VPC Peering

Amazon Virtual Private Cloud

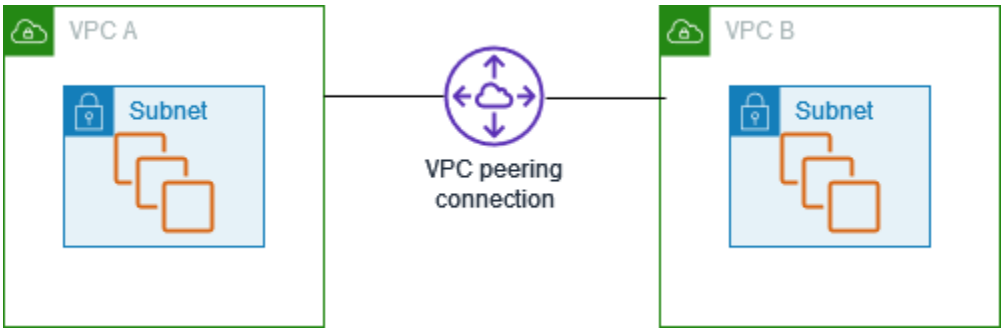
Amazon Virtual Private Cloud: VPC Peering

Table of Contents

What is VPC peering?

A *virtual private cloud* (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud. You can launch AWS resources, such as Amazon EC2 instances, into your VPC.

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account. The VPCs can be in different Regions (also known as an inter-Region VPC peering connection).



AWS uses the existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor a VPN connection, and does not rely on a separate piece of physical hardware. There is no single point of failure for communication or a bandwidth bottleneck.

A VPC peering connection helps you to facilitate the transfer of data. For example, if you have more than one AWS account, you can peer the VPCs across those accounts to create a file sharing network. You can also use a VPC peering connection to allow other VPCs to access resources you have in one of your VPCs.

When you establish peering relationships between VPCs across different AWS Regions, resources in the VPCs (for example, EC2 instances and Lambda functions) in different AWS Regions can communicate with each other using private IP addresses, without using a gateway, VPN connection, or network appliance. The traffic remains in the private IP address space. All inter-Region traffic is encrypted before leaving AWS facilities with no single point of failure, or bandwidth bottleneck. Traffic always stays on the global AWS backbone, and never traverses the public internet, which reduces threats, such as common exploits, and DDoS attacks. Inter-Region VPC peering provides a simple and cost-effective way to share resources between Regions or replicate data for geographic redundancy.

Pricing for a VPC peering connection

There is no charge to create a VPC peering connection. All data transfer over a VPC peering connection that stays within an Availability Zone is free, even if it's between different accounts. Charges apply for data transfer over VPC peering connections that cross Availability Zones and Regions. For more information, see Amazon EC2 Pricing.

How VPC peering connections work

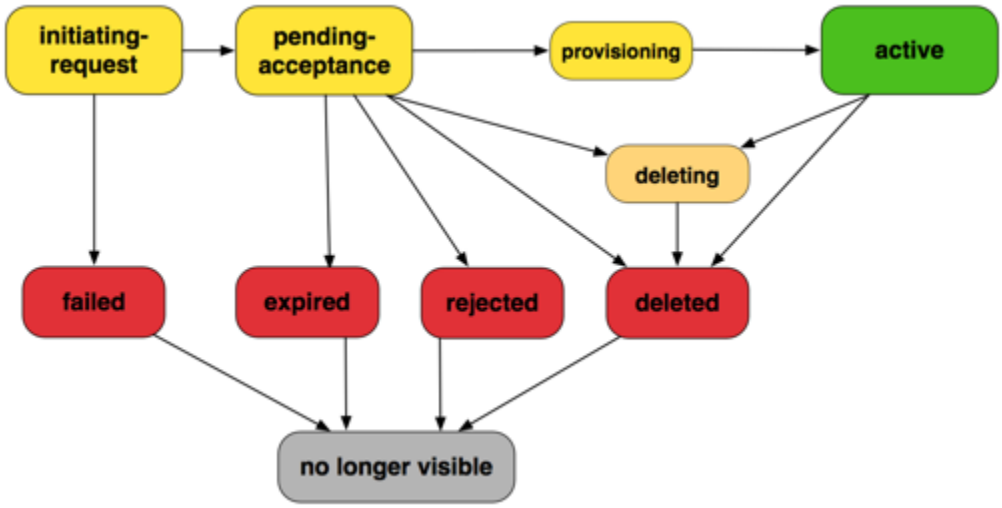
The following steps describe the VPC peering process:

1. The owner of the *requester VPC* sends a request to the owner of the *accepter VPC* to create the VPC peering connection. The accepter VPC can be owned by you, or another AWS account, and cannot have a CIDR block that overlaps with the CIDR block of the requester VPC.
2. The owner of the *accepter* VPC accepts the VPC peering connection request to activate the VPC peering connection.
3. To enable the flow of traffic between the VPCs using private IP addresses, the owner of each VPC in the VPC peering connection must manually add a route to one or more of their VPC route tables that points to the IP address range of the other VPC (the peer VPC).
4. If required, update the security group rules that are associated with your EC2 instance to ensure that traffic to and from the peer VPC is not restricted. If both VPCs are in the same Region, you can reference a security group from the peer VPC as a source or destination for inbound or outbound rules in your security group.
5. With the default VPC peering connection options, if EC2 instances on either side of a VPC peering connection address each other using a public DNS hostname, the hostname resolves to the public IP address of the EC2 instance. To change this behavior, enable DNS hostname resolution for your VPC connection. After enabling DNS hostname resolution, if EC2 instances on either side of the VPC peering connection address each other using a public DNS hostname, the hostname resolves to the private IP address of the EC2 instance.

For more information, see VPC peering connections.

VPC peering connection lifecycle

A VPC peering connection goes through various stages starting from when the request is initiated. At each stage, there may be actions that you can take, and at the end of its lifecycle, the VPC peering connection remains visible in the Amazon VPC console and API or command line output for a period of time.



- **Initiating-request**: A request for a VPC peering connection has been initiated. At this stage, the peering connection can fail, or can go to `pending-acceptance`.
- **Failed**: The request for the VPC peering connection has failed. While in this state, it cannot be accepted, rejected, or deleted. The failed VPC peering connection remains visible to the requester for 2 hours.
- **Pending-acceptance**: The VPC peering connection request is awaiting acceptance from the owner of the accepter VPC. During this state, the owner of the requester VPC can delete the request, and the owner of the accepter VPC can accept or reject the request. If no action is taken on the request, it expires after 7 days.
- **Expired**: The VPC peering connection request has expired, and no action can be taken on it by either VPC owner. The expired VPC peering connection remains visible to both VPC owners for 2 days.
- **Rejected**: The owner of the accepter VPC has rejected a `pending-acceptance` VPC peering connection request. While in this state, the request cannot be accepted. The rejected VPC peering connection remains visible to the owner of the requester VPC for 2 days, and visible to the owner of the accepter VPC for 2 hours. If the request was created within the same AWS account, the rejected request remains visible for 2 hours.
- **Provisioning**: The VPC peering connection request has been accepted, and will soon be in the `active` state.
- **Active**: The VPC peering connection is active, and traffic can flow between the VPCs (provided that your security groups and route tables allow the flow of traffic). While in this state, either of the VPC owners can delete the VPC peering connection, but cannot reject it.

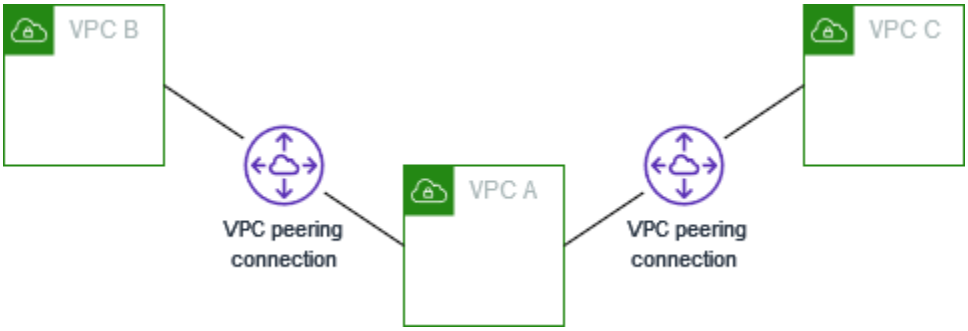
> **Note**
>
> If an event in a Region in which a VPC resides prevents the flow of traffic, the status of the VPC peering connection remains `Active`.

- **Deleting**: Applies to an inter-Region VPC peering connection that is in the process of being deleted. The owner of either VPC has submitted a request to delete an `active` VPC peering connection, or the owner of the requester VPC has submitted a request to delete a `pending-acceptance` VPC peering connection request.
- **Deleted**: An `active` VPC peering connection has been deleted by either of the VPC owners, or a `pending-acceptance` VPC peering connection request has been deleted by the owner of the requester VPC. While in this state, the VPC peering connection cannot be accepted or rejected. The VPC peering connection remains visible to the party that deleted it for 2 hours, and visible to the other party for 2 days. If the VPC peering connection was created within the same AWS account, the deleted request remains visible for 2 hours.

Multiple VPC peering connections

A VPC peering connection is a one to one relationship between two VPCs. You can create multiple VPC peering connections for each VPC that you own, but transitive peering relationships are not supported. You do not have any peering relationship with VPCs that your VPC is not directly peered with.

The following diagram is an example of one VPC peered to two different VPCs. There are two VPC peering connections: VPC A is peered with both VPC B and VPC C. VPC B and VPC C are not peered, and you cannot use VPC A as a transit point for peering between VPC B and VPC C. If you want to enable routing of traffic between VPC B and VPC C, you must create a unique VPC peering connection between them.



VPC peering limitations

Consider the following limitations for VPC peering connections. In some cases, you can use a transit gateway attachment instead of a VPC peering connection. For more information, see Example transit gateway scenarios in *Amazon VPC Transit Gateways*.

Connections

- There is a quota on the number of active and pending VPC peering connections per VPC. For more information, see *Quotas*.
- You cannot have more than one VPC peering connection between two VPCs at the same time.
- Any tags that you create for your VPC peering connection are only applied in the account or Region in which you create them.
- You cannot connect to or query the Amazon DNS server in a peer VPC.
- If the IPv4 CIDR block of a VPC in a VPC peering connection falls outside of the private IPv4 address ranges specified by RFC 1918, private DNS hostnames for that VPC cannot be resolved to private IP addresses. To resolve private DNS hostnames to private IP addresses, you can enable DNS resolution support for the VPC peering connection. For more information, see Enable DNS resolution for a VPC peering connection.
- You can enable resources on either side of a VPC peering connection to communicate over IPv6. You must associate an IPv6 CIDR block with each VPC, enable the instances in the VPCs for IPv6 communication, and route IPv6 traffic intended for the peer VPC to the VPC peering connection.
- Unicast reverse path forwarding in VPC peering connections is not supported. For more information, see Routing for response traffic.

Overlapping CIDR blocks

- You cannot create a VPC peering connection between VPCs that have matching or overlapping IPv4 or IPv6 CIDR blocks.
- If you have multiple IPv4 CIDR blocks, you can't create a VPC peering connection if any of the CIDR blocks overlap, even if you intend to use only the non-overlapping CIDR blocks or only IPv6 CIDR blocks.

Transitive peering

- VPC peering does not support transitive peering relationships. For example, if there are VPC peering connections between VPC A and VPC B, and between VPC A and VPC C, you can't route traffic from VPC B to VPC C through VPC A. To route traffic between VPC B and VPC C, you must create a VPC peering connection between them. For more information, see Three VPCs peered together.

Edge to edge routing through a gateway or private connection

- If VPC A has an internet gateway, resources in VPC B can't use the internet gateway in VPC A to access the internet.
- If VPC A has an NAT device that provides internet access to subnets in VPC A, resources in VPC B can't use the NAT device in VPC A to access the internet.
- If VPC A has a VPN connection to a corporate network, resources in VPC B can't use the VPN connection to communicate with the corporate network.
- If VPC A has an Direct Connect connection to a corporate network, resources in VPC B can't use the Direct Connect connection to communicate with the corporate network.
- If VPC A has a gateway endpoint that provides connectivity to Amazon S3 to private subnets in VPC A, resources in VPC B can't use the gateway endpoint to access Amazon S3.

Inter-Region VPC peering connections

- For jumbo frames, the Maximum Transmission Unit (MTU) between VPC peering connections within the same Region is 9001 bytes. The MTU for inter-region VPC peering connections is 8500 bytes. For more information about jumbo frames, see Jumbo frames (9001 MTU) in the *Amazon EC2 User Guide*.
- You must enable DNS resolution support for the VPC peering connection to resolve private DNS hostnames of the peered VPC to private IP addresses, even if the IPv4 CIDR for the VPC falls into the private IPv4 address ranges specified by RFC 1918.

Shared VPCs and subnets

- Only VPC owners can work with (describe, create, accept, reject, modify, or delete) peering connections. Participants cannot work with peering connections. For more information see, Share your VPC with other accounts in the *Amazon VPC User Guide*.

VPC peering connections

VPC peering enables you to connect two VPCs in the same or different AWS Regions. This enables instances in one VPC to communicate with instances in the other VPC as if they were all part of the same network.

VPC peering creates a direct network route between the two VPCs using private IPv4 addresses or IPv6 addresses. Traffic sent between the connected VPCs does not traverse the internet, a VPN connection, or an AWS Direct Connect connection. This makes VPC peering a secure way to share resources, such as databases or web servers, across VPC boundaries.

To establish a VPC peering connection, you create a peering connection request from one VPC and the owner of the other VPC accepts the request. After the connection is established, you can update your route tables to route traffic between the VPCs. This allows instances in one VPC to access resources in the other VPC.

VPC peering is an important tool for building multi-VPC architectures and sharing resources across organizational boundaries in AWS. It provides a simple, low-latency way to connect VPCs without the complexity of configuring a VPN or other networking service.

Use the following procedures to create and work with VPC peering connections.

Tasks

- Create a VPC peering connection
- Accept or reject a VPC peering connection
- Update your route tables for a VPC peering connection
- Update your security groups to reference peer security groups
- Enable DNS resolution for a VPC peering connection
- Delete a VPC peering connection
- Troubleshoot a VPC peering connection

Create a VPC peering connection

To create a VPC peering connection, first create a request to peer with another VPC. To activate the request, the owner of the accepter VPC must accept the request. The following peering connections are supported:

- Between VPCs in the same account and Region
- Between VPCs in the same account and different Regions
- Between VPCs in different accounts and the same Region
- Between VPCs in different accounts and Regions

For an inter-Region VPC peering connection, the request must be made from the Region of the requester VPC, and the request must be accepted from the Region of the accepter VPC. For more information, see the section called “Accept or reject”.

Tasks

Prerequisites

- Review the limitations for VPC peering connections.
- Ensure that the VPCs do not have overlapping IPv4 CIDR blocks. If they overlap, the status of the VPC peering connection immediately goes to `failed`. This limitation applies even if the VPCs have unique IPv6 CIDR blocks.

Create a peering connection using the console

Use the following procedure to create a VPC peering connection.

To create a peering connection using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Peering connections**.
3. Choose **Create peering connection**.
4. (Optional) For **Name**, specify a name the VPC peering connection. This creates a tag with a key of Name and the value that you specify.
5. For **VPC ID (Requester)**, select a VPC from the current account.
6. Under **Select another VPC to peer with**, do the following:

a. For **Account**, to peer with a VPC in another account, choose **Another account** and enter the account ID . Otherwise, keep **My account**.

b. For **Region**, to peer with a VPC in another Region, choose **Another Region** and choose the Region . Otherwise, keep **This Region**.

c. For **VPC ID (Accepter)**, select a VPC from the specified account and Region.

7. (Optional) To add a tag, choose **Add new tag** and enter the tag key and tag value.
8. Choose **Create peering connection**.
9. The owner of the accepter account must accept the peering connection. For more information, see the section called “Accept or reject”.
10. Update the route tables for both VPCs to enable communication between them. For more information, see the section called “Update route tables”.

Create a peering connection using the command line

You can create a VPC peering connection using the following commands:

- create-vpc-peering-connection (AWS CLI)
- New-EC2VpcPeeringConnection (AWS Tools for Windows PowerShell)

Accept or reject a VPC peering connection

A VPC peering connection that's in the `pending-acceptance` state must be accepted by the owner of the accepter VPC to be activated. For more information about the `Deleted` peering connection status, see VPC peering connection lifecycle. You can't accept a VPC peering connection request that you sent to another AWS account. To create a VPC peering connection between VPCs in the same AWS account, you can both create and accept the request yourself.

You can reject any VPC peering connection request that you've received that's in the `pending-acceptance` state. You should only accept VPC peering connections from AWS accounts that you know and trust; you can reject any unwanted requests. For more information about the `Rejected` peering connection status, see VPC peering connection lifecycle.

Important

Do not accept VPC peering connections from unknown AWS accounts. A malicious user may have sent you a VPC peering connection request to gain unauthorized network access to your VPC. This is known as peer phishing. You can safely reject unwanted VPC peering connection requests without any risk of the requester gaining access to any information about your AWS account or your VPC. For more information, see Accept or reject a VPC peering connection. You can also ignore the request and let it expire; by default, requests expire after 7 days.

To accept or reject a peering connection using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Use the Region selector to choose the Region of the accepter VPC.
3. In the navigation pane, choose **Peering connections**.
4. To reject a peering connection, select the VPC peering connection, and choose **Actions**, **Reject request**. When prompted for confirmation, choose **Reject request**.
5. To accept a peering connection, select the pending VPC peering connection (the status is `pending-acceptance`), and choose **Actions**, **Accept request**. For more information about peering connection lifecycle statuses, see VPC peering connection lifecycle.

   If there is no pending VPC peering connection, verify that you selected the Region of the accepter VPC.
6. When prompted for confirmation, choose **Accept request**.
7. Choose **Modify my route tables now** to add a route to the VPC route table so that you can send and receive traffic across the peering connection. For more information, see Update your route tables for a VPC peering connection.

To accept a peering connection using the command line

- accept-vpc-peering-connection (AWS CLI)
- Approve-EC2VpcPeeringConnection (AWS Tools for Windows PowerShell)

To reject a peering connection using the command line

- reject-vpc-peering-connection (AWS CLI)
- Deny-EC2VpcPeeringConnection (AWS Tools for Windows PowerShell)

Update your route tables for a VPC peering connection

To enable private IPv4 traffic between instances in peered VPCs, you must add a route to the route tables associated with the subnets for both instances. The route destination is the CIDR block (or portion of the CIDR block) of the peer VPC and the target is the ID of the VPC peering connection. For more information, see Configure route tables in the *Amazon VPC User Guide*.

The following is an example of the route tables that enables communication between instances in two peered VPCs, VPC A and VPC B. Each table has a local route and a route that sends traffic for the peer VPC to the VPC peering connection.

Route table	Destination	Target
VPC A	*VPC A CIDR*	Local
	VPC B CIDR	pcx-*11112222*
VPC B	*VPC B CIDR*	Local
	VPC A CIDR	pcx-*11112222*

Similarly, if the VPCs in the VPC peering connection have associated IPv6 CIDR blocks, you can add routes that enable communication with the peer VPC over IPv6.

For more information about supported route table configurations for VPC peering connections, see Common VPC peering connection configurations.

Considerations

- If you have a VPC peered with multiple VPCs that have overlapping or matching IPv4 CIDR blocks, ensure that your route tables are configured to avoid sending response traffic from your VPC to the incorrect VPC. AWS currently does not support unicast reverse path forwarding in

VPC peering connections that checks the source IP of packets and routes reply packets back to the source. For more information, see Routing for response traffic.

- Your account has a quota on the number of entries you can add per route table. If the number of VPC peering connections in your VPC exceeds the route table entry quota for a single route table, consider using multiple subnets that are each associated with a custom route table.
- You can add a route for a VPC peering connection that's in the `pending-acceptance` state. However, the route has a state of `blackhole`, and has no effect until the VPC peering connection is in the `active` state.

To add an IPv4 route for a VPC peering connection

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route tables**.
3. Select the check box next to the route table that's associated with the subnet in which your instance resides.

   If you do not have a route table explicitly associated with that subnet, the main route table for the VPC is implicitly associated with the subnet.
4. Choose **Actions**, **Edit routes**.
5. Choose **Add route**.
6. For **Destination**, enter the IPv4 address range to which the network traffic in the VPC peering connection must be directed. You can specify the entire IPv4 CIDR block of the peer VPC, a specific range, or an individual IPv4 address, such as the IP address of the instance with which to communicate. For example, if the CIDR block of the peer VPC is `10.0.0.0/16`, you can specify a portion `10.0.0.0/24`, or a specific IP address `10.0.0.7/32`.
7. For **Target**, select the VPC peering connection.
8. Choose **Save changes**.

The owner of the peer VPC must also complete these steps to add a route to direct traffic back to your VPC through the VPC peering connection.

If you have resources in different AWS Regions that use IPv6 addresses, you can create an inter-Region peering connection. You can then add an IPv6 route for communication between the resources.

To add an IPv6 route for a VPC peering connection

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route tables**.
3. Select the check box next to the route table that's associated with the subnet in which your instance resides.

   > **Note**
   >
   > If you do not have a route table associated with that subnet, select the main route table for the VPC, as the subnet then uses this route table by default.

4. Choose **Actions**, **Edit routes**.
5. Choose **Add route**.
6. For **Destination**, enter the IPv6 address range for the peer VPC. You can specify the entire IPv6 CIDR block of the peer VPC, a specific range, or an individual IPv6 address. For example, if the CIDR block of the peer VPC is `2001:db8:1234:1a00::/56`, you can specify a portion `2001:db8:1234:1a00::/64`, or a specific IP address `2001:db8:1234:1a00::123/128`.
7. For **Target**, select the VPC peering connection.
8. Choose **Save changes**.

For more information, see Route tables in the *Amazon VPC User Guide*.

To add or replace a route using the command line

- create-route and replace-route(AWS CLI)
- New-EC2Route and Set-EC2Route(AWS Tools for Windows PowerShell)

Update your security groups to reference peer security groups

You can update the inbound or outbound rules for your VPC security groups to reference security groups for peered VPCs. Doing so allows traffic to flow to and from instances that are associated with the referenced security group in the peered VPC.

> **Note**
>
> Security groups in a peer VPC are not displayed in the console for you to select.

Requirements

- To reference a security group in a peer VPC, the VPC peering connection must be in the `active` state.
- The peer VPC can be a VPC in your account, or a VPC in another AWS account. To reference a security group that is in another AWS account but the same Region, include the account number with the ID of the security group. For example, `123456789012/sg-1a2b3c4d`.
- You can't reference the security group of a peer VPC that's in a different Region. Instead, use the CIDR block of the peer VPC.
- If you configure routes to forward the traffic between two instances in different subnets through a middlebox appliance, you must ensure that the security groups for both instances allow traffic to flow between the instances. The security group for each instance must reference the private IP address of the other instance, or the CIDR range of the subnet that contains the other instance, as the source. If you reference the security group of the other instance as the source, this does not allow traffic to flow between the instances.

To update your security group rules using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security groups**.
3. Select the security group, and do one of the following:
   - To modify the inbound rules, choose **Actions**, **Edit inbound rules**.
   - To modify the outbound rules, choose **Actions**, **Edit outbound rules**.
4. To add a rule, choose **Add rule** and specify the type, protocol, and port range. For **Source** (inbound rule) or **Destination** (outbound rule), do one of the following:
   - For a peer VPC in same account and Region, enter the ID of the security group.
   - For a peer VPC in a different account but the same Region, enter the account ID and security group ID, separated by a forward slash (for example, `123456789012/sg-1a2b3c4d`).
   - For a peer VPC in a different Region, enter the CIDR block of the peer VPC.

5. To edit an existing rule, change its values (for example, the source or the description).
6. To delete a rule, choose **Delete** next to the rule.
7. Choose **Save rules**.

To update inbound rules using the command line

- authorize-security-group-ingress and revoke-security-group-ingress (AWS CLI)
- Grant-EC2SecurityGroupIngress and Revoke-EC2SecurityGroupIngress (AWS Tools for Windows PowerShell)

For example, to update your security group `sg-aaaa1111` to allow inbound access over HTTP from `sg-bbbb2222` for a peer VPC, use the following command. If the peer VPC is in the same Region but a different account, add `--group-owner` *`aws-account-id`*.

```
aws ec2 authorize-security-group-ingress --group-id sg-aaaa1111 --protocol tcp --
port 80 --source-group sg-bbbb2222
```

To update outbound rules using the command line

- authorize-security-group-egress and revoke-security-group-egress (AWS CLI)
- Grant-EC2SecurityGroupEgress and Revoke-EC2SecurityGroupEgress (AWS Tools for Windows PowerShell)

After you've updated the security group rules, use the describe-security-groups command to view the referenced security group in your security group rules.

Identify your referenced security groups

To determine if your security group is being referenced in the rules of a security group in a peer VPC, use one of the following commands for one or more security groups in your account.

- describe-security-group-references (AWS CLI)
- Get-EC2SecurityGroupReference (AWS Tools for Windows PowerShell)

In the following example, the response indicates that security group `sg-bbbb2222` is being referenced by a security group in VPC `vpc-aaaaaaaa`:

```
aws ec2 describe-security-group-references --group-id sg-bbbb2222
```

```
{
  "SecurityGroupsReferenceSet": [
    {
      "ReferencingVpcId": "vpc-aaaaaaaa",
      "GroupId": "sg-bbbb2222",
      "VpcPeeringConnectionId": "pcx-b04deed9"
    }
  ]
}
```

If the VPC peering connection is deleted, or if the owner of the peer VPC deletes the referenced security group, the security group rule becomes stale.

View and delete with stale security group rules

A stale security group rule is a rule that references a deleted security group in the same VPC or in a peer VPC, or that references a security group in a peer VPC for which the VPC peering connection has been deleted. When a security group rule becomes stale, it's not automatically removed from your security group—you must manually remove it. If a security group rule is stale because the VPC peering connection was deleted, the rule will no longer be marked as stale if you create a new VPC peering connection with the same VPCs.

You can view and delete the stale security group rules for a VPC using the Amazon VPC console.

To view and delete stale security group rules

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security groups**.
3. Choose **Actions**, **Manage stale rules**.
4. For **VPC**, choose the VPC with the stale rules.
5. Choose **Edit**.
6. Choose the **Delete** button next to the rule that you want to delete. Choose **Preview changes**, **Save rules**.

To describe your stale security group rules using the command line

- describe-stale-security-groups (AWS CLI)
- Get-EC2StaleSecurityGroup (AWS Tools for Windows PowerShell)

In the following example, VPC A (`vpc-aaaaaaaa`) and VPC B were peered, and the VPC peering connection was deleted. Your security group `sg-aaaa1111` in VPC A references `sg-bbbb2222` in VPC B. When you run the `describe-stale-security-groups` command for your VPC, the response indicates that security group `sg-aaaa1111` has a stale SSH rule that references `sg-bbbb2222`.

```
aws ec2 describe-stale-security-groups --vpc-id vpc-aaaaaaaa
```

```
{
    "StaleSecurityGroupSet": [
        {
            "VpcId": "vpc-aaaaaaaa",
            "StaleIpPermissionsEgress": [],
            "GroupName": "Access1",
            "StaleIpPermissions": [
                {
                    "ToPort": 22,
                    "FromPort": 22,
                    "UserIdGroupPairs": [
                        {
                            "VpcId": "vpc-bbbbbbbb",
                            "PeeringStatus": "deleted",
                            "UserId": "123456789101",
                            "GroupName": "Prod1",
                            "VpcPeeringConnectionId": "pcx-b04deed9",
                            "GroupId": "sg-bbbb2222"
                        }
                    ],
                    "IpProtocol": "tcp"
                }
            ],
            "GroupId": "sg-aaaa1111",
            "Description": "Reference remote SG"
        }
    ]
```

```
}
```

After you've identified the stale security group rules, you can delete them using the revoke-security-group-ingress or revoke-security-group-egress commands.

Enable DNS resolution for a VPC peering connection

The DNS settings for a VPC peering connection determine how public DNS hostnames are resolved for requests that traverse the VPC peering connection. If an EC2 instance on one side of a VPC peering connection sends a request to an EC2 instance on the other side using the public IPv4 DNS hostname of the instance, the DNS hostname is resolved as follows.

DNS resolution disabled (default)

The public IPv4 DNS hostname resolves to the public IPv4 address of the instance.

DNS resolution enabled

The public IPv4 DNS hostname resolves to the private IPv4 address of the instance.

Requirements

- Both VPCs must be enabled for DNS hostnames and DNS resolution. For more information, see DNS attributes for your VPC in the *Amazon VPC User Guide*.
- The peering connection must be in the `active` state. You can't enable DNS resolution when you create a peering connection.
- The owner of the requester VPC must modify the requester VPC peering options, and the owner of the accepter VPC must modify the accepter VPC peering options. If the VPCs are in the same account, you can enable DNS resolution for the requester and accepter VPCs at the same time. This works for both same-region and cross-region VPC peering connections.

To enable DNS resolution for a peering connection using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Peering connections**.
3. Select the VPC peering connection.
4. Choose **Actions**, **Edit DNS settings**.

5. To enable DNS resolution for requests from the requester VPC, select **Requester DNS resolution**, **Allow accepter VPC to resolve the DNS of requester VPC**.
6. To ensure DNS resolution for requests from the accepter VPC, select **Accepter DNS resolution**, **Allow requester VPC to resolve the DNS of accepter VPC**.
7. Choose **Save changes**.

To enable DNS resolution using the command line

- modify-vpc-peering-connection-options (AWS CLI)
- Edit-EC2VpcPeeringConnectionOption (AWS Tools for Windows PowerShell)

To describe VPC peering connection options using the command line

- describe-vpc-peering-connections (AWS CLI)
- Get-EC2VpcPeeringConnection (AWS Tools for Windows PowerShell)

Delete a VPC peering connection

Either owner of a VPC in a peering connection can delete the VPC peering connection at any time. You can also delete a VPC peering connection that you've requested that is still in the `pending-acceptance` state.

You cannot delete the VPC peering connection when the VPC peering connection is in the `rejected` state. We automatically delete the connection for you.

Deleting a VPC in the Amazon VPC console that's part of an active VPC peering connection also deletes the VPC peering connection. If you have requested a VPC peering connection with a VPC in another account, and you delete your VPC before the other party has accepted the request, the VPC peering connection is also deleted. You cannot delete a VPC for which you have a `pending-acceptance` request from a VPC in another account. You must first reject the VPC peering connection request.

When you delete a peering connection, the status is set to `Deleting` and then `Deleted`. After you delete a connection, it can't be accepted, rejected, or edited. For more information about how long the peering connection remains visible, see VPC peering connection lifecycle.

To delete a VPC peering connection

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Peering connections**.
3. Select the VPC peering connection.
4. Choose **Actions**, **Delete peering connection**.
5. When prompted for confirmation, enter **`delete`** and then choose **Delete**.

To delete a VPC peering connection using the command line

- delete-vpc-peering-connection (AWS CLI)
- Remove-EC2VpcPeeringConnection (AWS Tools for Windows PowerShell)

Troubleshoot a VPC peering connection

If you're having trouble connecting to a resource in a VPC from a resource in a peer VPC, do the following:

- For each resource in each VPC, verify that the route table for its subnet contains a route that sends traffic destined for the peer VPC to the VPC peering connection. This ensures network traffic can properly flow between the two VPCs. For more information, see Update route tables.
- For any EC2 instances involved, verify that the security groups for those instances allow inbound and outbound traffic from the peer VPC. Security group rules control which traffic is permitted to access your EC2 instances. For more information, see Reference peer security groups.
- Check that the network ACLs for the subnets containing your resources allow the necessary traffic from the peer VPC. Network ACLs are an additional layer of security that filter traffic at the subnet level.

If you're still having issues, you can leverage Reachability Analyzer. Reachability Analyzer can help identify the specific component - whether a route table, security group, or network ACL - that is causing the connectivity problem between the two VPCs. For more information, see the Reachability Analyzer Guide.

Thoroughly verifying your VPC networking configurations is key to troubleshooting and resolving any VPC peering connection issues you may encounter.

Common VPC peering connection configurations

This section describes two common types of VPC peering configurations that you can implement:

- **VPC peering configurations with routes to an entire VPC**: In this configuration, you create a route in each VPC's route table that sends all traffic destined for the peer VPC to the VPC peering connection. This allows any resource in one VPC to communicate with any resource in the peer VPC, simplifying management. However, it also means that all traffic between the VPCs will flow through the peering connection, which could become a bottleneck if the traffic volume is high.
- **VPC peering configurations with specific routes**: Alternatively, you can create more granular routes in each VPC's route table that only send traffic to specific subnets or resources in the peer VPC. This allows you to limit the traffic flowing through the peering connection to only what is necessary, which can be more efficient. However, it also requires more maintenance, as you'll need to update the route tables any time you add new resources in the peer VPC that need to communicate.

The best approach depends on factors like the size and complexity of your VPC architecture, the volume of traffic expected between the VPCs, and your organizational needs around security and resource access. Many enterprises use a hybrid approach, with broad routes for common traffic patterns and specific routes for more sensitive or bandwidth-intensive use cases.

Configurations

- VPC peering configurations with routes to an entire VPC
- VPC peering configurations with specific routes

VPC peering configurations with routes to an entire VPC

You can configure VPC peering connections so that your route tables have access to the entire CIDR block of the peer VPC. For more information about scenarios in which you might need a specific VPC peering connection configuration, see VPC peering connection networking scenarios. For more information about creating and working with VPC peering connections, see VPC peering connections.

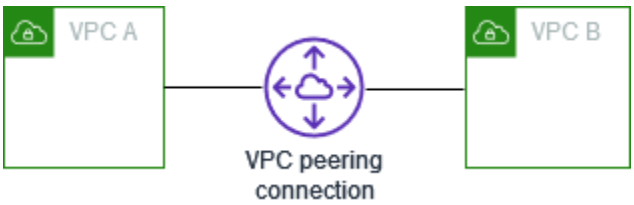
For more information about updating your route tables, see Update your route tables for a VPC peering connection.

Configurations

- Two VPCs peered together
- One VPC peered with two VPCs
- Three VPCs peered together
- Multiple VPCs peered together

Two VPCs peered together

In this configuration, there is a peering connection between VPC A and VPC B (`pcx-11112222`). The VPCs are in the same AWS account and their CIDR blocks do not overlap.



You might use this configuration when you have two VPCs that require access to each others' resources. For example, you set up VPC A for your accounting records and VPC B for your financial records, and these each VPC must be able to access resources from the other VPC without restriction.

Single VPC CIDR

Update the route table for each VPC with a route that sends traffic for the CIDR block of the peer VPC to the VPC peering connection.

Route table	Destination	Target
VPC A	*VPC A CIDR*	Local
	VPC B CIDR	pcx-11112222
VPC B	*VPC B CIDR*	Local
	VPC A CIDR	pcx-11112222

Multiple IPv4 VPC CIDRs

If VPC A and VPC B have multiple associated IPv4 CIDR blocks, you can update the route table for each VPC with routes for some or all of the IPv4 CIDR blocks of the peer VPC.

Route table	Destination	Target
VPC A	*VPC A CIDR 1*	Local
	VPC A CIDR 2	Local
	VPC B CIDR 1	pcx-11112222
	VPC B CIDR 2	pcx-11112222
VPC B	*VPC B CIDR 1*	Local
	VPC B CIDR 2	Local
	VPC A CIDR 1	pcx-11112222
	VPC A CIDR 2	pcx-11112222

IPv4 and IPv6 VPC CIDRs

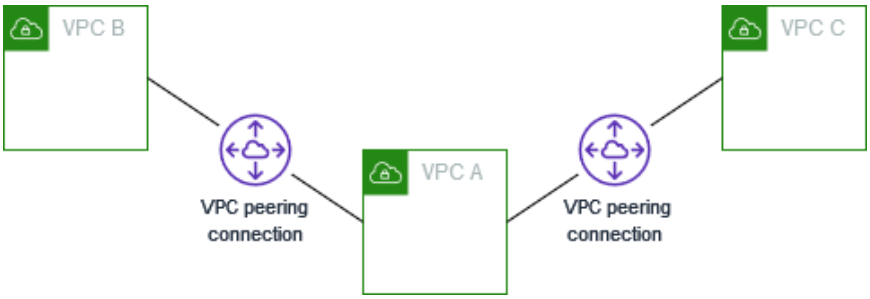
If VPC A and VPC B have associated IPv6 CIDR blocks, you can update the route table for each VPC with routes for both the IPv4 and IPv6 CIDR blocks of the peer VPC.

Route table	Destination	Target
VPC A	*VPC A IPv4 CIDR*	Local
	VPC A IPv6 CIDR	Local
	VPC B IPv4 CIDR	pcx-11112222
	VPC B IPv6 CIDR	pcx-11112222
VPC B	*VPC B IPv4 CIDR*	Local
	VPC B IPv6 CIDR	Local

Route table	Destination	Target
	VPC A IPv4 CIDR	pcx-11112222
	VPC A IPv6 CIDR	pcx-11112222

One VPC peered with two VPCs

In this configuration, there is a central VPC (VPC A), a peering connection between VPC A and VPC B (pcx-12121212), and a peering connection between VPC A and VPC C (pcx-23232323). All three VPCs are in the same AWS account and their CIDR blocks do not overlap.



VPC B and VPC C can't send traffic directly to each other through a VPC A, because VPC peering does not support transitive peering relationships. You can create a VPC peering connection between VPC B and VPC C, as shown in Three VPCs peered together. For more information about unsupported peering scenarios, see the section called “VPC peering limitations”.

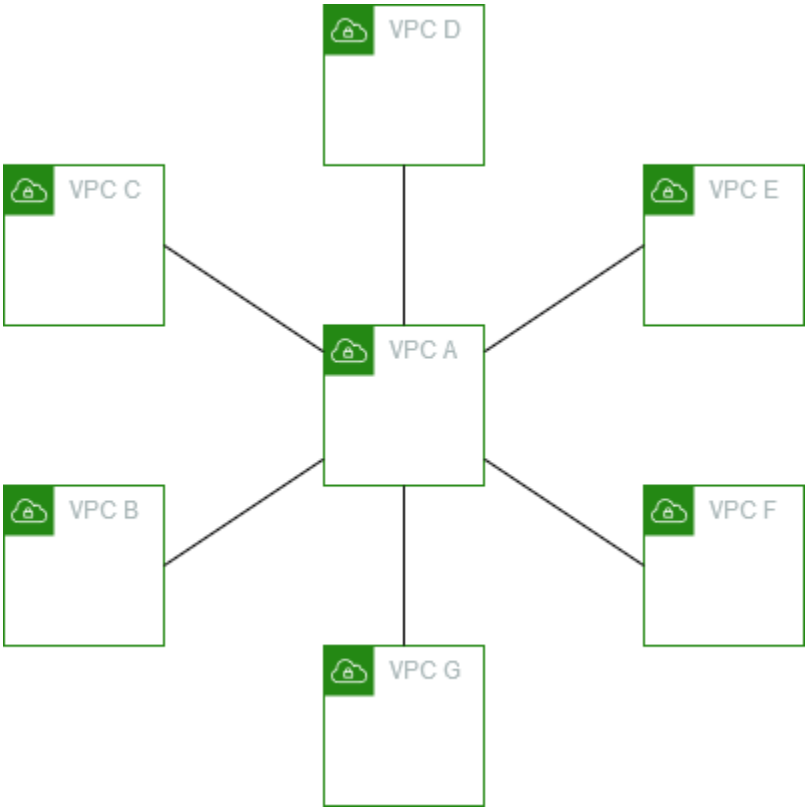
You might use this configuration when you have resources on a central VPC, such as a repository of services, that other VPCs need to access. The other VPCs do not need access to each others' resources; they only need to access resources in the central VPC.

Update the route table for each VPC as follows to implement this configuration using one CIDR block per VPC.

Route table	Destination	Target
VPC A	*VPC A CIDR*	Local
	VPC B CIDR	pcx-12121212
	VPC C CIDR	pcx-23232323

Route table	Destination	Target
VPC B	*VPC B CIDR*	Local
	VPC A CIDR	pcx-12121212
VPC C	*VPC C CIDR*	Local
	VPC A CIDR	pcx-23232323

You can extend this configuration to additional VPCs. For example, VPC A is peered with VPC B through VPC G using both IPv4 and IPv6 CIDRs, but the other VPCs are not peered to each other. In this diagram, the lines represent VPC peering connections.



Update the route table as follows.

Route table	Destination	Target
VPC A	*VPC A IPv4 CIDR*	Local

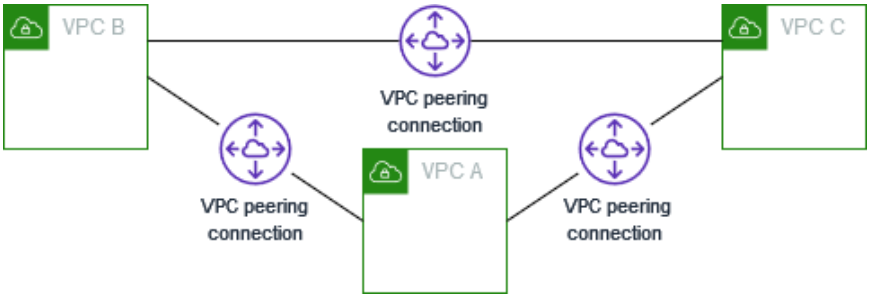
Route table	Destination	Target
	VPC A IPv6 CIDR	Local
	VPC B IPv4 CIDR	pcx-aaaabbbb
	VPC B IPv6 CIDR	pcx-aaaabbbb
	VPC C IPv4 CIDR	pcx-aaaacccc
	VPC C IPv6 CIDR	pcx-aaaacccc
	VPC D IPv4 CIDR	pcx-aaaadddd
	VPC D IPv6 CIDR	pcx-aaaadddd
	VPC E IPv4 CIDR	pcx-aaaaeeee
	VPC E IPv6 CIDR	pcx-aaaaeeee
	VPC F IPv4 CIDR	pcx-aaaaffff
	VPC F IPv6 CIDR	pcx-aaaaffff
	VPC G IPv4 CIDR	pcx-aaaagggg
	VPC G IPv6 CIDR	pcx-aaaagggg
VPC B	*VPC B IPv4 CIDR*	Local
	VPC B IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaabbbb
	VPC A IPv6 CIDR	pcx-aaaabbbb
VPC C	*VPC C IPv4 CIDR*	Local
	VPC C IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaacccc

Route table	Destination	Target
	VPC A IPv6 CIDR	pcx-aaaacccc
VPC D	*VPC D IPv4 CIDR*	Local
	VPC D IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaadddd
	VPC A IPv6 CIDR	pcx-aaaadddd
VPC E	*VPC E IPv4 CIDR*	Local
	VPC E IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaaeeee
	VPC A IPv6 CIDR	pcx-aaaaeeee
VPC F	*VPC F IPv4 CIDR*	Local
	VPC F IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaaffff
	VPC A IPv6 CIDR	pcx-aaaaffff
VPC G	*VPC G IPv4 CIDR*	Local
	VPC G IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaagggg
	VPC A IPv6 CIDR	pcx-aaaagggg

Three VPCs peered together

In this configuration, there are three VPCs in the same AWS account with CIDR blocks that do not overlap. The VPCs are peered in a full mesh as follows:

- VPC A is peered to VPC B through VPC peering connection `pcx-aaaabbbb`
- VPC A is peered to VPC C through VPC peering connection `pcx-aaaacccc`
- VPC B is peered to VPC C through VPC peering connection `pcx-bbbbcccc`



You might use this configuration when you have VPCs that need to share resources with each other without restriction. For example, as a file sharing system.

Update the route table for each VPC as follows to implement this configuration.

Route table	Destination	Target
VPC A	*VPC A CIDR*	Local
	VPC B CIDR	pcx-aaaabbbb
	VPC C CIDR	pcx-aaaacccc
VPC B	*VPC B CIDR*	Local
	VPC A CIDR	pcx-aaaabbbb
	VPC C CIDR	pcx-bbbbcccc
VPC C	*VPC C CIDR*	Local
	VPC A CIDR	pcx-aaaacccc
	VPC B CIDR	pcx-bbbbcccc

If VPC A and VPC B have both IPv4 and IPv6 CIDR blocks, but VPC C does not have an IPv6 CIDR block, update the route tables as follows. Resources in VPC A and VPC B can communicate using

IPv6 over the VPC peering connection. However, VPC C cannot communicate with either VPC A or VPC B using IPv6.

Route tables	Destination	Target
VPC A	*VPC A IPv4 CIDR*	Local
	VPC A IPv6 CIDR	Local
	VPC B IPv4 CIDR	pcx-aaaabbbb
	VPC B IPv6 CIDR	pcx-aaaabbbb
	VPC C IPv4 CIDR	pcx-aaaacccc
VPC B	*VPC B IPv4 CIDR*	Local
	VPC B IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaabbbb
	VPC A IPv6 CIDR	pcx-aaaabbbb
	VPC C IPv4 CIDR	pcx-bbbbcccc
VPC C	*VPC C IPv4 CIDR*	Local
	VPC A IPv4 CIDR	pcx-aaaacccc
	VPC B IPv4 CIDR	pcx-bbbbcccc

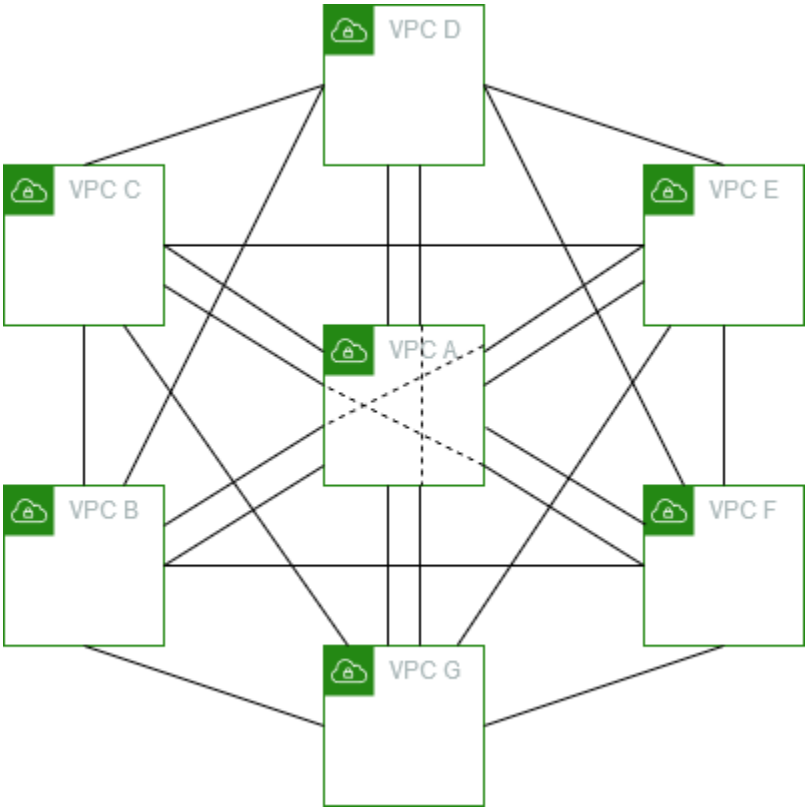
Multiple VPCs peered together

In this configuration, there are seven VPCs peered in a full mesh configuration. The VPCs are in the same AWS account and their CIDR blocks do not overlap.

VPC	VPC	VPC peering connection
A	B	pcx-aaaabbbb

VPC	VPC	VPC peering connection
A	C	pcx-aaaacccc
A	D	pcx-aaaadddd
A	E	pcx-aaaaeeee
A	F	pcx-aaaaffff
A	G	pcx-aaaagggg
B	C	pcx-bbbbcccc
B	D	pcx-bbbbdddd
B	E	pcx-bbbbeeee
B	F	pcx-bbbbffff
B	G	pcx-bbbbgggg
C	D	pcx-ccccdddd
C	E	pcx-cccceeee
C	F	pcx-ccccffff
C	G	pcx-ccccgggg
D	E	pcx-ddddeeee
D	F	pcx-ddddffff
D	G	pcx-ddddgggg
E	F	pcx-eeeeffff
E	G	pcx-eeeegggg
F	G	pcx-ffffgggg

You might use this configuration when you have multiple VPCs that must be able to access each others' resources without restriction. For example, as a file sharing network. In this diagram, the lines represent VPC peering connections.



Update the route table for each VPC as follows to implement this configuration.

Route table	Destination	Target
VPC A	*VPC A CIDR*	Local
	VPC B CIDR	pcx-aaaabbbb
	VPC C CIDR	pcx-aaaacccc
	VPC D CIDR	pcx-aaaadddd
	VPC E CIDR	pcx-aaaaeeee
	VPC F CIDR	pcx-aaaaffff
	VPC G CIDR	pcx-aaaagggg

Route table	Destination	Target
VPC B	*VPC B CIDR*	Local
	VPC A CIDR	pcx-aaaabbbb
	VPC C CIDR	pcx-bbbbcccc
	VPC D CIDR	pcx-bbbbdddd
	VPC E CIDR	pcx-bbbbeeee
	VPC F CIDR	pcx-bbbbffff
	VPC G CIDR	pcx-bbbbgggg
VPC C	*VPC C CIDR*	Local
	VPC A CIDR	pcx-aaaacccc
	VPC B CIDR	pcx-bbbbcccc
	VPC D CIDR	pcx-ccccdddd
	VPC E CIDR	pcx-cccceeee
	VPC F CIDR	pcx-ccccffff
	VPC G CIDR	pcx-ccccgggg
VPC D	*VPC D CIDR*	Local
	VPC A CIDR	pcx-aaaadddd
	VPC B CIDR	pcx-bbbbdddd
	VPC C CIDR	pcx-ccccdddd
	VPC E CIDR	pcx-ddddeeee
	VPC F CIDR	pcx-ddddffff

Route table	Destination	Target
	VPC G CIDR	pcx-ddddgggg
VPC E	*VPC E CIDR*	Local
	VPC A CIDR	pcx-aaaaeeee
	VPC B CIDR	pcx-bbbbeeee
	VPC C CIDR	pcx-cccceeee
	VPC D CIDR	pcx-ddddeeee
	VPC F CIDR	pcx-eeeeffff
	VPC G CIDR	pcx-eeeegggg
VPC F	*VPC F CIDR*	Local
	VPC A CIDR	pcx-aaaaffff
	VPC B CIDR	pcx-bbbbffff
	VPC C CIDR	pcx-ccccffff
	VPC D CIDR	pcx-ddddffff
	VPC E CIDR	pcx-eeeeffff
	VPC G CIDR	pcx-ffffgggg
VPC G	*VPC G CIDR*	Local
	VPC A CIDR	pcx-aaaagggg
	VPC B CIDR	pcx-bbbbgggg
	VPC C CIDR	pcx-ccccgggg
	VPC D CIDR	pcx-ddddgggg

Route table	Destination	Target
	VPC E CIDR	pcx-eeeegggg
	VPC F CIDR	pcx-ffffgggg

If all VPCs have associated IPv6 CIDR blocks, update the route tables as follows.

Route table	Destination	Target
VPC A	*VPC A IPv4 CIDR*	Local
	VPC A IPv6 CIDR	Local
	VPC B IPv4 CIDR	pcx-aaaabbbb
	VPC B IPv6 CIDR	pcx-aaaabbbb
	VPC C IPv4 CIDR	pcx-aaaacccc
	VPC C IPv6 CIDR	pcx-aaaacccc
	VPC D IPv4 CIDR	pcx-aaaadddd
	VPC D IPv6 CIDR	pcx-aaaadddd
	VPC E IPv4 CIDR	pcx-aaaaeeee
	VPC E IPv6 CIDR	pcx-aaaaeeee
	VPC F IPv4 CIDR	pcx-aaaaffff
	VPC F IPv6 CIDR	pcx-aaaaffff
	VPC G IPv4 CIDR	pcx-aaaagggg
	VPC G IPv6 CIDR	pcx-aaaagggg
VPC B	*VPC B IPv4 CIDR*	Local

Route table	Destination	Target
	VPC B IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaabbbb
	VPC A IPv6 CIDR	pcx-aaaabbbb
	VPC C IPv4 CIDR	pcx-bbbbcccc
	VPC C IPv6 CIDR	pcx-bbbbcccc
	VPC D IPv4 CIDR	pcx-bbbbdddd
	VPC D IPv6 CIDR	pcx-bbbbdddd
	VPC E IPv4 CIDR	pcx-bbbbeeee
	VPC E IPv6 CIDR	pcx-bbbbeeee
	VPC F IPv4 CIDR	pcx-bbbbffff
	VPC F IPv6 CIDR	pcx-bbbbffff
	VPC G IPv4 CIDR	pcx-bbbbgggg
	VPC G IPv6 CIDR	pcx-bbbbgggg
VPC C	*VPC C IPv4 CIDR*	Local
	VPC C IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaacccc
	VPC A IPv6 CIDR	pcx-aaaacccc
	VPC B IPv4 CIDR	pcx-bbbbcccc
	VPC B IPv6 CIDR	pcx-bbbbcccc
	VPC D IPv4 CIDR	pcx-ccccdddd

Route table	Destination	Target
	VPC D IPv6 CIDR	pcx-ccccdddd
	VPC E IPv4 CIDR	pcx-cccceeee
	VPC E IPv6 CIDR	pcx-cccceeee
	VPC F IPv4 CIDR	pcx-ccccffff
	VPC F IPv6 CIDR	pcx-ccccffff
	VPC G IPv4 CIDR	pcx-ccccgggg
	VPC G IPv6 CIDR	pcx-ccccgggg
VPC D	*VPC D IPv4 CIDR*	Local
	VPC D IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaadddd
	VPC A IPv6 CIDR	pcx-aaaadddd
	VPC B IPv4 CIDR	pcx-bbbbdddd
	VPC B IPv6 CIDR	pcx-bbbbdddd
	VPC C IPv4 CIDR	pcx-ccccdddd
	VPC C IPv6 CIDR	pcx-ccccdddd
	VPC E IPv4 CIDR	pcx-ddddeeee
	VPC E IPv6 CIDR	pcx-ddddeeee
	VPC F IPv4 CIDR	pcx-ddddffff
	VPC F IPv6 CIDR	pcx-ddddffff
	VPC G IPv4 CIDR	pcx-ddddgggg

Route table	Destination	Target
	VPC G IPv6 CIDR	pcx-ddddgggg
VPC E	*VPC E IPv4 CIDR*	Local
	VPC E IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaaeeee
	VPC A IPv6 CIDR	pcx-aaaaeeee
	VPC B IPv4 CIDR	pcx-bbbbeeee
	VPC B IPv6 CIDR	pcx-bbbbeeee
	VPC C IPv4 CIDR	pcx-cccceeee
	VPC C IPv6 CIDR	pcx-cccceeee
	VPC D IPv4 CIDR	pcx-ddddeeee
	VPC D IPv6 CIDR	pcx-ddddeeee
	VPC F IPv4 CIDR	pcx-eeeeffff
	VPC F IPv6 CIDR	pcx-eeeeffff
	VPC G IPv4 CIDR	pcx-eeeegggg
	VPC G IPv6 CIDR	pcx-eeeegggg
VPC F	*VPC F IPv4 CIDR*	Local
	VPC F IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaaffff
	VPC A IPv6 CIDR	pcx-aaaaffff
	VPC B IPv4 CIDR	pcx-bbbbffff

Route table	Destination	Target
	VPC B IPv6 CIDR	pcx-bbbbffff
	VPC C IPv4 CIDR	pcx-ccccffff
	VPC C IPv6 CIDR	pcx-ccccffff
	VPC D IPv4 CIDR	pcx-ddddffff
	VPC D IPv6 CIDR	pcx-ddddffff
	VPC E IPv4 CIDR	pcx-eeeeffff
	VPC E IPv6 CIDR	pcx-eeeeffff
	VPC G IPv4 CIDR	pcx-ffffgggg
	VPC G IPv6 CIDR	pcx-ffffgggg
VPC G	*VPC G IPv4 CIDR*	Local
	VPC G IPv6 CIDR	Local
	VPC A IPv4 CIDR	pcx-aaaagggg
	VPC A IPv6 CIDR	pcx-aaaagggg
	VPC B IPv4 CIDR	pcx-bbbbgggg
	VPC B IPv6 CIDR	pcx-bbbbgggg
	VPC C IPv4 CIDR	pcx-ccccgggg
	VPC C IPv6 CIDR	pcx-ccccgggg
	VPC D IPv4 CIDR	pcx-ddddgggg
	VPC D IPv6 CIDR	pcx-ddddgggg
	VPC E IPv4 CIDR	pcx-eeeegggg

Route table	Destination	Target
	VPC E IPv6 CIDR	pcx-eeeegggg
	VPC F IPv4 CIDR	pcx-ffffgggg
	VPC F IPv6 CIDR	pcx-ffffgggg

VPC peering configurations with specific routes

You can configure route tables for a VPC peering connection to restrict access to a subnet CIDR block, a specific CIDR block (if the VPC has multiple CIDR blocks), or a specific resource in the peer VPC. In these examples, a central VPC is peered to at least two VPCs that have overlapping CIDR blocks.

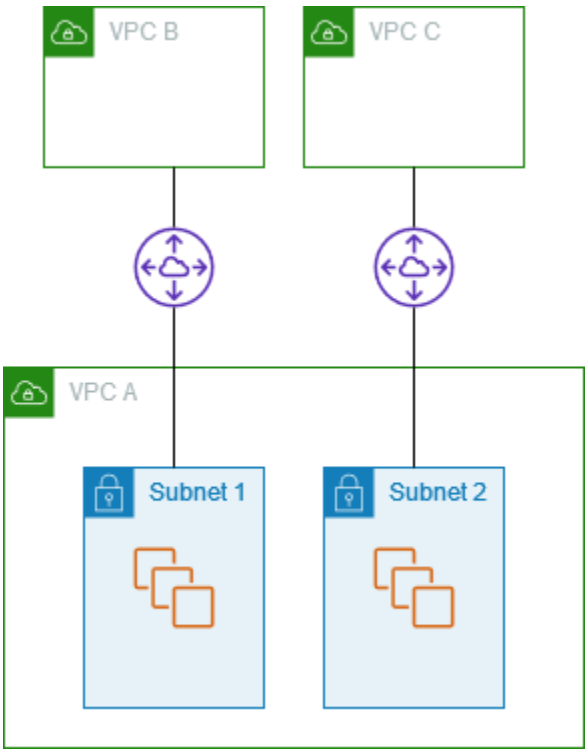
For examples of scenarios in which you might need a specific VPC peering connection configuration, see VPC peering connection networking scenarios. For more information about working with VPC peering connections, see VPC peering connections. For more information about updating your route tables, see Update your route tables for a VPC peering connection.

Configurations

- Two VPCs that access specific subnets in one VPC
- Two VPCs that access specific CIDR blocks in one VPC
- One VPC that accesses specific subnets in two VPCs
- Instances in one VPC that access specific instances in two VPCs
- One VPC that accesses two VPCs using longest prefix matches
- Multiple VPC configurations

Two VPCs that access specific subnets in one VPC

In this configuration, there is a central VPC with two subnets (VPC A), a peering connection between VPC A and VPC B (`pcx-aaaabbbb`), and a peering connection between VPC A and VPC C (`pcx-aaaacccc`). Each VPC requires access to the resources in only one of the subnets in VPC A.



The route table for subnet 1 uses VPC peering connection pcx-aaaabbbb to access the entire CIDR block of VPC B. The route table for VPC B uses pcx-aaaabbbb to access the CIDR block of subnet 1 in VPC A. The route table for subnet 2 uses VPC peering connection pcx-aaaacccc to access the entire CIDR block of VPC C. The route table for VPC C table uses pcx-aaaacccc to access the CIDR block of subnet 2 in VPC A.

Route table	Destination	Target
Subnet 1 (VPC A)	*VPC A CIDR*	Local
	VPC B CIDR	pcx-aaaabbbb
Subnet 2 (VPC A)	*VPC A CIDR*	Local
	VPC C CIDR	pcx-aaaacccc
VPC B	*VPC B CIDR*	Local
	Subnet 1 CIDR	pcx-aaaabbbb
VPC C	*VPC C CIDR*	Local

Route table	Destination	Target
	Subnet 2 CIDR	pcx-aaaacccc

You can extend this configuration to multiple CIDR blocks. Suppose that VPC A and VPC B have both IPv4 and IPv6 CIDR blocks, and that subnet 1 has an associated IPv6 CIDR block. You can enable VPC B to communicate with subnet 1 in VPC A over IPv6 using the VPC peering connection. To do this, add a route to the route table for VPC A with a destination of the IPv6 CIDR block for VPC B, and a route to the route table for VPC B with a destination of the IPv6 CIDR of subnet 1 in VPC A.

Route table	Destination	Target	Notes
Subnet 1 in VPC A	*VPC A IPv4 CIDR*	Local	
	VPC A IPv6 CIDR	Local	Local route that's automatically added for IPv6 communication within the VPC.
	VPC B IPv4 CIDR	pcx-aaaabbbb	
	VPC B IPv6 CIDR	pcx-aaaabbbb	Route to the IPv6 CIDR block of VPC B.
Subnet 2 in VPC A	*VPC A IPv4 CIDR*	Local	
	VPC A IPv6 CIDR	Local	Local route that's automatically added for IPv6 communication within the VPC.
	VPC C IPv4 CIDR	pcx-aaaacccc	
VPC B	*VPC B IPv4 CIDR*	Local	
	VPC B IPv6 CIDR	Local	Local route that's automatically added

Route table	Destination	Target	Notes
			for IPv6 communication within the VPC.
	Subnet 1 IPv4 CIDR	pcx-aaaabbbb	
	Subnet 1 IPv6 CIDR	pcx-aaaabbbb	Route to the IPv6 CIDR block of VPC A.
VPC C	*VPC C IPv4 CIDR*	Local	
	Subnet 2 IPv4 CIDR	pcx-aaaacccc	

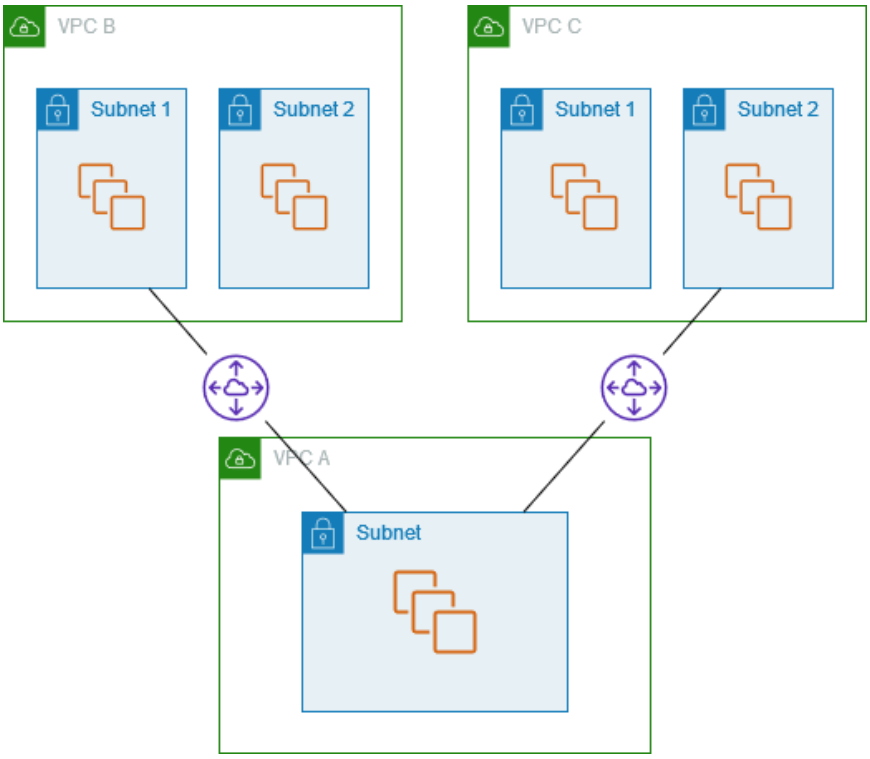
Two VPCs that access specific CIDR blocks in one VPC

In this configuration, there is a central VPC (VPC A), a peering connection between VPC A and VPC B (`pcx-aaaabbbb`), and a peering connection between VPC A and VPC C (`pcx-aaaacccc`). VPC A has one CIDR block for each peering connection.

Route table	Destination	Target
VPC A	*VPC A CIDR 1*	Local
	VPC A CIDR 2	Local
	VPC B CIDR	pcx-aaaabbbb
	VPC C CIDR	pcx-aaaacccc
VPC B	*VPC B CIDR*	Local
	VPC A CIDR 1	pcx-aaaabbbb
VPC C	*VPC C CIDR*	Local
	VPC A CIDR 2	pcx-aaaacccc

One VPC that accesses specific subnets in two VPCs

In this configuration, there is a central VPC (VPC A) with one subnet, a peering connection between VPC A and VPC B (pcx-aaaabbbb), and a peering connection between VPC A and VPC C (pcx-aaaacccc). VPC B and VPC C each have two subnets. The peering connection between VPC A and VPC B uses only one of the subnets in VPC B. The peering connection between VPC A and VPC C uses only one of the subnets in VPC C.



Use this configuration when you have a central VPC that has a single set of resources, such as Active Directory services, that other VPCs need to access. The central VPC does not require full access to the VPCs that it's peered with.

The route table for VPC A uses the peering connections to access only specific subnets in the peered VPCs. The route table for subnet 1 uses the peering connection with VPC A to access the subnet in VPC A. The route table for subnet 2 uses the peering connection with VPC A to access the subnet in VPC A.

Route table	Destination	Target
VPC A	*VPC A CIDR*	Local
	Subnet 1 CIDR	pcx-aaaabbbb

Route table	Destination	Target
	Subnet 2 CIDR	pcx-aaaacccc
Subnet 1 (VPC B)	*VPC B CIDR*	Local
	Subnet in VPC A CIDR	pcx-aaaabbbb
Subnet 2 (VPC C)	*VPC C CIDR*	Local
	Subnet in VPC A CIDR	pcx-aaaacccc

Routing for response traffic

If you have a VPC peered with multiple VPCs that have overlapping or matching CIDR blocks, ensure that your route tables are configured to avoid sending response traffic from your VPC to the incorrect VPC. AWS does not support unicast reverse path forwarding in VPC peering connections that checks the source IP of packets and routes reply packets back to the source.

For example, VPC A is peered with VPC B and VPC C. VPC B and VPC C have matching CIDR blocks, and their subnets have matching CIDR blocks. The route table for subnet 2 in VPC B points to the VPC peering connection `pcx-aaaabbbb` to access the VPC A subnet. The VPC A route table is configured to send traffic destined for the VPC CIDR to peering connection `pcx-aaaaccccc`.

Route table	Destination	Target
Subnet 2 (VPC B)	*VPC B CIDR*	Local
	Subnet in VPC A CIDR	pcx-aaaabbbb
VPC A	*VPC A CIDR*	Local
	VPC C CIDR	pcx-aaaacccc

Suppose that an instance in subnet 2 in VPC B sends traffic to the Active Directory server in VPC A using VPC peering connection `pcx-aaaabbbb`. VPC A sends the response traffic to Active Directory server. However, the VPC A route table is configured to send all traffic within the VPC CIDR range to VPC peering connection `pcx-aaaacccc`. If subnet 2 in VPC C has an instance with the same IP

address as the instance in subnet two of VPC B, it receives the response traffic from VPC A. The instance in subnet 2 in VPC B does not receive a response to its request to VPC A.

To prevent this, you can add a specific route to the VPC A route table with the CIDR of subnet 2 in VPC B as the destination and a target of `pcx-aaaabbbb`. The new route is more specific, therefore traffic destined for the subnet 2 CIDR is routed to the VPC peering connection `pcx-aaaabbbb`

Alternatively, in the following example, the VPC A route table has a route for each subnet for each VPC peering connection. VPC A can communicate with subnet 2 in VPC B and with subnet 1 in VPC C. This scenario is useful if you need to add another VPC peering connection with another subnet that falls within the same address range as VPC B and VPC C —you can simply add another route for that specific subnet.

Destination	Target
VPC A CIDR	Local
Subnet 2 CIDR	pcx-aaaabbbb
Subnet 1 CIDR	pcx-aaaacccc

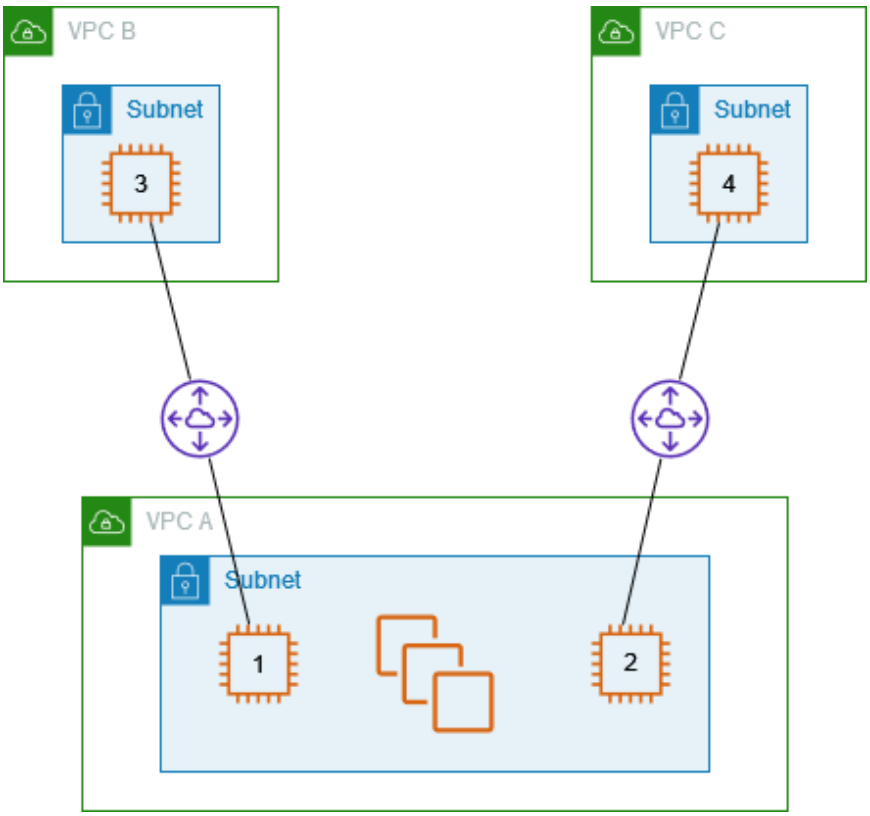
Alternatively, depending on your use case, you can create a route to a specific IP address in VPC B to ensure that traffic routed back to the correct server (the route table uses longest prefix match to prioritize the routes):

Destination	Target
VPC A CIDR	Local
Specific IP address in subnet 2	pcx-aaaabbbb
VPC B CIDR	pcx-aaaacccc

Instances in one VPC that access specific instances in two VPCs

In this configuration, there is a central VPC (VPC A) with one subnet, a peering connection between VPC A and VPC B (`pcx-aaaabbbb`), and a peering connection between VPC A and VPC C (`pcx-`

aaaacccc). VPC A has a subnet with one instance for each peering connection. You can use this configuration to limit peering traffic to specific instances.

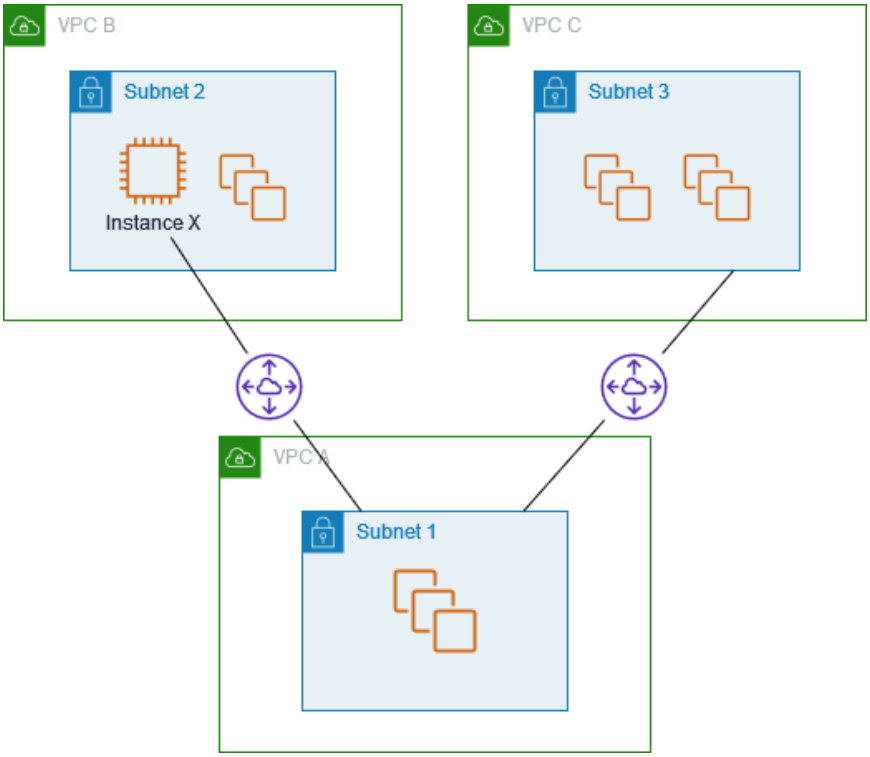


Each VPC route table points to the relevant VPC peering connection to access a single IP address (and therefore a specific instance) in the peer VPC.

Route table	Destination	Target
VPC A	*VPC A CIDR*	Local
	Instance 3 IP address	pcx-aaaabbbb
	Instance 4 IP address	pcx-aaaacccc
VPC B	*VPC B CIDR*	Local
	Instance 1 IP address	pcx-aaaabbbb
VPC C	*VPC C CIDR*	Local
	Instance 2 IP address	pcx-aaaacccc

One VPC that accesses two VPCs using longest prefix matches

In this configuration, there is a central VPC (VPC A) with one subnet, a peering connection between VPC A and VPC B (pcx-aaaabbbb), and a peering connection between VPC A and VPC C (pcx-aaaacccc). VPC B and VPC C have matching CIDR blocks. You use VPC peering connection pcx-aaaabbbb to route traffic between VPC A and a specific instance in VPC B. All other traffic destined for the CIDR address range shared by VPC B and VPC C is routed to VPC C through pcx-aaaacccc.



VPC route tables use longest prefix match to select the most specific route across the intended VPC peering connection. All other traffic is routed through the next matching route, in this case, across the VPC peering connection pcx-aaaacccc.

Route table	Destination	Target
VPC A	*VPC A CIDR block*	Local
	Instance X IP address	pcx-aaaabbbb
	VPC C CIDR block	pcx-aaaacccc
VPC B	*VPC B CIDR block*	Local

Route table	Destination	Target
	VPC A CIDR block	pcx-aaaabbbb
VPC C	*VPC C CIDR block*	Local
	VPC A CIDR block	pcx-aaaacccc

Important

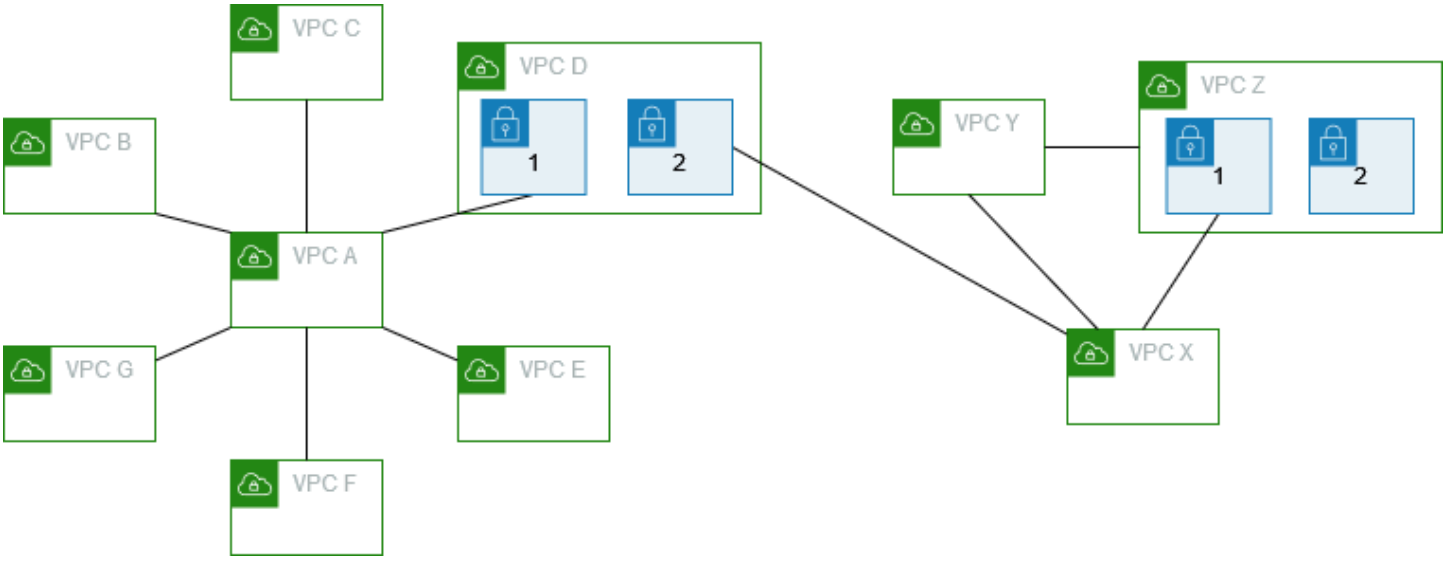
If an instance other than instance X in VPC B sends traffic to VPC A, the response traffic might be routed to VPC C instead of VPC B. For more information, see Routing for response traffic.

Multiple VPC configurations

In this configuration, there is a central VPC (VPC A) is peered with multiple VPCs in a spoke configuration. You also have three VPCs (VPCs X, Y, and Z) peered in a full mesh configuration.

VPC D also has a VPC peering connection with VPC X (`pcx-ddddxxxx`). VPC A and VPC X have overlapping CIDR blocks. This means that peering traffic between VPC A and VPC D is limited to a specific subnet (subnet 1) in VPC D. This is to ensure that if VPC D receives a request from VPC A or VPC X, it sends the response traffic to the correct VPC. AWS does not support unicast reverse path forwarding in VPC peering connections that checks the source IP of packets and routes reply packets back to the source. For more information, see Routing for response traffic.

Similarly, VPC D and VPC Z have overlapping CIDR blocks. Peering traffic between VPC D and VPC X is limited to subnet 2 in VPC D, and peering traffic between VPC X and VPC Z is limited to subnet 1 in VPC Z. This is to ensure that if VPC X receives peering traffic from VPC D or VPC Z, it sends the response traffic back to the correct VPC.



The route tables for VPCs B, C, E, F, and G point to the relevant peering connections to access the full CIDR block for VPC A, and the VPC A route table points to the relevant peering connections for VPCs B, C, E, F, and G to access their full CIDR blocks. For peering connection `pcx-aaaadddd`, the VPC A route table routes traffic only to subnet 1 in VPC D and the subnet 1 route table in VPC D points to the full CIDR block of VPC A.

The VPC Y route table points to the relevant peering connections to access the full CIDR blocks of VPC X and VPC Z, and the VPC Z route table points to the relevant peering connection to access the full CIDR block of VPC Y. The subnet 1 route table in VPC Z points to the relevant peering connection to access the full CIDR block of VPC Y. The VPC X route table points to the relevant peering connection to access subnet 2 in VPC D and subnet 1 in VPC Z.

Route table	Destination	Target
VPC A	*VPC A CIDR*	Local
	VPC B CIDR	pcx-aaaabbbb
	VPC C CIDR	pcx-aaaacccc
	Subnet 1 CIDR in VPC D	pcx-aaaadddd
	VPC E CIDR	pcx-aaaaeeee
	VPC F CIDR	pcx-aaaaffff

Route table	Destination	Target
	VPC G CIDR	pcx-aaaagggg
VPC B	*VPC B CIDR*	Local
	VPC A CIDR	pcx-aaaabbbb
VPC C	*VPC C CIDR*	Local
	VPC A CIDR	pcx-aaaacccc
Subnet 1 in VPC D	*VPC D CIDR*	Local
	VPC A CIDR	pcx-aaaadddd
Subnet 2 in VPC D	*VPC D CIDR*	Local
	VPC X CIDR	pcx-ddddxxxx
VPC E	*VPC E CIDR*	Local
	VPC A CIDR	pcx-aaaaeeee
VPC F	*VPC F CIDR*	Local
	VPC A CIDR	pcx-aaaaffff
VPC G	*VPC G CIDR*	Local
	VPC A CIDR	pcx-aaaagggg
VPC X	*VPC X CIDR*	Local
	Subnet 2 CIDR in VPC D	pcx-ddddxxxx
	VPC Y CIDR	pcx-xxxxyyyy
	Subnet 1 CIDR in VPC Z	pcx-xxxxzzzz
VPC Y	*VPC Y CIDR*	Local

Route table	Destination	Target
	VPC X CIDR	pcx-xxxxyyyy
	VPC Z CIDR	pcx-yyyyzzzz
VPC Z	*VPC Z CIDR*	Local
	VPC Y CIDR	pcx-yyyyzzzz
	VPC X CIDR	pcx-xxxxzzzz

VPC peering connection networking scenarios

There are a number of reasons you might need to set up a VPC peering connection between your VPCs, or between a VPC that you own and a VPC in a different AWS account. The following scenarios can help you determine which configuration is best suited to your networking requirements.

Scenarios

- Peering two or more VPCs to provide full access to resources
- Peering to one VPC to access centralized resources

Peering two or more VPCs to provide full access to resources

In this scenario, you have two or more VPCs that you want to peer to enable full sharing of resources between all VPCs. The following are some examples:

- Your company has a VPC for the finance department, and another VPC for the accounting department. The finance department requires access to all resources that are in the accounting department, and the accounting department requires access to all resources in the finance department.
- Your company has multiple IT departments, each with their own VPC. Some VPCs are located within the same AWS account, and others in a different AWS account. You want to peer together all VPCs to enable the IT departments to have full access to each others' resources.

For more information about how to set up the VPC peering connection configuration and route tables for this scenario, see the following documentation:

- Two VPCs peered together
- Three VPCs peered together
- Multiple VPCs peered together

For more information about creating and working with VPC peering connections in the Amazon VPC console, see VPC peering connections.

Peering to one VPC to access centralized resources

In this scenario, you have a central VPC that contains resources that you want to share with other VPCs. Your central VPC may require full or partial access to the peer VPCs, and similarly, the peer VPCs may require full or partial access to the central VPC. The following are some examples:

- Your company's IT department has a VPC for file sharing. You want to peer other VPCs to that central VPC, however, you do not want the other VPCs to send traffic to each other.
- Your company has a VPC that you want to share with your customers. Each customer can create a VPC peering connection with your VPC, however, your customers cannot route traffic to other VPCs that are peered to yours, nor are they aware of the other customers' routes.
- You have a central VPC that is used for Active Directory services. Specific instances in peer VPCs send requests to the Active Directory servers and require full access to the central VPC. The central VPC does not require full access to the peer VPCs; it only needs to route response traffic to the specific instances.

For more information about creating and working with VPC peering connections in the Amazon VPC console, see VPC peering connections.

Identity and access management for VPC peering

By default, users cannot create or modify VPC peering connections. To grant access to VPC peering resources, attach an IAM policy to an IAM identity, such as a role.

Examples

- Example: Create a VPC peering connection
- Example: Accept a VPC peering connection
- Example: Delete a VPC peering connection
- Example: Work within a specific account
- Example: Manage VPC peering connections using the console

For a list of Amazon VPC actions, and the supported resources and conditions keys for each action, see Actions, resources, and condition keys for Amazon EC2 in the *Service Authorization Reference*.

Example: Create a VPC peering connection

The following policy grants users permission to create VPC peering connection requests using VPCs that are tagged with `Purpose=Peering`. The first statement applies a condition key (`ec2:ResourceTag`) to the VPC resource. Note that the VPC resource for the `CreateVpcPeeringConnection` action is always the requester VPC.

The second statement grants users permission to create the VPC peering connection resources, and therefore uses the * wildcard in place of a specific resource ID.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action": "ec2:CreateVpcPeeringConnection",
      "Resource": "arn:aws:ec2:us-east-1:123456789012:vpc/*",
      "Condition": {
```

```
        "StringEquals": {
          "ec2:ResourceTag/Purpose": "Peering"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateVpcPeeringConnection",
      "Resource": "arn:aws:ec2:us-east-1:123456789012:vpc-peering-connection/*"
    }
  ]
}
```

The following policy grants users in the specified AWS account permission to create VPC peering connections using any VPC in the specified Region, but only if the VPC that accepts the peering connection is a specific VPC in specific account.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:CreateVpcPeeringConnection",
            "Resource": "arn:aws:ec2:us-east-1:123456789012:vpc/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateVpcPeeringConnection",
            "Resource": "arn:aws:ec2:us-east-1:123456789012:vpc-peering-
connection/*",
            "Condition": {
                "ArnEquals": {
                    "ec2:AccepterVpc": "arn:aws:ec2:us-
east-1:111122223333:vpc/vpc-1234567890abcdef0"
                }
            }
        }
    ]
}
```

Example: Accept a VPC peering connection

The following policy grants users permission to accept VPC peering connection requests from a specific AWS account. This helps to prevent users from accepting VPC peering connection requests from unknown accounts. The statement uses the `ec2:RequesterVpc` condition key to enforce this.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action": "ec2:AcceptVpcPeeringConnection",
      "Resource": "arn:aws:ec2:us-east-1:123456789012:vpc-peering-connection/*",
      "Condition": {
        "ArnEquals": {
          "ec2:RequesterVpc": "arn:aws:ec2:us-east-1:111122223333:vpc/*"
        }
      }
    },
    {
      "Effect":"Allow",
      "Action": "ec2:AcceptVpcPeeringConnection",
      "Resource": "arn:aws:ec2:us-east-1:123456789012:vpc/*"
    }
  ]
}
```

The following policy grants users permission to accept VPC peering requests if the VPC has the tag `Purpose=Peering`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
```

```
      "Effect":"Allow",
      "Action": "ec2:AcceptVpcPeeringConnection",
      "Resource": "arn:aws:ec2:us-east-1:123456789012:vpc-peering-connection/*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:AcceptVpcPeeringConnection",
      "Resource": "arn:aws:ec2:us-east-1:123456789012:vpc/*",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/Purpose": "Peering"
        }
      }
    }
  ]
}
```

Example: Delete a VPC peering connection

The following policy grants users in the specified account permission to delete any VPC peering connection, except those that use the specified VPC, which is in the same account. The policy specifies both the `ec2:AccepterVpc` and `ec2:RequesterVpc` condition keys, as the VPC might have been the requester VPC or the peer VPC in the original VPC peering connection request.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteVpcPeeringConnection",
            "Resource": "arn:aws:ec2:us-east-1:123456789012:vpc-peering-
connection/*",
            "Condition": {
                "ArnNotEquals": {
                    "ec2:AccepterVpc": "arn:aws:ec2:us-
east-1:123456789012:vpc/vpc-1234567890abcdef0",
                    "ec2:RequesterVpc": "arn:aws:ec2:us-
east-1:123456789012:vpc/vpc-0abcdef1234567890"
                }
```

```
            }
        }
    ]
}
```

Example: Work within a specific account

The following policy grants users permission to work with VPC peering connections within a specific account. Users can view, create, accept, reject, and delete VPC peering connections, provided they are all within the same AWS account.

The first statement grants users permission to view all VPC peering connections. The `Resource` element requires a * wildcard in this case, as this API action (`DescribeVpcPeeringConnections`) currently does not support resource-level permissions.

The second statement grants users permission to create VPC peering connections, and access to all VPCs in the specified account in order to do so.

The third statement uses a * wildcard as part of the `Action` element to grant permission for all VPC peering connection actions. The condition keys ensure that the actions can only be performed on VPC peering connections with VPCs that are part of the account. For example, a user is cannot delete a VPC peering connection if either the accepter or requester VPC is in a different account. A user cannot create a VPC peering connection with a VPC in a different account.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeVpcPeeringConnections",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcPeeringConnection",
                "ec2:AcceptVpcPeeringConnection"
            ],
```

```
            "Resource": "arn:aws:ec2:*:111122223333:vpc/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:*VpcPeeringConnection",
            "Resource": "arn:aws:ec2:*:111122223333:vpc-peering-connection/*",
            "Condition": {
                "ArnEquals": {
                    "ec2:AccepterVpc": "arn:aws:ec2:*:111122223333:vpc/*",
                    "ec2:RequesterVpc": "arn:aws:ec2:*:111122223333:vpc/*"
                }
            }
        }
    ]
}
```

Example: Manage VPC peering connections using the console

To view VPC peering connections in the Amazon VPC console, users must have permission to use the `ec2:DescribeVpcPeeringConnections` action. To use the **Create Peering Connection** page, users must have permission to use the `ec2:DescribeVpcs` action. This grants them permission to view and select a VPC. You can apply resource-level permissions to all the `ec2:*PeeringConnection` actions, except `ec2:DescribeVpcPeeringConnections`.

The following policy grants users permission to view VPC peering connections, and to use the **Create VPC Peering Connection** dialog box to create a VPC peering connection using a specific requester VPC only. If users try to create a VPC peering connection with a different requester VPC, the request fails.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect":"Allow",
      "Action": [
        "ec2:DescribeVpcPeeringConnections", "ec2:DescribeVpcs"
      ],
      "Resource": "*"
```

```
    },
    {
      "Effect":"Allow",
      "Action": "ec2:CreateVpcPeeringConnection",
      "Resource": [
        "arn:aws:ec2:*:*:vpc/vpc-1234567890abcdef0",
        "arn:aws:ec2:*:*:vpc-peering-connection/*"
      ]
    }
  ]
}
```

VPC peering connection quotas for an account

VPC peering allows you to connect two VPCs. This enables resources in one VPC to communicate with resources in the other VPC as if they were in the same network. VPC peering is a useful feature for connecting your VPCs, whether they are in the same AWS Region or different Regions. This section describes the quotas you should be aware of when working with VPC peering connections.

The following table lists the quotas, formerly referred to as limits, for VPC peering connections for your AWS account. Unless indicated otherwise, you can request an increase for these quotas.

If you find that your current VPC peering connection requirements exceed the default quotas, we encourage you to submit a service limit increase request. We will review your use case and work with you to adjust the quotas accordingly, ensuring your VPC environment can support your growing business needs.

Name	Default	Adjustable
Active VPC peering connections per VPC	50	Yes (up to 125)
Outstanding VPC peering connection requests	25	Yes
Expiry time for an unaccepted VPC peering connection request	1 week (168 hours)	No

For more information about the rules for using VPC peering connections, see VPC peering limitations. For additional information about quotas for Amazon VPC, see Amazon VPC quotas in the *Amazon VPC User Guide*.

Document history for the Amazon VPC Peering Guide

The following table describes the documentation releases for the *Amazon VPC Peering Guide*.

Change	Description	Date
Tag on create	You can add tags when you create a VPC peering connection and route table.	July 20, 2020
Inter-Region peering	DNS hostname resolution is supported for inter-Region VPC peering connections in the Asia Pacific (Hong Kong) Region.	August 26, 2019
Inter-Region peering	You can create a VPC peering connection between VPCs in different AWS Regions.	November 29, 2017
DNS resolution support for VPC peering	You can enable a local VPC to resolve public DNS hostnames to private IP addresses when queried from instances in the peer VPC.	July 28, 2016
Stale security group rules	You can identify if your security group is being referenced in the rules of a security group in a peer VPC and you can identify stale security group rules.	May 12, 2016
Using ClassicLink over a VPC peering connection	You can modify your VPC peering connection to enable local linked EC2-Classic instances to communicate	April 26, 2016

	with instances in a peer VPC, or vice versa.	
VPC peering	You can create a VPC peering connection between two VPCs, which allows instances in either VPC to communicate with each other using private IP addresses	March 24, 2014