cidr_block	*target*	*target*	Allow traffic from VPC Lattice to targets

Source	Protocol	Port range	Comment
vpc_lattice_cidr_block	*health check*	*health check*	Allow health check traffic from VPC Lattice to targets

The following is an example outbound rule.

Destination	Protocol	Port range	Comment
vpc_lattice_cidr_block	*target*	1024-65535	Allow traffic from targets to VPC Lattice
vpc_lattice_cidr_block	*health check*	1024-65535	Allow health check traffic from targets to VPC Lattice

SIGv4 authenticated requests for Amazon VPC Lattice

VPC Lattice uses Signature Version 4 (SIGv4) or Signature Version 4A (SIGv4A) for client authentication. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Considerations

- VPC Lattice attempts to authenticate any request that is signed with SIGv4 or SIGv4A. The request fails without authentication.
- VPC Lattice does not support payload signing. You must send an `x-amz-content-sha256` header with the value set to `"UNSIGNED-PAYLOAD"`.

Examples

- Python
- Java
- Node.js
- Golang

- Golang - GRPC

Python

This example sends the signed requests over a secure connection to a service registered in the network. If you prefer to use requests, the botocore package simplifies the authentication process, but is not strictly required. For more information, see Credentials in the Boto3 documentation.

To install the `botocore` and `awscrt` packages, use the following command. For more information, see AWS CRT Python.

```
pip install botocore awscrt
```

If you run the client application on Lambda, install the required modules using Lambda layers, or include them in your deployment package.

In the following example, replace the placeholder values with your own values.

SIGv4

```
from botocore import crt
import requests
from botocore.awsrequest import AWSRequest
import botocore.session

if __name__ == '__main__':
    session = botocore.session.Session()
    signer = crt.auth.CrtSigV4Auth(session.get_credentials(), 'vpc-lattice-svcs',
 'us-west-2')
    endpoint = 'https://data-svc-022f67d3a42.1234abc.vpc-lattice-svcs.us-
west-2.on.aws'
    data = "some-data-here"
    headers = {'Content-Type': 'application/json', 'x-amz-content-sha256':
 'UNSIGNED-PAYLOAD'}
    request = AWSRequest(method='POST', url=endpoint, data=data, headers=headers)
    request.context["payload_signing_enabled"] = False
    signer.add_auth(request)

    prepped = request.prepare()

    response = requests.post(prepped.url, headers=prepped.headers, data=data)
```

```
    print(response.text)
```

SIGv4A

```
from botocore import crt
import requests
from botocore.awsrequest import AWSRequest
import botocore.session

if __name__ == '__main__':
    session = botocore.session.Session()
    signer = crt.auth.CrtSigV4AsymAuth(session.get_credentials(), 'vpc-lattice-
svcs', '*')
    endpoint = 'https://data-svc-022f67d3a42.1234abc.vpc-lattice-svcs.us-
west-2.on.aws'
    data = "some-data-here"
    headers = {'Content-Type': 'application/json', 'x-amz-content-sha256':
 'UNSIGNED-PAYLOAD'}
    request = AWSRequest(method='POST', url=endpoint, data=data, headers=headers)
    request.context["payload_signing_enabled"] = False
    signer.add_auth(request)

    prepped = request.prepare()

    response = requests.post(prepped.url, headers=prepped.headers, data=data)
    print(response.text)
```

Java

This example shows how you can perform request signing by using custom interceptors. It uses the default credentials provider class from AWS SDK for Java 2.x, which gets the correct credentials for you. If you would prefer to use a specific credential provider, you can select one from the AWS SDK for Java 2.x. The AWS SDK for Java allows only unsigned payloads over HTTPS. However, you can extend the signer to support unsigned payloads over HTTP.

SIGv4

```
package com.example;

import software.amazon.awssdk.http.auth.aws.signer.AwsV4HttpSigner;
import software.amazon.awssdk.http.auth.spi.signer.SignedRequest;
```

```
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.identity.spi.AwsCredentialsIdentity;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import java.io.IOException;
import java.net.URI;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;

public class sigv4 {

    public static void main(String[] args) {
        AwsV4HttpSigner signer = AwsV4HttpSigner.create();

        AwsCredentialsIdentity credentials =
 DefaultCredentialsProvider.create().resolveCredentials();

        if (args.length < 2) {
            System.out.println("Usage: sample <url> <region>");
            System.exit(1);
        }
        // Create the HTTP request to be signed
        var url = args[0];
        SdkHttpRequest httpRequest = SdkHttpRequest.builder()
                .uri(URI.create(url))
                .method(SdkHttpMethod.GET)
                .build();

        SignedRequest signedRequest = signer.sign(r -> r.identity(credentials)
                .request(httpRequest)
                .putProperty(AwsV4HttpSigner.SERVICE_SIGNING_NAME, "vpc-lattice-
svcs")
                .putProperty(AwsV4HttpSigner.PAYLOAD_SIGNING_ENABLED, false)
                .putProperty(AwsV4HttpSigner.REGION_NAME, args[1]));

        System.out.println("[*] Raw request headers:");
        signedRequest.request().headers().forEach((key, values) -> {
            values.forEach(value -> System.out.println("  " + key + ": " + value));
        });
```

```
        try (SdkHttpClient httpClient = ApacheHttpClient.create()) {
            HttpExecuteRequest httpExecuteRequest = HttpExecuteRequest.builder()
                    .request(signedRequest.request())
                    .contentStreamProvider(signedRequest.payload().orElse(null))
                    .build();

            System.out.println("[*] Sending request to: " + url);

            HttpExecuteResponse httpResponse =
 httpClient.prepareRequest(httpExecuteRequest).call();

            System.out.println("[*] Request sent");

            System.out.println("[*] Response status code: " +
 httpResponse.httpResponse().statusCode());
            // Read and print the response body
            httpResponse.responseBody().ifPresent(inputStream -> {
                try {
                    String responseBody = new String(inputStream.readAllBytes());
                    System.out.println("[*] Response body: " + responseBody);
                } catch (IOException e) {
                    System.err.println("[*] Failed to read response body");
                    e.printStackTrace();
                } finally {
                    try {
                        inputStream.close();
                    } catch (IOException e) {
                        System.err.println("[*] Failed to close input stream");
                        e.printStackTrace();
                    }
                }
            });
        } catch (IOException e) {
            System.err.println("[*] HTTP Request Failed.");
            e.printStackTrace();
        }

    }
}
```

SIGv4A

This example requires an additional dependency on `software.amazon.awssdk:http-auth-aws-crt`.

```
package com.example;


import software.amazon.awssdk.http.auth.aws.signer.AwsV4aHttpSigner;
import software.amazon.awssdk.http.auth.aws.signer.RegionSet;
import software.amazon.awssdk.http.auth.spi.signer.SignedRequest;

import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.identity.spi.AwsCredentialsIdentity;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import java.io.IOException;
import java.net.URI;
import java.util.Arrays;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;

public class sigv4a {

    public static void main(String[] args) {
        AwsV4aHttpSigner signer = AwsV4aHttpSigner.create();

        AwsCredentialsIdentity credentials =
 DefaultCredentialsProvider.create().resolveCredentials();

        if (args.length < 2) {
            System.out.println("Usage: sample <url> <regionset>");
            System.exit(1);
        }
        // Create the HTTP request to be signed
        var url = args[0];
        SdkHttpRequest httpRequest = SdkHttpRequest.builder()
                .uri(URI.create(url))
                .method(SdkHttpMethod.GET)
                .build();

        SignedRequest signedRequest = signer.sign(r -> r.identity(credentials)
                .request(httpRequest)
                .putProperty(AwsV4aHttpSigner.SERVICE_SIGNING_NAME, "vpc-lattice-
svcs")
```

```
                .putProperty(AwsV4aHttpSigner.PAYLOAD_SIGNING_ENABLED, false)
                .putProperty(AwsV4aHttpSigner.REGION_SET,
RegionSet.create(String.join(" ",Arrays.copyOfRange(args, 1, args.length)))));

        System.out.println("[*] Raw request headers:");
        signedRequest.request().headers().forEach((key, values) -> {
            values.forEach(value -> System.out.println("  " + key + ": " + value));
        });

        try (SdkHttpClient httpClient = ApacheHttpClient.create()) {
            HttpExecuteRequest httpExecuteRequest = HttpExecuteRequest.builder()
                    .request(signedRequest.request())
                    .contentStreamProvider(signedRequest.payload().orElse(null))
                    .build();

            System.out.println("[*] Sending request to: " + url);

            HttpExecuteResponse httpResponse =
httpClient.prepareRequest(httpExecuteRequest).call();

            System.out.println("[*] Request sent");

            System.out.println("[*] Response status code: " +
httpResponse.httpResponse().statusCode());
            // Read and print the response body
            httpResponse.responseBody().ifPresent(inputStream -> {
                try {
                    String responseBody = new String(inputStream.readAllBytes());
                    System.out.println("[*] Response body: " + responseBody);
                } catch (IOException e) {
                    System.err.println("[*] Failed to read response body");
                    e.printStackTrace();
                } finally {
                    try {
                        inputStream.close();
                    } catch (IOException e) {
                        System.err.println("[*] Failed to close input stream");
                        e.printStackTrace();
                    }
                }
            });
        } catch (IOException e) {
            System.err.println("[*] HTTP Request Failed.");
            e.printStackTrace();
```

```
        }
    }
}
```

Node.js

This example uses aws-crt NodeJS bindings to send a signed request using HTTPS.

To install the `aws-crt` package, use the following command.

```
npm -i aws-crt
```

If the `AWS_REGION` environment variable exists, the example uses the Region specified by `AWS_REGION`. The default Region is `us-east-1`.

SIGv4

```
const https = require('https')
const crt = require('aws-crt')
const { HttpRequest } = require('aws-crt/dist/native/http')

function sigV4Sign(method, endpoint, service, algorithm) {
    const host = new URL(endpoint).host
    const request = new HttpRequest(method, endpoint)
    request.headers.add('host', host)
    // crt.io.enable_logging(crt.io.LogLevel.INFO)
    const config = {
        service: service,
        region: process.env.AWS_REGION ? process.env.AWS_REGION : 'us-east-1',
        algorithm: algorithm,
        signature_type: crt.auth.AwsSignatureType.HttpRequestViaHeaders,
        signed_body_header: crt.auth.AwsSignedBodyHeaderType.XAmzContentSha256,
        signed_body_value: crt.auth.AwsSignedBodyValue.UnsignedPayload,
        provider: crt.auth.AwsCredentialsProvider.newDefault()
    }

    return crt.auth.aws_sign_request(request, config)
}

if (process.argv.length === 2) {
  console.error(process.argv[1] + ' <url>')
  process.exit(1)
```

```
}

const algorithm = crt.auth.AwsSigningAlgorithm.SigV4;

sigV4Sign('GET', process.argv[2], 'vpc-lattice-svcs', algorithm).then(
  httpResponse => {
    var headers = {}

    for (const sigv4header of httpResponse.headers) {
      headers[sigv4header[0]] = sigv4header[1]
    }

    const options = {
      hostname: new URL(process.argv[2]).host,
      path: new URL(process.argv[2]).pathname,
      method: 'GET',
      headers: headers
    }

    req = https.request(options, res => {
      console.log('statusCode:', res.statusCode)
      console.log('headers:', res.headers)
      res.on('data', d => {
        process.stdout.write(d)
      })
    })
    req.on('error', err => {
      console.log('Error: ' + err)
    })
    req.end()
  }
)
```

SIGv4A

```
const https = require('https')
const crt = require('aws-crt')
const { HttpRequest } = require('aws-crt/dist/native/http')

function sigV4Sign(method, endpoint, service, algorithm) {
    const host = new URL(endpoint).host
    const request = new HttpRequest(method, endpoint)
    request.headers.add('host', host)
```

```
    // crt.io.enable_logging(crt.io.LogLevel.INFO)
    const config = {
        service: service,
        region: process.env.AWS_REGION ? process.env.AWS_REGION : 'us-east-1',
        algorithm: algorithm,
        signature_type: crt.auth.AwsSignatureType.HttpRequestViaHeaders,
        signed_body_header: crt.auth.AwsSignedBodyHeaderType.XAmzContentSha256,
        signed_body_value: crt.auth.AwsSignedBodyValue.UnsignedPayload,
        provider: crt.auth.AwsCredentialsProvider.newDefault()
    }

    return crt.auth.aws_sign_request(request, config)
}

if (process.argv.length === 2) {
  console.error(process.argv[1] + ' <url>')
  process.exit(1)
}

const algorithm = crt.auth.AwsSigningAlgorithm.SigV4Asymmetric;

sigV4Sign('GET', process.argv[2], 'vpc-lattice-svcs', algorithm).then(
  httpResponse => {
    var headers = {}

    for (const sigv4header of httpResponse.headers) {
      headers[sigv4header[0]] = sigv4header[1]
    }

    const options = {
      hostname: new URL(process.argv[2]).host,
      path: new URL(process.argv[2]).pathname,
      method: 'GET',
      headers: headers
    }

    req = https.request(options, res => {
      console.log('statusCode:', res.statusCode)
      console.log('headers:', res.headers)
      res.on('data', d => {
        process.stdout.write(d)
      })
    })
    req.on('error', err => {
```

```
      console.log('Error: ' + err)
    })
    req.end()
  }
)
```

Golang

This example uses the Smithy code generators for Go and the AWS SDK for the Go programming language to handle request signing requests. The example requires a Go version of 1.21 or higher.

SIGv4

```
package main

import (
        "context"
        "flag"
        "fmt"
        "io"
        "log"
        "net/http"
        "net/http/httputil"
        "os"
        "strings"

        "github.com/aws/aws-sdk-go-v2/aws"
        "github.com/aws/aws-sdk-go-v2/config"
        "github.com/aws/smithy-go/aws-http-auth/credentials"
        "github.com/aws/smithy-go/aws-http-auth/sigv4"
        v4 "github.com/aws/smithy-go/aws-http-auth/v4"
)

type nopCloser struct {
        io.ReadSeeker
}

func (nopCloser) Close() error {
        return nil
}

type stringFlag struct {
```

```
        set   bool
        value string
}



        flag.PrintDefaults()
        os.Exit(1)
}

func main() {
        flag.Parse()
        if !url.set || !region.set {
                Usage()
        }

        cfg, err := config.LoadDefaultConfig(context.TODO(),
 config.WithClientLogMode(aws.LogSigning))
        if err != nil {
                log.Fatalf("failed to load SDK configuration, %v", err)
        }

        if len(os.Args) < 2 {
                log.Fatalf("Usage: go run main.go  <url>")
        }

        // Retrieve credentials from an SDK source, such as the instance profile
        sdkCreds, err := cfg.Credentials.Retrieve(context.TODO())
        if err != nil {
                log.Fatalf("Unable to retrieve credentials from SDK, %v", err)
        }

        creds := credentials.Credentials{
                AccessKeyID:     sdkCreds.AccessKeyID,
                SecretAccessKey: sdkCreds.SecretAccessKey,
                SessionToken:    sdkCreds.SessionToken,
        }

        // Add a payload body, which will not be part of the signature calculation
        body := nopCloser{strings.NewReader(`Example payload body`)}

        req, _ := http.NewRequest(http.MethodPost, url.value, body)

        // Create a sigv4a signer with specific options
        signer := sigv4.New(func(o *v4.SignerOptions) {
```

```
                o.DisableDoublePathEscape = true
                // This will add the UNSIGNED-PAYLOAD sha256 header
                o.AddPayloadHashHeader = true
                o.DisableImplicitPayloadHashing = true
        })

        // Perform the signing on req, using the credentials we retrieved from the
SDK
        err = signer.SignRequest(&sigv4.SignRequestInput{
                Request:     req,
                Credentials: creds,
                Service:     "vpc-lattice-svcs",
                Region: region.String(),
        })

        if err != nil {
                log.Fatalf("%s", err)
        }

        res, err := httputil.DumpRequest(req, true)

        if err != nil {
                log.Fatalf("%s", err)
        }

        log.Printf("[*] Raw request\n%s\n", string(res))

        log.Printf("[*] Sending request to %s\n", url.value)

        resp, err := http.DefaultClient.Do(req)
        if err != nil {
                log.Fatalf("%s", err)
        }

        log.Printf("[*] Request sent\n")

        log.Printf("[*] Response status code: %d\n", resp.StatusCode)

        respBody, err := io.ReadAll(resp.Body)
        if err != nil {
                log.Fatalf("%s", err)
        }

        log.Printf("[*] Response body: \n%s\n", respBody)
```

```
}
```

SIGv4A

```
package main

import (
        "context"
        "flag"
        "fmt"
        "io"
        "log"
        "net/http"
        "net/http/httputil"
        "os"
        "strings"

        "github.com/aws/aws-sdk-go-v2/aws"
        "github.com/aws/aws-sdk-go-v2/config"
        "github.com/aws/smithy-go/aws-http-auth/credentials"
        "github.com/aws/smithy-go/aws-http-auth/sigv4a"
        v4 "github.com/aws/smithy-go/aws-http-auth/v4"
)

type nopCloser struct {
        io.ReadSeeker
}

func (nopCloser) Close() error {
        return nil
}

type stringFlag struct {

func main() {
        flag.Parse()
        if !url.set || !regionSet.set {
                Usage()
        }

        cfg, err := config.LoadDefaultConfig(context.TODO(),
 config.WithClientLogMode(aws.LogSigning))
        if err != nil {
```

```
                log.Fatalf("failed to load SDK configuration, %v", err)
        }

        if len(os.Args) < 2 {
                log.Fatalf("Usage: go run main.go <url>")
        }

        // Retrieve credentials from an SDK source, such as the instance profile
        sdkCreds, err := cfg.Credentials.Retrieve(context.TODO())
        if err != nil {
                log.Fatalf("Unable to retrieve credentials from SDK, %v", err)
        }

        creds := credentials.Credentials{
                AccessKeyID:     sdkCreds.AccessKeyID,
                SecretAccessKey: sdkCreds.SecretAccessKey,
                SessionToken:    sdkCreds.SessionToken,
        }

        // Add a payload body, which will not be part of the signature calculation
        body := nopCloser{strings.NewReader(`Example payload body`)}

        req, _ := http.NewRequest(http.MethodPost, url.value, body)

        // Create a sigv4a signer with specific options
        signer := sigv4a.New(func(o *v4.SignerOptions) {
                o.DisableDoublePathEscape = true
                // This will add the UNSIGNED-PAYLOAD sha256 header
                o.AddPayloadHashHeader = true
                o.DisableImplicitPayloadHashing = true
        })

        // Create a slice out of the provided regionset
        rs := strings.Split(regionSet.value, ",")

        // Perform the signing on req, using the credentials we retrieved from the
SDK
        err = signer.SignRequest(&sigv4a.SignRequestInput{
                Request:     req,
                Credentials: creds,
                Service:     "vpc-lattice-svcs",
                RegionSet: rs,
        })
```

```
        if err != nil {
                log.Fatalf("%s", err)
        }

        res, err := httputil.DumpRequest(req, true)

        if err != nil {
                log.Fatalf("%s", err)
        }

        log.Printf("[*] Raw request\n%s\n", string(res))

        log.Printf("[*] Sending request to %s\n", url.value)

        resp, err := http.DefaultClient.Do(req)
        if err != nil {
                log.Fatalf("%s", err)
        }

        log.Printf("[*] Request sent\n")

        log.Printf("[*] Response status code: %d\n", resp.StatusCode)

        respBody, err := io.ReadAll(resp.Body)
        if err != nil {
                log.Fatalf("%s", err)
        }

        log.Printf("[*] Response body: \n%s\n", respBody)
}
```

Golang - GRPC

This example uses the AWS SDK for the Go programming language to handle request signing for GRPC requests. This can be used with the echo server from the GRPC sample code repository.

```
package main

import (
    "context"
    "crypto/tls"
    "crypto/x509"
```

```
    "flag"
    "fmt"
    "log"
    "net/http"
    "net/url"
    "strings"
    "time"

    "google.golang.org/grpc"
    "google.golang.org/grpc/credentials"

    "github.com/aws/aws-sdk-go-v2/aws"
    v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
    "github.com/aws/aws-sdk-go-v2/config"

    ecpb "google.golang.org/grpc/examples/features/proto/echo"
)

const (
    headerContentSha    = "x-amz-content-sha256"
    headerSecurityToken = "x-amz-security-token"
    headerDate          = "x-amz-date"
    headerAuthorization = "authorization"
    unsignedPayload     = "UNSIGNED-PAYLOAD"
)

type SigV4GrpcSigner struct {
    service      string
    region       string
    credProvider aws.CredentialsProvider
    signer       *v4.Signer
}

func NewSigV4GrpcSigner(service string, region string, credProvider
 aws.CredentialsProvider) *SigV4GrpcSigner {
    signer := v4.NewSigner()
    return &SigV4GrpcSigner{
        service:      service,
        region:       region,
        credProvider: credProvider,
        signer:       signer,
    }
}
```

```
func (s *SigV4GrpcSigner) GetRequestMetadata(ctx context.Context, uri ...string)
 (map[string]string, error) {
    ri, _ := credentials.RequestInfoFromContext(ctx)
    creds, err := s.credProvider.Retrieve(ctx)
    if err != nil {
        return nil, fmt.Errorf("failed to load credentials: %w", err)
    }

    // The URI we get here is scheme://authority/service/ - for siging we want to
 include the RPC name
    // But RequestInfoFromContext only has the combined /service/rpc-name - so read the
 URI, and
    // replace the Path with what we get from RequestInfo.
    parsed, err := url.Parse(uri[0])
    if err != nil {
        return nil, err
    }
    parsed.Path = ri.Method

    // Build a request for the signer.
    bodyReader := strings.NewReader("")
    req, err := http.NewRequest("POST", uri[0], bodyReader)
    if err != nil {
        return nil, err
    }
    date := time.Now()
    req.Header.Set(headerContentSha, unsignedPayload)
    req.Header.Set(headerDate, date.String())
    if creds.SessionToken != "" {
        req.Header.Set(headerSecurityToken, creds.SessionToken)
    }
    // The signer wants this as //authority/path
    // So get this by triming off the scheme and the colon before the first slash.
    req.URL.Opaque = strings.TrimPrefix(parsed.String(), parsed.Scheme+":")

    err = s.signer.SignHTTP(context.Background(), creds, req, unsignedPayload,
 s.service, s.region, date)
    if err != nil {
        return nil, fmt.Errorf("failed to sign request: %w", err)
    }

    // Pull the relevant headers out of the signer, and return them to get
    // included in the request we make.
```

```
    reqHeaders := map[string]string{
        headerContentSha:    req.Header.Get(headerContentSha),
        headerDate:          req.Header.Get(headerDate),
        headerAuthorization: req.Header.Get(headerAuthorization),
    }
    if req.Header.Get(headerSecurityToken) != "" {
        reqHeaders[headerSecurityToken] = req.Header.Get(headerSecurityToken)
    }

    return reqHeaders, nil
}

func (c *SigV4GrpcSigner) RequireTransportSecurity() bool {
    return true
}

var addr = flag.String("addr", "some-lattice-service:443", "the address to connect to")
var region = flag.String("region", "us-west-2", "region")

func callUnaryEcho(client ecpb.EchoClient, message string) {
    ctx, cancel := context.WithTimeout(context.Background(), 10*time.Second)
    defer cancel()
    resp, err := client.UnaryEcho(ctx, &ecpb.EchoRequest{Message: message})
    if err != nil {
        log.Fatalf("client.UnaryEcho(_) = _, %v: ", err)
    }
    fmt.Println("UnaryEcho: ", resp.Message)
}

func main() {
    flag.Parse()
    cfg, err := config.LoadDefaultConfig(context.TODO(),
 config.WithClientLogMode(aws.LogSigning))
    if err != nil {
        log.Fatalf("failed to load SDK configuration, %v", err)
    }

    pool, _ := x509.SystemCertPool()
    tlsConfig := &tls.Config{
        RootCAs: pool,
    }

    authority, _, _ := strings.Cut(*addr, ":") // Remove the port from the addr
    opts := []grpc.DialOption{
```

```
        grpc.WithTransportCredentials(credentials.NewTLS(tlsConfig)),

        // Lattice needs both the Authority to be set (without a port), and the SigV4
 signer
        grpc.WithAuthority(authority),
        grpc.WithPerRPCCredentials(NewSigV4GrpcSigner("vpc-lattice-svcs", *region,
 cfg.Credentials)),
    }

    conn, err := grpc.Dial(*addr, opts...)

    if err != nil {
        log.Fatalf("did not connect: %v", err)
    }
    defer conn.Close()
    rgc := ecpb.NewEchoClient(conn)

    callUnaryEcho(rgc, "hello world")
}
```

Data protection in Amazon VPC Lattice

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

Encryption in transit

VPC Lattice is a fully managed service that consists of a control plane and a data plane. Each plane serves a distinct purpose in the service. The control plane provides the administrative APIs used to create, read/describe, update, delete, and list (CRUDL) resources (for example, `CreateService` and `UpdateService`). Communications to the VPC Lattice control plane are protected in-transit by TLS. The data plane is the VPC Lattice Invoke API, which provides the interconnection between services. TLS encrypts communications to the VPC Lattice data plane when you use HTTPS or TLS. The cipher suite and protocol version use defaults provided by VPC Lattice and are not configurable. For more information, see HTTPS listeners for VPC Lattice services.

Encryption at rest

By default, encryption of data at rest helps reduce the operational overhead and complexity involved in protecting sensitive data. At the same time, it enables you to build secure applications that meet strict encryption compliance and regulatory requirements.

Contents

Server-side encryption with Amazon S3 managed keys (SSE-S3)

When you use server-side encryption with Amazon S3 managed keys (SSE-S3), each object is encrypted with a unique key. As an additional safeguard, we encrypt the key itself with a root key that we regularly rotates. Amazon S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256) GCM, to encrypt your data. For objects encrypted prior to AES-GCM, AES-CBC is still supported to decrypt those objects. For more information, see Using server-side encryption with Amazon S3-managed encryption keys (SSE-S3).

If you enable server-side encryption with Amazon S3-managed encryption keys (SSE-S3) for your S3 bucket for VPC Lattice access logs, we automatically encrypt each access log file before it is stored in your S3 bucket. For more information, see Logs sent to Amazon S3 in the *Amazon CloudWatch User Guide*.

Server-side encryption with AWS KMS keys stored in AWS KMS (SSE-KMS)

Server-side encryption with AWS KMS keys (SSE-KMS) is similar to SSE-S3, but with additional benefits and charges for using this service. There are separate permissions for the AWS KMS key that provides added protection against unauthorized access of your objects in Amazon S3. SSE-KMS also provides you with an audit trail that shows when your AWS KMS key was used and by whom. For more information, see Using server-side encryption with AWS Key Management Service (SSE-KMS).

Contents

Encryption and decryption of your certificate's private key

Your ACM certificate and private key are encrypted using an AWS managed KMS key that has the alias **aws/acm**. You can view the key ID with this alias in the AWS KMS console under **AWS managed keys**.

VPC Lattice does not directly access your ACM resources. It uses AWS TLS Connection Manager to secure and access the private keys for your certificate. When you use your ACM certificate to create a VPC Lattice service, VPC Lattice associates your certificate with AWS TLS Connection Manager. This is done by creating a grant in AWS KMS against your AWS Managed Key with the prefix **aws/acm**. A grant is a policy instrument that allows TLS Connection Manager to use KMS keys in cryptographic operations. The grant allows the grantee principal (TLS Connection Manager) to call the specified grant operations on the KMS key to decrypt your certificate's private key. TLS Connection Manager then uses the certificate and the decrypted (plaintext) private key to establish a secure connection (SSL/TLS session) with clients of VPC Lattice services. When the certificate is disassociated from a VPC Lattice service, the grant is retired.

If you want to remove access to the KMS key, we recommend that you replace or delete the certificate from the service using the AWS Management Console or the `update-service` command in the AWS CLI.

Encryption context for VPC Lattice

An encryption context is an optional set of key-value pairs that contain contextual information about what your private key might be used for. AWS KMS binds the encryption context to the encrypted data and uses it as additional authenticated data to support authenticated encryption.

When your TLS keys are used with VPC Lattice and TLS Connection manager, the name of your VPC Lattice service is included in the encryption context used to encrypt your key at rest. You can verify which VPC Lattice service your certificate and private key are being used for by viewing the encryption context in your CloudTrail logs as shown in the next section, or by looking at the **Associated Resources** tab in the ACM console.

To decrypt data, the same encryption context is included in the request. VPC Lattice uses the same encryption context in all AWS KMS cryptographic operations, where the key is `aws:vpc-lattice:arn` and the value is the Amazon Resource Name (ARN) of the VPC Lattice service.

The following example shows the encryption context in the output of an operation such as `CreateGrant`.

```
"encryptionContextEquals": {
    "aws:acm:arn": "arn:aws:acm:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "aws:vpc-lattice:arn": "arn:aws:vpc-lattice:us-west-2:111122223333:service/
svc-0b23c1234567890ab"
}
```

Monitoring your encryption keys for VPC Lattice

When you use an AWS managed key with your VPC Lattice service, you can use AWS CloudTrail to track requests that VPC Lattice sends to AWS KMS.

CreateGrant

When you add your ACM certificate to a VPC Lattice service, a `CreateGrant` request is sent on your behalf for TLS Connection Manager to be able to decrypt the private key associated with your ACM certificate

You can view the `CreateGrant` operation as an event in **CloudTrail**, **Event history**, **CreateGrant**.

The following is an example event record in the CloudTrail event history for the `CreateGrant` operation.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111122223333:user/Alice",
                "accountId": "111122223333",
                "userName": "Alice"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-02-06T23:30:50Z",
                "mfaAuthenticated": "false"
```

```
            }
        },
        "invokedBy": "acm.amazonaws.com"
    },
    "eventTime": "2023-02-07T00:07:18Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "acm.amazonaws.com",
    "userAgent": "acm.amazonaws.com",
    "requestParameters": {
        "granteePrincipal": "tlsconnectionmanager.amazonaws.com",
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextEquals": {
                "aws:acm:arn": "arn:aws:acm:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                "aws:vpc-lattice:arn": "arn:aws:vpc-lattice:us-
west-2:111122223333:service/svc-0b23c1234567890ab"
            }
        },
        "retiringPrincipal": "acm.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId": "f020fe75197b93991dc8491d6f19dd3cebb24ee62277a05914386724f3d48758",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "ba178361-8ab6-4bdd-9aa2-0d1a44b2974a",
    "eventID": "8d449963-1120-4d0c-9479-f76de11ce609",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

In the above `CreateGrant` example, the grantee principal is TLS Connection Manager, and the encryption context has the VPC Lattice service ARN.

ListGrants

You can use your KMS key ID and your account ID to call the `ListGrants` API. This gets you a list of all grants for the specified KMS key. For more information, see ListGrants.

Use the following `ListGrants` command in the AWS CLI to see the details of all the grants.

```
aws kms list-grants –key-id your-kms-key-id
```

The following is example output.

```
{
    "Grants": [
        {
            "Operations": [
                "Decrypt"
            ],
            "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "Name": "IssuedThroughACM",
            "RetiringPrincipal": "acm.us-west-2.amazonaws.com",
            "GranteePrincipal": "tlsconnectionmanager.amazonaws.com",
            "GrantId":
 "f020fe75197b93991dc8491d6f19dd3cebb24ee62277a05914386724f3d48758",
            "IssuingAccount": "arn:aws:iam::111122223333:root",
            "CreationDate": "2023-02-06T23:30:50Z",
            "Constraints": {
                "encryptionContextEquals": {
                  "aws:acm:arn": "arn:aws:acm:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                  "aws:vpc-lattice:arn": "arn:aws:vpc-lattice:us-
west-2:111122223333:service/svc-0b23c1234567890ab"
                }
            }
        }
```

```
    ]
}
```

In the above `ListGrants` example, the grantee principal is TLS Connection Manager and the encryption context has the VPC Lattice service ARN.

Decrypt

VPC Lattice uses TLS Connection Manager to call the `Decrypt` operation to decrypt your private key in order to serve TLS connections in your VPC Lattice service. You can view the `Decrypt` operation as an event in **CloudTrail Event history**, **Decrypt**.

The following is an example event record in the CloudTrail event history for the `Decrypt` operation.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "tlsconnectionmanager.amazonaws.com"
    },
    "eventTime": "2023-02-07T00:07:23Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "tlsconnectionmanager.amazonaws.com",
    "userAgent": "tlsconnectionmanager.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:acm:arn": "arn:aws:acm:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "aws:vpc-lattice:arn": "arn:aws:vpc-lattice:us-west-2:111122223333:service/
svc-0b23c1234567890ab"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "12345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
```

```
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "sharedEventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "eventCategory": "Management"
}
```

Identity and access management for Amazon VPC Lattice

The following sections describe how you can use AWS Identity and Access Management (IAM) to help secure your VPC Lattice resources, by controlling who can perform VPC Lattice API actions.

Topics

How Amazon VPC Lattice works with IAM

Before you use IAM to manage access to VPC Lattice, learn what IAM features are available to use with VPC Lattice.

IAM feature	VPC Lattice support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes

IAM feature	VPC Lattice support
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Service roles	No
Service-linked roles	Yes

For a high-level view of how VPC Lattice and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for VPC Lattice

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Resource-based policies within VPC Lattice

Supports resource-based policies: Yes

Resource-based policies are JSON policy documents that you attach to a resource in AWS. In AWS services that support resource-based policies, service administrators can use them to control access

to a specific resource of that AWS service. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy.

VPC Lattice supports *auth policies*, a resource-based policy that lets you control access to services in your service network. For more information, see Control access to VPC Lattice services using auth policies.

VPC Lattice also supports resource-based permissions policies for integration with AWS Resource Access Manager. You can use these resource-based policies to grant permission to manage connectivity to other AWS accounts or organizations for services, resource configurations, and service networks. For more information, see Share your VPC Lattice entities.

Policy actions for VPC Lattice

Supports policy actions: Yes

In an IAM policy statement, you can specify any API action from any service that supports IAM. For VPC Lattice, use the following prefix with the name of the API action: `vpc-lattice:`. For example: `vpc-lattice:CreateService`, `vpc-lattice:CreateTargetGroup`, and `vpc-lattice:PutAuthPolicy`.

To specify multiple actions in a single statement, separate them with commas, as follows:

```
"Action": [ "vpc-lattice:action1", "vpc-lattice:action2" ]
```

You can also specify multiple actions using wildcards. For example, you can specify all actions whose names begin with the word `Get`, as follows:

```
"Action": "vpc-lattice:Get*"
```

For a complete list of VPC Lattice API actions, see Actions defined by Amazon VPC Lattice in the *Service Authorization Reference*.

Policy resources for VPC Lattice

Supports policy resources: Yes

In an IAM policy statement, the `Resource` element specifies the object or objects that the statement covers. For VPC Lattice, each IAM policy statement applies to the resources that you specify using their ARNs.

The specific Amazon Resource Name (ARN) format depends on the resource. When you provide an ARN, replace the *italicized* text with your resource-specific information.

- **Access log subscriptions:**

```
"Resource": "arn:aws:vpc-lattice:region:account-id:accesslogsubscription/access-log-subscription-id"
```

- **Listeners:**

```
"Resource": "arn:aws:vpc-lattice:region:account-id:service/service-id/
listener/listener-id"
```

- **Resource gateways**

```
"Resource": "arn:aws:vpc-lattice:region:account-id:resourcegateway/resource-gateway-id"
```

- **Resource configuration**

```
"Resource": "arn:aws:vpc-lattice:region:account-id:resourceconfiguration/resource-configuration-id"
```

- **Rules:**

```
"Resource": "arn:aws:vpc-lattice:region:account-id:service/service-id/
listener/listener-id/rule/rule-id"
```

- **Services:**

```
"Resource": "arn:aws:vpc-lattice:region:account-id:service/service-id"
```

- **Service networks:**

```
"Resource": "arn:aws:vpc-lattice:region:account-id:servicenetwork/service-network-id"
```

- **Service network service associations:**

```
"Resource": "arn:aws:vpc-lattice:region:account-
id:servicenetworkserviceassociation/service-network-service-association-id"
```

- **Service network resource configuration associations**

```
"Resource": "arn:aws:vpc-lattice:region:account-
id:servicenetworkresourceassociation/service-network-resource-association-id"
```

- **Service network VPC associations:**

```
"Resource": "arn:aws:vpc-lattice:region:account-
id:servicenetworkvpcassociation/service-network-vpc-association-id"
```

- **Target groups:**

```
"Resource": "arn:aws:vpc-lattice:region:account-id:targetgroup/target-group-id"
```

Policy condition keys for VPC Lattice

Supports service-specific policy condition keys: Yes

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

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

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

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

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

To see a list of the VPC Lattice condition keys, see Condition keys for Amazon VPC Lattice in the *Service Authorization Reference*.

AWS supports global condition keys and service-specific condition keys. For information about AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

Access control lists (ACLs) in VPC Lattice

Supports ACLs: No

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

Attribute-based access control (ABAC) with VPC Lattice

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with VPC Lattice

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switch from a user to an IAM role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Service roles for VPC Lattice

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break VPC Lattice functionality. Edit service roles only when VPC Lattice provides guidance to do so.

Service-linked roles for VPC Lattice

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For information about creating or managing VPC Lattice service-linked roles, see Using service-linked roles for Amazon VPC Lattice.

Amazon VPC Lattice API permissions

You must grant IAM identities (such as users or roles) permission to call the VPC Lattice API actions they need, as described in Policy actions for VPC Lattice. In addition, for some VPC Lattice actions, you must grant IAM identities permission to call specific actions from other AWS APIs.

Required permissions for the API

When calling the following actions from the API, you must grant IAM users permission to call the specified actions.

`CreateResourceConfiguration`

- `vpc-lattice:CreateResourceConfiguration`
- `ec2:DescribeSubnets`
- `rds:DescribeDBInstances`
- `rds:DescribeDBClusters`

`CreateResourceGateway`

- `vpc-lattice:CreateResourceGateway`
- `ec2:AssignPrivateIpAddresses`
- `ec2:AssignIpv6Addresses`
- `ec2:CreateNetworkInterface`
- `ec2:CreateNetworkInterfacePermission`
- `ec2:DeleteNetworkInterface`
- `ec2:DescribeNetworkInterfaces`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSubnets`

`DeleteResourceGateway`

- `vpc-lattice:DeleteResourceGateway`
- `ec2:DeleteNetworkInterface`

`UpdateResourceGateway`

- `vpc-lattice:UpdateResourceGateway`

- ec2:AssignPrivateIpAddresses
- ec2:AssignIpv6Addresses
- ec2:UnassignPrivateIpAddresses
- ec2:CreateNetworkInterface
- ec2:CreateNetworkInterfacePermission
- ec2:DeleteNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- ec2:ModifyNetworkInterfaceAttribute

CreateServiceNetworkResourceAssociation

- vpc-lattice:CreateServiceNetworkResourceAssociation
- ec2:AssignIpv6Addresses
- ec2:CreateNetworkInterface
- ec2:CreateNetworkInterfacePermission
- ec2:DescribeNetworkInterfaces

CreateServiceNetworkVpcAssociation

- vpc-lattice:CreateServiceNetworkVpcAssociation
- ec2:DescribeVpcs
- ec2:DescribeSecurityGroups (Only needed when security groups are provided)

UpdateServiceNetworkVpcAssociation

- vpc-lattice:UpdateServiceNetworkVpcAssociation
- ec2:DescribeSecurityGroups (Only needed when security groups are provided)

CreateTargetGroup

- vpc-lattice:CreateTargetGroup
- ec2:DescribeVpcs

RegisterTargets

- vpc-lattice:RegisterTargets
- ec2:DescribeInstances (Only needed when INSTANCE is the target group type)

- `ec2:DescribeVpcs` (Only needed when `INSTANCE` or `IP` is the target group type)
- `ec2:DescribeSubnets` (Only needed when `INSTANCE` or `IP` is the target group type)
- `lambda:GetFunction` (Only needed when `LAMBDA` is the target group type)
- `lambda:AddPermission` (Only needed if the target group doesn't already have permission to invoke the specified Lambda function)

`DeregisterTargets`

- `vpc-lattice:DeregisterTargets`

`CreateAccessLogSubscription`

- `vpc-lattice:CreateAccessLogSubscription`
- `logs:GetLogDelivery`
- `logs:CreateLogDelivery`

`DeleteAccessLogSubscription`

- `vpc-lattice:DeleteAccessLogSubscription`
- `logs:DeleteLogDelivery`

`UpdateAccessLogSubscription`

- `vpc-lattice:UpdateAccessLogSubscription`
- `logs:UpdateLogDelivery`

Identity-based policies for Amazon VPC Lattice

By default, users and roles don't have permission to create or modify VPC Lattice resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by VPC Lattice, including the format of the ARNs for each of the resource types, see Actions, Resources, and Condition Keys for Amazon VPC Lattice in the *Service Authorization Reference*.

Contents

Policy best practices

Identity-based policies determine whether someone can create, access, or delete VPC Lattice resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API

operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Additional required permissions for full access

To use other AWS services that VPC Lattice is integrated with and the entire suite of VPC Lattice features, you must have specific additional permissions. These permissions are not included in the `VPCLatticeFullAccess` managed policy because of the confused deputy privilege escalation risk.

You must attach the following policy to your role and use it along with the `VPCLatticeFullAccess` managed policy.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "firehose:TagDeliveryStream",
                "lambda:AddPermission",
                "s3:PutBucketPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutResourcePolicy"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": [
                        "vpc-lattice.amazonaws.com"
                    ]
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/vpc-
lattice.amazonaws.com/AWSServiceRoleForVpcLattice"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
delivery.logs.amazonaws.com/AWSServiceRoleForLogDelivery*"
        }
    ]
}
```

This policy provides the following additional permissions:

- `iam:AttachRolePolicy`: Allows you to attach the specified managed policy to the specified IAM role.
- `iam:PutRolePolicy`: Allows you to add or update an inline policy document that is embedded in the specified IAM role.
- `s3:PutBucketPolicy`: Allows you to apply a bucket policy to an Amazon S3 bucket.
- `firehose:TagDeliveryStream`: Allows you to add or update tags for Firehose delivery streams.

Identity-based policy examples for VPC Lattice

Topics

- Example policy: Add tags to resources
- Example policy: Create a service-linked role

Example policy: Manage VPC associations to a service network

The following example demonstrates a policy that gives users with this policy the permission to create, update, and delete the VPC associations to a service network, but only for the VPC and service network specified in the condition. For more information about specifying condition keys, see Policy condition keys for VPC Lattice.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "vpc-lattice:CreateServiceNetworkVpcAssociation",
            "vpc-lattice:UpdateServiceNetworkVpcAssociation",
            "vpc-lattice:DeleteServiceNetworkVpcAssociation"
         ],
         "Resource": [
            "*"
         ],
         "Condition": {
            "StringEquals": {
               "vpc-lattice:ServiceNetworkArn": "arn:aws:vpc-lattice:us-
west-2:123456789012:servicenetwork/sn-903004f88example",
               "vpc-lattice:VpcId": "vpc-1a2b3c4d"
            }
         }
      }
   ]
}
```

Example policy: Create service associations to a service network

If you are not using condition keys to control access to VPC Lattice resources, you can specify the ARNs of resources in the `Resource` element to control access instead.

The following example demonstrates a policy that limits the service associations to a service network that users with this policy can create by specifying the ARNs of the service and service network that can be used with the `CreateServiceNetworkServiceAssociation` API action. For more information about specifying the ARN values, see Policy resources for VPC Lattice.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "vpc-lattice:CreateServiceNetworkServiceAssociation"
         ],
         "Resource": [
            "arn:aws:vpc-lattice:us-
west-2:123456789012:servicenetworkserviceassociation/*",
            "arn:aws:vpc-lattice:us-west-2:123456789012:service/
svc-04d5cc9b88example",
            "arn:aws:vpc-lattice:us-west-2:123456789012:servicenetwork/
sn-903004f88example"
         ]
      }
   ]
}
```

Example policy: Add tags to resources

The following example demonstrates a policy that gives users with this policy permission to create tags on VPC Lattice resources.

JSON

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

```
            "vpc-lattice:TagResource"
         ],
         "Resource": "arn:aws:vpc-lattice:us-west-2:123456789012:*/*"
      }
   ]
}
```

Example policy: Create a service-linked role

VPC Lattice requires permissions to create a service-linked role the first time that any user in your AWS account creates VPC Lattice resources. If the service-linked role does not exist already, VPC Lattice creates it in your account. The service-linked role gives permissions to VPC Lattice so that it can call other AWS services on your behalf. For more information, see the section called "Using service-linked roles".

For automatic role creation to succeed, users must have permissions for the `iam:CreateServiceLinkedRole` action.

```
"Action": "iam:CreateServiceLinkedRole"
```

The following example demonstrates a policy that gives users with this policy permission to create a service-linked role for VPC Lattice.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "iam:CreateServiceLinkedRole",
         "Resource": "arn:aws:iam::*:role/aws-service-role/vpc-
lattice.amazonaws.com/AWSServiceRoleForVpcLattice",
         "Condition": {
            "StringLike": {
               "iam:AWSServiceName":"vpc-lattice.amazonaws.com"
            }
         }
      }
   ]
```

```
}
```

For more information, see Service-linked role permissions in the *IAM User Guide*.

Using service-linked roles for Amazon VPC Lattice

Amazon VPC Lattice uses a service-linked role for the permissions that it requires to call other AWS services on your behalf. For more information, see Service-linked roles in the *IAM User Guide*.

VPC Lattice uses the service-linked role named AWSServiceRoleForVpcLattice.

Service-linked role permissions for VPC Lattice

The **AWSServiceRoleForVpcLattice** service-linked role trusts the following service to assume the role:

- `vpc-lattice.amazonaws.com`

The role permissions policy named AWSVpcLatticeServiceRolePolicy allows VPC Lattice to publish CloudWatch metrics in the `AWS/VpcLattice` namespace. For more information, see AWSVpcLatticeServiceRolePolicy in the *AWS Managed Policy Reference*.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see the section called “Example policy: Create a service-linked role”.

Create a service-linked role for VPC Lattice

You don't need to manually create a service-linked role. When you create VPC Lattice resources in the AWS Management Console, the AWS CLI, or the AWS API, VPC Lattice creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create VPC Lattice resources, VPC Lattice creates the service-linked role for you again.

Edit a service-linked role for VPC Lattice

You can edit the description of **AWSServiceRoleForVpcLattice** using IAM. For more information, see Edit a service-linked role description in the *IAM User Guide*.

Delete a service-linked role for VPC Lattice

If you no longer need to use Amazon VPC Lattice, we recommend that you delete **AWSServiceRoleForVpcLattice**.

You can delete this service-linked role only after you delete all VPC Lattice resources in your AWS account.

Use the IAM console, the AWS CLI, or the AWS API to delete the **AWSServiceRoleForVpcLattice** service-linked role. For more information, see Delete a service-linked role in the *IAM User Guide*.

After you delete a service-linked role, VPC Lattice creates the role again when you create VPC Lattice resources in your AWS account.

Supported Regions for VPC Lattice service-linked roles

VPC Lattice supports using service-linked roles in all of the Regions where the service is available.

AWS managed policies for Amazon VPC Lattice

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: VPCLatticeFullAccess

This policy provides full access to Amazon VPC Lattice and limited access to other dependent services. It includes permissions to do the following:

- ACM – Retrieve the SSL/TLS certificate ARN for custom domain names.
- CloudWatch – View access logs and monitoring data.
- CloudWatch Logs – Set up and send access logs to CloudWatch Logs.
- Amazon EC2 – Configure network interfaces and retrieve information about EC2 instances and VPCs. This is used to create resource configurations, resource gateways, and target groups, configure VPC Lattice entity associations, and register targets.
- Elastic Load Balancing – Retrieve information about an Application Load Balancer to register it as a target.
- Firehose – Retrieve information about delivery streams used to store access logs.
- Lambda – Retrieve information about a Lambda function to register it as a target.
- Amazon RDS – Retrieve information about RDS clusters and instances.
- Amazon S3 – Retrieve information about S3 buckets used to store access logs.

To view the permissions for this policy, see VPCLatticeFullAccess in the *AWS Managed Policy Reference*.

To use other AWS services that VPC Lattice is integrated with and the entire suite of VPC Lattice features, you must have specific additional permissions. These permissions are not included in the `VPCLatticeFullAccess` managed policy because of the confused deputy privilege escalation risk. For more information, see Additional required permissions for full access.

AWS managed policy: VPCLatticeReadOnlyAccess

This policy provides read-only access to Amazon VPC Lattice and limited access to other dependent services. It includes permissions to do the following:

- ACM – Retrieve the SSL/TLS certificate ARN for custom domain names.
- CloudWatch – View access logs and monitoring data.
- CloudWatch Logs – View log delivery information for access log subscriptions.
- Amazon EC2 – Retrieve information about EC2 instances and VPCs to create target groups and register targets.
- Elastic Load Balancing – Retrieve information about an Application Load Balancer.
- Firehose – Retrieve information about delivery streams for access log delivery.
- Lambda – View information about a Lambda function.
- Amazon RDS – Retrieve information about RDS clusters and instances.

- Amazon S3 – Retrieve information about S3 buckets for access log delivery.

To view the permissions for this policy, see VPCLatticeReadOnlyAccess in the *AWS Managed Policy Reference*.

AWS managed policy: VPCLatticeServicesInvokeAccess

This policy provides access to invoke Amazon VPC Lattice services.

To view the permissions for this policy, see VPCLatticeServicesInvokeAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AWSVpcLatticeServiceRolePolicy

This policy is attached to a service-linked role named **AWSServiceRoleForVpcLattice** to allow VPC Lattice to perform actions on your behalf. You can't attach this policy to your IAM entities. For more information, see Using service-linked roles for Amazon VPC Lattice.

To view the permissions for this policy, see AWSVpcLatticeServiceRolePolicy in the *AWS Managed Policy Reference*.

VPC Lattice updates to AWS managed policies

View details about updates to AWS managed policies for VPC Lattice since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed for the VPC Lattice User Guide.

Change	Description	Date
VPCLatticeFullAccess	VPC Lattice adds read-only permissions to describe Amazon RDS clusters and instances.	December 1, 2024
VPCLatticeReadOnlyAccess	VPC Lattice adds read-only permissions to describe Amazon RDS clusters and instances.	December 1, 2024
AWSVpcLatticeServiceRolePolicy	VPC Lattice adds permissions to allow VPC Lattice to create	December 1, 2024

Change	Description	Date
	a requester-managed network interface.	
VPCLatticeFullAccess	VPC Lattice adds a new policy to grant permissions for full access to Amazon VPC Lattice and limited access to other dependent services.	March 31, 2023
VPCLatticeReadOnlyAccess	VPC Lattice adds a new policy to grant permissions for read-only access to Amazon VPC Lattice and limited access to other dependent services.	March 31, 2023
VPCLatticeServicesInvokeAccess	VPC Lattice adds a new policy to grant access to invoke Amazon VPC Lattice services.	March 31, 2023
AWSVpcLatticeServiceRolePolicy	VPC Lattice adds permissions to its service-linked role to allow VPC Lattice to publish CloudWatch metrics in the `AWS/VpcLattice` namespace. The `AWSVpcLatticeServiceRolePolicy` policy includes permission to call the CloudWatch PutMetricData API action. For more information, see Using service-linked roles for Amazon VPC Lattice.	December 5, 2022
VPC Lattice started tracking changes	VPC Lattice started tracking changes for its AWS managed policies.	December 5, 2022

Compliance validation for Amazon VPC Lattice

Third-party auditors assess the security and compliance of Amazon VPC Lattice as part of multiple AWS compliance programs.

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious

activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.

- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Access Amazon VPC Lattice using interface endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and Amazon VPC Lattice by creating an *interface VPC endpoint*. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access VPC Lattice APIs without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with VPC Lattice APIs.

Each interface endpoint is represented by one or more network interfaces in your subnets.

Considerations for interface VPC endpoints

Before you set up an interface VPC endpoint for VPC Lattice, ensure that you review Access AWS services through AWS PrivateLink in the *AWS PrivateLink Guide*.

VPC Lattice supports making calls to all of its API actions from your VPC.

Creating an interface VPC endpoint for VPC Lattice

You can create a VPC endpoint for the VPC Lattice service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Create an interface VPC endpoint in the *AWS PrivateLink Guide*.

Create a VPC endpoint for VPC Lattice using the following service name:

`com.amazonaws.region.vpc-lattice`

If you enable private DNS for the endpoint, you can make API requests to VPC Lattice using its default DNS name for the Region, for example, `vpc-lattice.us-east-1.amazonaws.com`.

Resilience in Amazon VPC Lattice

The AWS global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure security in Amazon VPC Lattice

As a managed service, Amazon VPC Lattice is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access VPC Lattice through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

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

Monitoring Amazon VPC Lattice

Use the features in this section to monitor your Amazon VPC Lattice service networks, services, target groups, and VPC connections.

Contents

CloudWatch metrics for Amazon VPC Lattice

Amazon VPC Lattice sends data related to your target groups and services to Amazon CloudWatch, and processes it into readable, near real-time metrics. These metrics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds and send notifications or take actions when those thresholds are met. For more information, see the Amazon CloudWatch User Guide.

Amazon VPC Lattice uses a service-linked role in your AWS account to send metrics to Amazon CloudWatch. For more information, see Using service-linked roles for Amazon VPC Lattice.

Contents

View Amazon CloudWatch metrics

You can view the Amazon CloudWatch metrics for your target groups and services using the CloudWatch console or AWS CLI.

To view metrics using the CloudWatch console

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.

3. Select the `AWS/VpcLattice` namespace.
4. (Optional) To view a metric across all dimensions, enter its name in the search field.
5. (Optional) To filter by dimension, select one of the following:

   - To display only the metrics reported for your target groups, choose **Target groups**. To view the metrics for a single target group, enter its name in the search field.
   - To display only the metrics reported for your services, choose **Services**. To view the metrics for a single service, enter its name in the search field.

To view metrics using the AWS CLI

Use the following CloudWatch list-metrics AWS CLI command to list the available metrics:

```
aws cloudwatch list-metrics --namespace AWS/VpcLattice
```

For information about each of the metrics and their dimensions, see Target group metrics and Service metrics.

Target group metrics

VPC Lattice automatically stores metrics related to target groups in the `AWS/VpcLattice` Amazon CloudWatch namespace. For more information about target groups, see Target groups in VPC Lattice.

Dimensions

To filter the metrics for target groups, use the following dimensions:

- `AvailabilityZone`
- `TargetGroup`

Metric	Description
`TotalConnectionCount`	Total connections. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic.

Metric	Description
	Reporting frequency • Once a minute. **Statistics** • The most useful statistic is Sum.
ActiveConnectionCount	Active connections. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.
ConnectionErrorCount	Total connection failures. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.

Metric	Description
HTTP1_ConnectionCount	Total HTTP/1.1 connections. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.
HTTP2_ConnectionCount	Total HTTP/2 connections. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.

Metric	Description
ConnectionTimeoutCount	Total connection connect timeouts. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.
TotalReceivedConnectionBytes	Total received connection bytes. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.

Metric	Description
TotalSentConnectionBytes	Total sent connection bytes. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.
TotalRequestCount	Total requests. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.

Metric	Description
`ActiveRequestCount`	Total active requests. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is `Sum`.
`RequestTime`	Request time to the last byte in milliseconds. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistics are `Average` and `pNN.NN` (percentiles).

Metric	Description
`HTTPCode_2XX_Count`, `HTTPCode_3XX_Count`, `HTTPCode_4XX_Count`, `HTTPCode_5XX_Count`	Aggregate HTTP response codes. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is `Sum`.
`TLSConnectionErrorCount`	Total TLS connection errors not including failed certificate verifications. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is `Sum`.

Metric	Description
`TotalTLSConnection HandshakeCount`	Total successful TLS connection handshakes. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is `Sum`.

Service metrics

VPC Lattice automatically stores metrics related to services in the `AWS/VpcLattice` Amazon CloudWatch namespace. For more information about services, see Services in VPC Lattice.

Dimensions

To filter the metrics for target groups, use the following dimensions:

- `AvailabilityZone`
- `Service`

Metric	Description
`RequestTimeoutCount`	Total requests that timed out waiting for a response. **Reporting criteria** • Always reported (whether it's a zero or nonzero value) from the time the resource receives traffic.

Metric	Description
	Reporting frequency • Once a minute. **Statistics** • The most useful statistic is Sum.
TotalRequestCount	Total requests. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.

Metric	Description
RequestTime	Request time in milliseconds. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistics are Average and pNN.NN (percentiles).
HTTPCode_2XX_Count, HTTPCode_3XX_Count, HTTPCode_4XX_Count, HTTPCode_5XX_Count	Aggregate HTTP response codes. **Reporting criteria** • Always reported (whether it's a zero or non-zero value) from the time the resource receives traffic. **Reporting frequency** • Once a minute. **Statistics** • The most useful statistic is Sum.

Access logs for Amazon VPC Lattice

Access logs capture detailed information about your VPC Lattice services and resource configurations. You can use these access logs to analyze traffic patterns and audit all of the services

in the network. For VPC Lattice services, we publish `VpcLatticeAccessLogs` and for resource configurations, we publish `VpcLatticeResourceAccessLogs` which needs to be configured separately.

Access logs are optional and are disabled by default. After you enable access logs, you can disable them at any time.

Pricing

Charges apply when access logs are published. Logs that AWS natively publishes on your behalf are called *vended logs*. For more information about pricing for vended logs, see Amazon CloudWatch Pricing, choose **Logs**, and view the pricing under **Vended Logs**.

Contents

IAM permissions required to enable access logs

To enable access logs and send the logs to their destinations, you must have the following actions in the policy attached to the IAM user, group, or role that you are using.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "ManageVPCLatticeAccessLogSetup",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
```

```
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "vpc-lattice:CreateAccessLogSubscription",
                "vpc-lattice:GetAccessLogSubscription",
                "vpc-lattice:UpdateAccessLogSubscription",
                "vpc-lattice:DeleteAccessLogSubscription",
                "vpc-lattice:ListAccessLogSubscriptions"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

For more information, see Adding and removing IAM identity permissions in the *AWS Identity and Access Management User Guide*.

After you've updated the policy attached to the IAM user, group, or role that you are using, go to Enable access logs.

Access log destinations

You can send access logs to the following destinations.

Amazon CloudWatch Logs

- VPC Lattice typically delivers logs to CloudWatch Logs within 2 minutes. However, keep in mind that actual log delivery time is on a best effort basis and there may be additional latency.
- A resource policy is created automatically and added to the CloudWatch log group if the log group does not have certain permissions. For more information, see Logs sent to CloudWatch Logs in the *Amazon CloudWatch User Guide*.
- You can find access logs that are sent to CloudWatch under Log Groups in the CloudWatch console. For more information, see View log data sent to CloudWatch Logs in the *Amazon CloudWatch User Guide*.

Amazon S3

- VPC Lattice typically delivers logs to Amazon S3 within 6 minutes. However, keep in mind that actual log delivery time is on a best effort basis and there may be additional latency.
- A bucket policy will be created automatically and added to your Amazon S3 bucket if the bucket does not have certain permissions. For more information, see Logs sent to Amazon S3 in the *Amazon CloudWatch User Guide*.
- Access logs that are sent to Amazon S3 use the following naming convention:

```
[bucket]/[prefix]/AWSLogs/[accountId]/VpcLattice/AccessLogs/[region]/[YYYY/
MM/DD]/[resource-id]/[accountId]_VpcLatticeAccessLogs_[region]_[resource-
id]_YYYYMMDDTHHmmZ_[hash].json.gz
```

- VpcLatticeResourceAccessLogs that are sent to Amazon S3 use the following naming convention:

```
[bucket]/[prefix]/AWSLogs/[accountId]/VpcLattice/ResourceAccessLogs/[region]/[YYYY/
MM/DD]/[resource-id]/[accountId]_VpcLatticeResourceAccessLogs_[region]_[resource-
id]_YYYYMMDDTHHmmZ_[hash].json.gz
```

Amazon Data Firehose

- VPC Lattice typically delivers logs to Firehose within 2 minutes. However, keep in mind that actual log delivery time is on a best effort basis and there may be additional latency.
- A service-linked role is automatically created that grants VPC Lattice permission to send access logs to Amazon Data Firehose. For automatic role creation to succeed, users must have permission for the `iam:CreateServiceLinkedRole` action. For more information, see Logs sent to Amazon Data Firehose in the *Amazon CloudWatch User Guide*.
- For more information about viewing the logs sent to Amazon Data Firehose, see Monitoring Amazon Kinesis Data Streams in the *Amazon Data Firehose Developer Guide*.

Enable access logs

Complete the following procedure to configure access logs to capture and deliver access logs to the destination that you choose.

Contents

- Enable access logs using the AWS CLI

Enable access logs using the console

You can enable access logs for a service network, a service, or a resource configuration during creation. You can also enable access logs after you create a service network, service, or resource configuration as described in the following procedure.

To create a basic service using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Select the service network, service, or resource configuration.
3. Choose **Actions**, **Edit log settings**.
4. Turn on the **Access logs** toggle switch.
5. Add a delivery destination for your access logs as follows:
   - Select **CloudWatch Log group** and choose a log group. To create a log group, choose **Create a log group in CloudWatch**.
   - Select **S3 bucket** and enter the S3 bucket path, including any prefix. To search your S3 buckets, choose **Browse S3**.
   - Select **Kinesis Data Firehose delivery stream** and choose a delivery stream. To create a delivery stream, choose **Create a delivery stream in Kinesis**.
6. Choose **Save changes**.

Enable access logs using the AWS CLI

Use the CLI command create-access-log-subscription to enable access logs for service networks or services.

Access log contents

The following table describes the fields of an access log entry.

Field	Description	Format
`hostHeader`	The authority header of the request.	string

Field	Description	Format
`sslCipher`	The OpenSSL name for the set of ciphers used to establish the client TLS connection.	string
`serviceNetworkArn`	The service network ARN.	arn:aws:vpc-lattice:*region*:*account*:servicenetwork/*id*
`resolvedUser`	The ARN of the user when authentication is enabled and authentication is done.	null | ARN | "Anonymous" | "Unknown"
`authDeniedReason`	The reason that access is denied when authentication is enabled.	null | "Service" | "Network" | "Identity"
`requestMethod`	The method header of the request.	string
`targetGroupArn`	The target host group to which the target host belongs.	string
`tlsVersion`	The TLS version.	TLSv*x*
`userAgent`	The user-agent header.	string
`ServerNameIndication`	[HTTPS only] The value set on ssl connection socket for Server Name Indication (SNI).	string
`destinationVpcId`	The destination VPC ID.	vpc-*xxxxxxxx*
`sourceIpPort`	The IP address and :port of the source.	*ip*:*port*

Field	Description	Format
`targetIpPort`	The IP address and port of the target.	*ip*:*port*
`serviceArn`	The service ARN.	arn:aws:vpc-lattice:*region*:*account*:service/*id*
`sourceVpcId`	The source VPC ID.	vpc-*xxxxxxxx*
`requestPath`	The path of the request.	LatticePath?:*path*
`startTime`	The request start time.	*YYYY*-*MM*-*DD*T*HH*:*MM*:*SS*Z
`protocol`	The protocol. Currently either HTTP/1.1 or HTTP/2.	string
`responseCode`	The HTTP response code. Only the response code for the final headers are logged. For more information, see Troubleshoot access logs.	integer
`bytesReceived`	The body and header bytes received.	integer
`bytesSent`	The body and header bytes sent.	integer
`duration`	Total duration in milliseconds of the request from the start time to the last byte out.	integer
`requestToTargetDuration`	Total duration in milliseconds of the request from the start time to the last byte sent to the target.	integer

Field	Description	Format
`responseFromTargetDuration`	Total duration in milliseconds of the request from the first byte read from the target host to the last byte sent to the client.	integer
`grpcResponseCode`	The gRPC response code. For more information, see Status codes and their use in gRPC. This field is logged only if the service supports gRPC.	integer
`callerPrincipal`	The authenticated principal.	string
`callerX509SubjectCN`	The subject name (CN).	string
`callerX509IssuerOU`	The issuer (OU).	string
`callerX509SANNameCN`	The issuer alternative (Name/CN).	string
`callerX509SANDNS`	The subject alternative name (DNS).	string
`callerX509SANURI`	The subject alternative name (URI).	string
`sourceVpcArn`	The ARN of the VPC where the request originated.	arn:aws:ec2:*region*:*account*:vpc/*id*

Example

The following is an example log entry.

```
{
    "hostHeader": "example.com",
    "sslCipher": "-",
```

```
    "serviceNetworkArn": "arn:aws:vpc-lattice:us-west-2:123456789012:servicenetwork/
svn-1a2b3c4d",
    "resolvedUser": "Unknown",
    "authDeniedReason": "null",
    "requestMethod": "GET",
    "targetGroupArn": "arn:aws:vpc-lattice:us-west-2:123456789012:targetgroup/
tg-1a2b3c4d",
    "tlsVersion": "-",
    "userAgent": "-",
    "serverNameIndication": "-",
    "destinationVpcId": "vpc-0abcdef1234567890",
    "sourceIpPort": "178.0.181.150:80",
    "targetIpPort": "131.31.44.176:80",
    "serviceArn": "arn:aws:vpc-lattice:us-west-2:123456789012:service/svc-1a2b3c4d",
    "sourceVpcId": "vpc-0abcdef1234567890",
    "requestPath": "/billing",
    "startTime": "2023-07-28T20:48:45Z",
    "protocol": "HTTP/1.1",
    "responseCode": 200,
    "bytesReceived": 42,
    "bytesSent": 42,
    "duration": 375,
    "requestToTargetDuration": 1,
    "responseFromTargetDuration": 1,
    "grpcResponseCode": 1
}
```


Resource access log contents

The following table describes the fields of a resource access log entry.

Field	Description	Format
`serviceNetworkArn`	The service network ARN.	arn:*partition* vpc-latti ce:*region*:*account*:servicen etwork/*id*
`serviceNetworkReso urceAssociationId`	The service network resource ID.	*snra-xxx*

Field	Description	Format
`vpcEndpointId`	The endpoint ID that was used to access the resource.	string
`sourceVpcArn`	The source VPC ARN or the VPC from where connection was initiated.	string
`resourceConfigurationArn`	The ARN of the resource configuration that was accessed.	string
`protocol`	The protocol used to communicate to the resource configuration. Currently only tcp is supported.	string
`sourceIpPort`	The IP address and port of the source which initiated the connection.	*ip*:*port*
`destinationIpPort`	The IP address and port against which the connection was initiated. This will be the IP of SN-E/SN-A.	*ip*:*port*
`gatewayIpPort`	The IP address and port used by the resource gateway to access the resource.	*ip*:*port*
`resourceIpPort`	The IP address and port of the resource.	*ip*:*port*

Example

The following is an example log entry.

```
{
```

```
    "eventTimestamp": "2024-12-02T10:10:10.123Z",
    "serviceNetworkArn": "arn:aws:vpc-lattice:us-west-2:1234567890:servicenetwork/
sn-1a2b3c4d",
    "serviceNetworkResourceAssociationId": "snra-1a2b3c4d",
    "vpcEndpointId": "vpce-01a2b3c4d",
    "sourceVpcArn": "arn:aws:ec2:us-west-2:1234567890:vpc/vpc-01a2b3c4d",
    "resourceConfigurationArn": "arn:aws:vpc-lattice:us-
west-2:0987654321:resourceconfiguration/rcfg-01a2b3c4d",
    "protocol": "tcp",
    "sourceIpPort": "172.31.23.56:44076",
    "destinationIpPort": "172.31.31.226:80",
    "gatewayIpPort": "10.0.28.57:49288",
    "resourceIpPort": "10.0.18.190:80"
}
```

Troubleshoot access logs

This section contains an explanation of the HTTP error codes that you may see in access logs.

Error code	Possible causes
HTTP 400: Bad Request	• The client sent a malformed request that doesn't meet the HTTP specification. • The request header exceeded 60K for the entire request header or more than 100 headers. • The client closed the connection before sending the full request body.
HTTP 403: Forbidden	Authentication has been configured for the service, but the incoming request is not authenticated or authorized.
HTTP 404: Non Existent Service	You're trying to connect to a service that does not exist or is not registered to the right service network.
HTTP 500: Internal Server Error	VPC Lattice has encountered an error, such as failure to connect to targets.
HTTP 502: Bad Gateway	VPC Lattice has encountered an error.

CloudTrail logs for Amazon VPC Lattice

Amazon VPC Lattice is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for VPC Lattice as events. The calls captured include calls from the VPC Lattice console and code calls to the VPC Lattice API operations. Using the information collected by CloudTrail, you can determine the request that was made to VPC Lattice, the IP address from which the request was made, when it was made, and additional details.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

CloudTrail is active in your AWS account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management events in an AWS Region. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your AWS account past 90 days, create a trail or a CloudTrail Lake event data store.

CloudTrail trails

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the AWS Management Console are multi-Region. You can create a single-Region or a multi-Region trail by using the AWS CLI. Creating a multi-Region trail is recommended because you capture activity in all AWS Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's AWS Region. For more information about trails, see Creating a trail for your AWS account and Creating a trail for an organization in the *AWS CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For

more information about CloudTrail pricing, see AWS CloudTrail Pricing. For information about Amazon S3 pricing, see Amazon S3 Pricing.

CloudTrail Lake event data stores

CloudTrail Lake lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to Apache ORC format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying advanced event selectors. The selectors that you apply to an event data store control which events persist and are available for you to query. For more information about CloudTrail Lake, see Working with AWS CloudTrail Lake in the *AWS CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the pricing option you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

To monitor additional actions, use access logs. For more information, see Access logs.

VPC Lattice management events in CloudTrail

Management events provide information about management operations that are performed on resources in your AWS account. These are also known as control plane operations. By default, CloudTrail logs management events.

Amazon VPC Lattice logs VPC Lattice control plane operations as management events. For a list of the Amazon VPC Lattice control plane operations that VPC Lattice logs to CloudTrail, see the Amazon VPC Lattice API Reference.

VPC Lattice event examples

An event represents a single request from any source and includes information about the requested API operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so events don't appear in any specific order.

The following example shows a CloudTrail event for the CreateService operation.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "abcdef01234567890",
    "arn": "arn:abcdef01234567890",
    "accountId": "abcdef01234567890",
    "accessKeyId": "abcdef01234567890",
    "sessionContext": {
        "sessionIssuer": {
            "type": "Role",
            "principalId": "abcdef01234567890",
            "arn": "arn:abcdef01234567890",
            "accountId": "abcdef01234567890",
            "userName": "abcdef01234567890"
        },
        "webIdFederationData": {},
        "attributes": {
            "creationDate": "2022-08-16T03:34:54Z",
            "mfaAuthenticated": "false"
        }
    }
  },
  "eventTime": "2022-08-16T03:36:12Z",
  "eventSource": "vpc-lattice.amazonaws.com",
  "eventName": "CreateService",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "abcdef01234567890",
  "userAgent": "abcdef01234567890",
  "requestParameters": {
    "name": "rates-service"
  },
  "responseElements": {
    "name": "rates-service",
    "id": "abcdef01234567890",
    "arn": "arn:abcdef01234567890",
    "status": "CREATE_IN_PROGRESS"
  },
  "requestID": "abcdef01234567890",
  "eventID": "abcdef01234567890",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
```

```
  "recipientAccountId": "abcdef01234567890",
  "eventCategory": "Management"
}
```

The following example shows a CloudTrail event for the DeleteService operation.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "abcdef01234567890",
    "arn": "arn:ABCXYZ123456",
    "accountId": "abcdef01234567890",
    "accessKeyId": "abcdef01234567890",
    "sessionContext": {
        "sessionIssuer": {
            "type": "Role",
            "principalId": "abcdef01234567890",
            "arn": "arn:aws:iam::AIDACKCEVSQ6C2EXAMPLE:role/Admin",
            "accountId": "abcdef01234567890",
            "userName": "Admin"
        },
        "webIdFederationData": {},
        "attributes": {
            "creationDate": "2022-10-27T17:42:36Z",
            "mfaAuthenticated": "false"
        }
    }
  },
  "eventTime": "2022-10-27T17:56:41Z",
  "eventSource": "vpc-lattice.amazonaws.com",
  "eventName": "DeleteService",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "72.21.198.64",
  "userAgent": "abcdef01234567890",
  "requestParameters": {
    "serviceIdentifier": "abcdef01234567890"
  },
  "responseElements": {
    "name": "test",
    "id": "abcdef01234567890",
    "arn": "arn:abcdef01234567890",
    "status": "DELETE_IN_PROGRESS"
```

```
  },
  "requestID": "abcdef01234567890",
  "eventID": "abcdef01234567890",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "abcdef01234567890",
  "eventCategory": "Management"
}
```

For information about CloudTrail record contents, see CloudTrail record contents in the *AWS CloudTrail User Guide*.

Quotas for Amazon VPC Lattice

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas can't be increased.

To view the quotas for VPC Lattice, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **VPC Lattice**.

To request a quota increase, see Requesting a quota increase in the *Service Quotas User Guide*.

Your AWS account has the following quotas related to VPC Lattice.

Name	Default	Adjustable	Description
Auth policy size	Each supported Region: 10 Kilobytes	No	The maximum size of a JSON file in an Auth policy.
Child Resource Configurations per Group Resource Configuration	Each supported Region: 60	Yes	The maximum number of child resource configurations in a group resource configuration. For additional capacity and limit increases, contact AWS Support.
Listeners per service	Each supported Region: 2	Yes	The maximum number of listeners that you can create for a service. For additional capacity and limit increases, contact AWS Support.
Resource Configurations per service network	Each supported Region: 500	Yes	The maximum number of resource configurations associated with

Name	Default	Adjustable	Description
			a service network. For additional capacity and limit increases, contact AWS Support.
Resource configurations per AWS Region	Each supported Region: 2,000	Yes	The maximum number of resource configurations an AWS account can have per AWS Region. For additional capacity and limit increases, contact AWS Support.
Resource gateways per VPC	Each supported Region: 500	Yes	The maximum number of resource gateways in a VPC. For additional capacity and limit increases, contact AWS Support.
Rules per listener	Each supported Region: 10	Yes	The maximum number of rules that you can define for your service listener. For additional capacity and limit increases, contact AWS Support.
Security groups per association	Each supported Region: 5	No	The maximum number of security groups that you can add to an association between a VPC and a service network.

Name	Default	Adjustable	Description
Service associations per service network	Each supported Region: 500	Yes	The maximum number of services that you can associate with a single service network. For additional capacity and limit increases, contact AWS Support.
Service networks per region	Each supported Region: 50	Yes	The maximum number of service networks per region. For additional capacity and limit increases, contact AWS Support.
Services per region	Each supported Region: 2,000	Yes	The maximum number of services per region. For additional capacity and limit increases, contact AWS Support.
Target groups per region	Each supported Region: 500	Yes	The maximum number of target groups per region. For additional capacity and limit increases, contact AWS Support.
Target groups per service	Each supported Region: 10	Yes	The maximum number of target groups that you can associate with a service. For additional capacity and limit increases, contact AWS Support.

Name	Default	Adjustable	Description
Targets per target group	Each supported Region: 1,000	Yes	The maximum number of targets that you can associate with a single target group. For additional capacity and limit increases, contact AWS Support.
VPC associations per service network	Each supported Region: 500	Yes	The maximum number of VPCs that you can associate with a single service network. For additional capacity and limit increases, contact AWS Support.
VPC endpoints of type service network per service network	Each supported Region: 200	Yes	The maximum number of service network endpoints associated with a service network. For additional capacity and limit increases, contact AWS Support.

The following Availability Zones aren't supported for VPC Lattice: `use1-az3`, `usw1-az2`, `apne1-az3`, `apne2-az2`, `euc1-az2`, `euw1-az4`, `cac1-az3`, `ilc1-az2`.

The following limits also apply.

Limit	Value	Description
Bandwidth per service per Availability Zone	10 Gbps	The maximum bandwidth allocated per service per Availability Zone.

Limit	Value	Description
Bandwidth per resource gateway per Availability Zone	100 Gbps	The maximum bandwidth allocated per resource gateway per Availability Zone.
Maximum transmission unit (MTU) per connection	8500 bytes	The size of the largest data packet that a service can accept.
Requests per second per service per Availability Zone	10,000	For HTTP services, this is the maximum number of requests per second per service per Availability Zone.
Connection idle time per connection	1 minute	The maximum time that a connection can sit idle with no active requests (for HTTP and GRPC), or with no active data transfer (for TLS-PASSTHROUGH).
Maximum connection lifetime per connection	10 minutes	The maximum time that a connection can be open.
Service network per VPC	1 service network	You can connect a VPC to only one service network through an association. To connect a VPC to multiple service networks, you can use VPC endpoints of type service network.

Document history for the Amazon VPC Lattice User Guide

The following table describes the documentation releases for VPC Lattice.

Change	Description	Date
Added VPC Lattice for Oracle Database@AWS	VPC Lattice for Oracle Database@AWS released.	June 26, 2025
Added dual-stack support for management endpoints	VPC Lattice now supports dual-stack (IPv4 and IPv6) endpoints for all VPC Lattice management APIs.	April 30, 2025
Share and access resources	VPC Lattice now supports sharing and accessing resources across VPC and account boundaries. This includes updates to the VPCLatticeReadOnlyAccess and VPCLatticeFullAccess policies.	December 1, 2024
TLS passthrough	VPC Lattice now supports TLS passthrough, which allows you to perform TLS terminati on in your application for end-to-end authentication.	May 14, 2024
Lambda event structure version	VPC Lattice now supports a new version of the Lambda event structure.	September 7, 2023
Support for shared VPCs	Participants can create VPC Lattice target groups in a shared VPC.	July 5, 2023

General Availability release	The release of the VPC Lattice User Guide for General Availability (GA)	March 31, 2023
VPC Lattice now reports changes to its AWS managed policies	Changes to managed policies are reported in "AWS managed policies for VPC Lattice" in the "Security" chapter.	March 29, 2023
Support for Application Load Balancer target type	VPC Lattice now supports creating an Application Load Balancer type target group.	March 29, 2023
Support for all instance types	VPC Lattice now supports all instance types.	March 27, 2023
IPv6 support	VPC Lattice now supports both IPv4 and IPv6 IP target groups.	March 27, 2023
HTTP2 protocol version for health checks	Health checks are now supported when the target group protocol version is HTTP2.	March 27, 2023
Fixed response action for listener rules	Listeners for VPC Lattice services now support fixed response actions in addition to forward actions.	March 27, 2023
Support for custom domain names	You can now configure a custom domain name for your VPC Lattice service	February 14, 2023

Support for BYOC (Bring Your Own Certificate)	VPC Lattice supports using your own an SSL/TLS certificate in ACM for custom domain names.	February 14, 2023
VPC Lattice now reports an updated list of unsupported instance types	Three additional instances have been added to the unsupported list of instances.	January 26, 2023
VPC Lattice now reports changes to its AWS managed policies	Beginning December 5, 2022, changes to managed policies are reported in the topic "AWS managed policies for VPC Lattice" in the "Security " chapter. The first change listed is the addition of permissions needed for CloudWatch monitoring.	December 5, 2022
Initial release	Initial release of the VPC Lattice User Guide	December 5, 2022