About Cisco Cloud Network Controller and Google Cloud

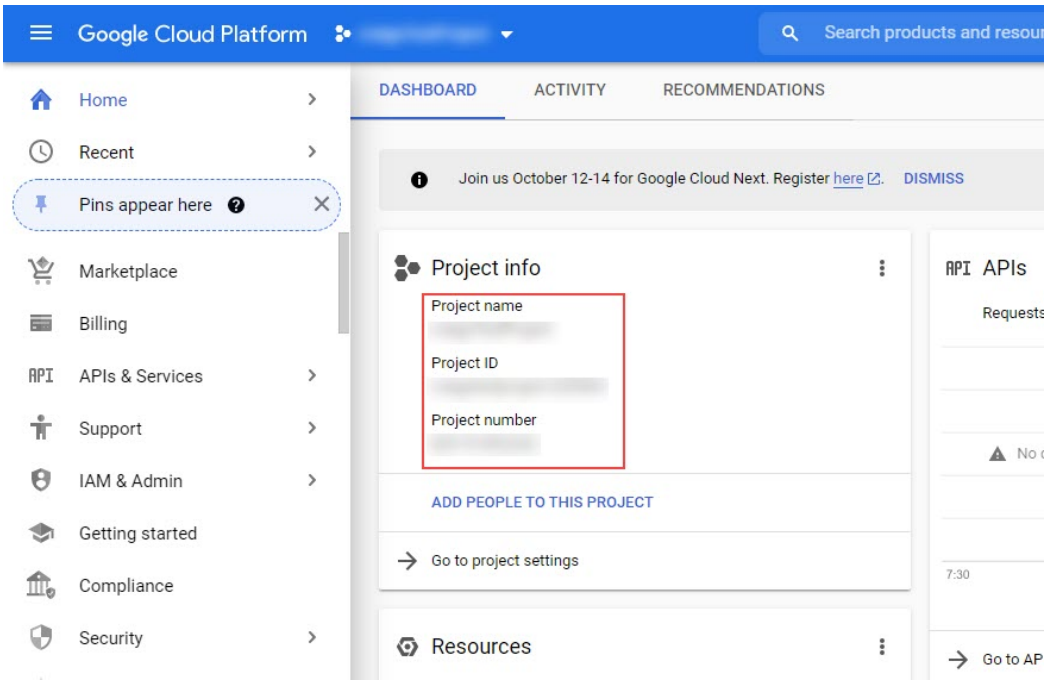
The following topics in this chapter provide information on how Cisco Cloud Network Controller deployments work with Google Cloud.

Locating Important Google Cloud Project Information

After you create a Google Cloud project, that project will be assigned three unique identifiers:

- Project name
- Project ID
- Project number

You will need these three identifiers for your Google Cloud project at various points in the Cisco Cloud Network Controller configuration process. To locate the **Project Info** pane with these Google Cloud project identifiers, log into your Google Cloud account and select your particular Google Cloud project in the **Select a project** window. The **Dashboard** for this project is displayed, which provides the Project Info pane with these three unique identifiers for your Google Cloud project.



Understanding Google Cloud Deployments with Cisco Cloud Network Controller

Google Cloud organizes resources in a way that resembles a file system, where:

- The *Organization* at the top level can have multiple *Folders*.
- Every *Folder* can contain other *Folders*, or can contain *Projects*, where every *Project* has a unique ID.
- Cloud *resources* (such as VMs, VPCs, and subnets) are contained within a *Project*.

While the Organization and Folder levels are useful areas to understand from the Google Cloud perspective, the Project level is the most relevant from the Cisco Cloud Network Controller perspective.

Each Cisco Cloud Network Controller tenant is mapped one-to-one to a Google Cloud Project, which means that:

- A Cisco Cloud Network Controller tenant cannot span multiple Google Cloud Projects
- There cannot be more than one Cisco Cloud Network Controller tenant in a Google Cloud Project

With Cisco Cloud Network Controller, Google Cloud provides access to Projects using **Service Accounts**. These accounts are meant for applications that need to access Google Cloud services. They can be used to run and deploy Cisco Cloud Network Controller and to push policies for other tenants. Service accounts used in applications running within Google Cloud do not need credentials, whereas applications that are run external to Google Cloud need a pre-generated private key. Service Accounts reside in one Google Cloud Project, but they can also be given access to manage policies for other Projects (for Cisco Cloud Network Controller, other tenants).

The following sections provide more information on different ways that Cisco Cloud Network Controller tenants can be configured with Google Cloud:

- User Tenants With Managed Credentials, on page 3
- User Tenants With Unmanaged Credentials, on page 3

User Tenants With Managed Credentials

This type of user tenant has the following characteristics:

- This tenant account is managed by the Cisco Cloud Network Controller.
- You will first choose **Managed Identity** in the Cisco Cloud Network Controller GUI as part of the tenant configuration process for this type of user tenant.
- After you have configured the necessary parameters in the Cisco Cloud Network Controller, you must then set the necessary roles for this tenant in Google Cloud. Add the service account created by the Cisco Cloud Network Controller as an IAM user with the following rules:
  - Cloud Functions Service Agent
  - Compute Instance Admin (v1)
  - Compute Network Admin
  - Compute Security Admin
  - Logging Admin
  - Pub/Sub Admin
  - Storage Admin

For instructions on creating this sort of tenant, see Creating a Managed Tenant Using the Cisco Cloud Network Controller GUI.

User Tenants With Unmanaged Credentials

This type of user tenant has the following characteristics:

- This tenant account is not managed by the Cisco Cloud Network Controller.
- Before configuring the necessary parameters in the Cisco Cloud Network Controller for this type of tenant, you must first download the JSON file that contains the necessary private key information from Google Cloud for the service account associated with this tenant.
- You will then choose **Unmanaged Identity** in the Cisco Cloud Network Controller GUI as part of the tenant configuration process for this type of user tenant. As part of the configuration process for this type of tenant in Cisco Cloud Network Controller, you will provide the following information from the downloaded JSON file:
  - Key ID
  - RSA Private Key
  - Client ID
  - Email

For instructions on creating this sort of tenant, see Creating an Unmanaged Tenant Using the Cisco Cloud Network Controller GUI.

External Network Connectivity Using Cloud Native Routers

Support is available for external connectivity between a Google Cloud site and non-Google Cloud sites or an external device. You can have this IPv4 connection by creating a VPN connection between a Google Cloud router and an external device.

The following sections provide more information on the components that allow for the new external network connectivity:

External VRF

An **external VRF** is a unique VRF that does not have any presence in the cloud. This VRF is not referred to in any cloud context profile used by Cisco Cloud Network Controller.

An external VRF represents an external network that is connected to other cloud sites or to on-premises sites. Multiple cloud VRFs can leak routes to an external VRF or can get the routes from an external VRF. When an external network is created on an external VRF, inter-VRF routing is set up so that routes received and advertised on the external network are received or advertised on the external VRF.

Cloud Native Routers

When configuring Cisco Cloud Network Controller with Google Cloud, the infra VPC uses Google Cloud native routers (Cloud Router and Cloud VPN gateway) to create IPsec tunnels and BGP sessions to on-premises sites, other cloud sites, or any remote device. Only IPv4 connectivity is supported for this type of connectivity using cloud native routers, where IPv4 sessions are created on an external VRF.

Google Cloud supports VPN connections both with static routes and with BGP. To create a VPN connection with BGP, Cisco Cloud Network Controller needs both a Cloud Router and a VPN gateway. A VPC can have multiple Cloud Routers and VPN gateways. However, Google Cloud has a restriction that both the Cloud Routers and the VPN gateways must be in the same region and in the same VPC. In addition, Cisco Cloud Network Controller has a restriction where only one cloud router and one cloud VPN gateway is supported per region.

VPN Communication

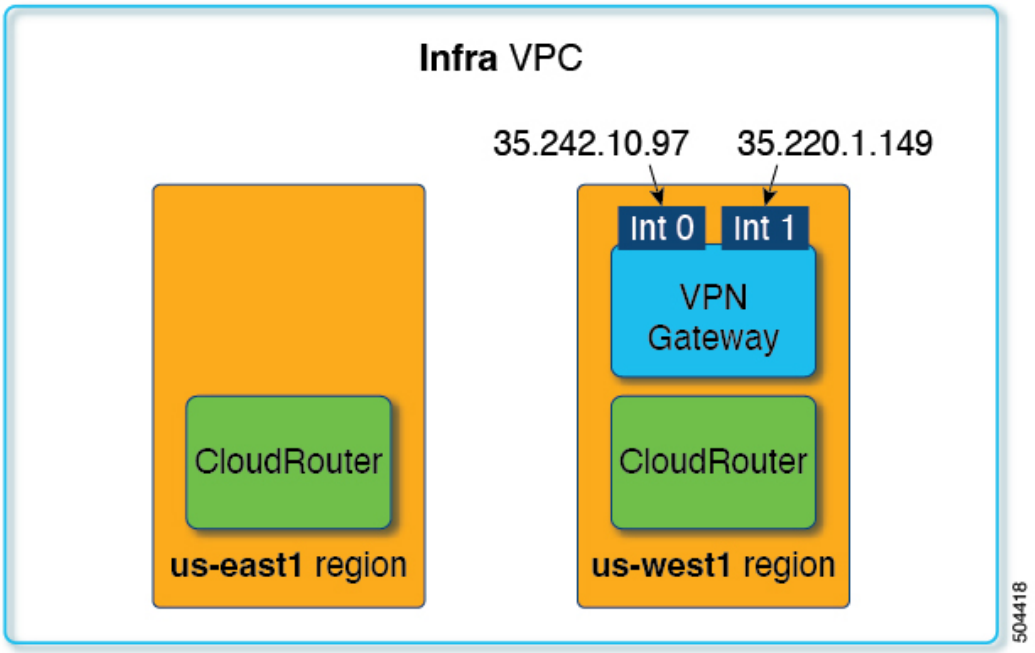
When configuring Cisco Cloud Network Controller with Google Cloud, the infra VPC is used to host the Cisco Cloud Network Controller and to host the VPN connections to external devices and sites. However, the infra VPC is not used as a transit to implement spoke-to-spoke communication. Instead, when configuring Cisco Cloud Network Controller with Google Cloud, spoke-to-spoke communication is done though spoke-to-spoke VPC peering.

The infra VPC uses the Google Cloud Router and Google Cloud VPN Gateway to create IPsec tunnels and BGP sessions to on-premises sites or to other cloud sites. Spoke VPCs peer with the infra VPC to share the VPN connections to external sites, where:

- Routes received on the VPN connections are leaked to the spoke VPCs
- Spoke VPC routes are advertised on the VPN connections

Using inter-VRF routing, the route is leaked between the external VRF of the VPN connections and the cloud local spoke VRFs.

A VPN gateway has two interfaces, and Google Cloud allocates public IP addresses to each of the interfaces. While the Google Cloud VPN gateway could have one or two interfaces, Cisco Cloud Network Controller only supports VPN gateways with two interfaces because two interfaces are required to achieve high availability.



Hub Network Configuration

Rather than creating the hub network in a region based on the spoke attachments, the `cloudRegionName` MOs under a `cloudtemplateHubNetworkName` represents the regions where the hub network will be deployed, where `cloudtemplateHubNetworkName` represents a Google Cloud Router. Cisco Cloud Network Controller has a restriction of only one `cloudtemplateHubNetworkName`.

The hub network provides a way for establishing connectivity to an external site. Creating a hub network is a pre-requisite to creating an external network. You can create a hub network by specifying a name for the hub and the regions where the hub network should be deployed. For example, you may choose to deploy the hub network in us-central1 and us-east1. Cisco Cloud Network Controller will provision the Google Cloud Routers in these regions. Remember that only one hub network can be created, which means that Cisco Cloud Network Controller will only deploy one Cloud Router per region.

The following POST shows an example of network connectivity using this model. The `cloudtemplateHubNetwork` is used to create the hub network. In this example, the hub network is deployed

in four regions. External networks are created from each of the four regions using the `cloudtemplateExtNetwork` MOs.

```
<polUni>
    <fvTenant name="infra" status="">
        <fvCtx name="extv1" pcEnfPref="enforced" status=""/>
         <fvCtx name="extv2" pcEnfPref="enforced" status=""/>
          <fvCtx name="extv3" pcEnfPref="enforced" status=""/>

           <cloudtemplateInfraNetwork name="default" vrfName="overlay-1"
hostRouterMode="manual" status="">
          <cloudtemplateIpSecTunnelSubnetPool subnetpool= "169.254.7.0/24" poolname="pool1"
 />
           <cloudtemplateIpSecTunnelSubnetPool subnetpool= "169.254.8.0/24" poolname="pool2"
 />
              <cloudtemplateIpSecTunnelSubnetPool subnetpool= "169.254.10.0/24"
poolname="pool3" />

            <cloudtemplateHubNetwork name="default" status="" >
                <cloudtemplateHubNetworkName name="foo1" asn="64514" status="">
                    <cloudRegionName provider="gcp" region="us-west4" status="" />
                    <cloudRegionName provider="gcp" region="us-west2" status="" />
                    <cloudRegionName provider="gcp" region="us-east1" status="" />
                    <cloudRegionName provider="gcp" region="us-west1" status=""/>
                </cloudtemplateHubNetworkName>
            </cloudtemplateHubNetwork>

            <cloudtemplateIntNetwork name="default">
                <cloudRegionName provider="gcp" region="us-west1">
                    <cloudtemplateVpnRouter name="default" status=""/>
                </cloudRegionName>
                <cloudRegionName provider="gcp" region="us-west2">
                    <cloudtemplateVpnRouter name="default" status=""/>
                </cloudRegionName>
                <cloudRegionName provider="gcp" region="us-east1">
                    <cloudtemplateVpnRouter name="default" status=""/>
                  </cloudRegionName>
                    <cloudRegionName provider="gcp" region="us-west4">
                    <cloudtemplateVpnRouter name="default" status=""/>
                </cloudRegionName>
            </cloudtemplateIntNetwork>

            <cloudtemplateExtNetwork name="default">
            </cloudtemplateExtNetwork>
             <cloudtemplateExtNetwork name="extnwfoo1" vrfName="extv1" hubNetworkName="foo1"
 vpnRouterName="default" status="">
                    <cloudRegionName provider="gcp" region="us-west1" status=""/>
                      <cloudtemplateVpnNetwork name="onprem01" remoteSiteId="1" status="">

                    <cloudtemplateIpSecTunnel peeraddr="128.1.1.1" preSharedKey="abcd"
poolname="pool1" status="">
                     <cloudtemplateBgpIpv4 peeraddr="0.0.0.0/0" peerasn="64529" status=""/>

                    </cloudtemplateIpSecTunnel>
                </cloudtemplateVpnNetwork>
            </cloudtemplateExtNetwork>
           <cloudtemplateExtNetwork name="extnwfoo2" vrfName="extv2" hubNetworkName="foo1"
 vpnRouterName="default" status="">
             <cloudRegionName provider="gcp" region="us-west2" status=""/>
                <cloudtemplateVpnNetwork name="onprem02" remoteSiteId="2" status="">
                       <cloudtemplateIpSecTunnel peeraddr="128.1.1.2" preSharedKey="def"
poolname="pool2" status="">
                     <cloudtemplateBgpIpv4 peeraddr="0.0.0.0/0" peerasn="64529" status=""/>
```

```
                    </cloudtemplateIpSecTunnel>
                </cloudtemplateVpnNetwork>
            </cloudtemplateExtNetwork>
           <cloudtemplateExtNetwork name="extnwfoo3" vrfName="extv3" hubNetworkName="foo1"
 vpnRouterName="default" status="">
                <cloudRegionName provider="gcp" region="us-east1" status=""/>
                   <cloudtemplateVpnNetwork name="onprem03" remoteSiteId="3" status="">
                       <cloudtemplateIpSecTunnel peeraddr="128.1.1.3" preSharedKey="abc"
poolname="pool3" status="">
                    <cloudtemplateBgpIpv4 peeraddr="0.0.0.0/0" peerasn="64529" status=""/>

                    </cloudtemplateIpSecTunnel>
                </cloudtemplateVpnNetwork>
                  </cloudtemplateExtNetwork>
        </cloudtemplateInfraNetwork>
    </fvTenant>
</polUni>
```

In this example POST:

- **cloudtemplateExtNetwork**: You can have multiple `cloudtemplateExtNetwork` entries, each with a unique name, that represent an external network on an external VRF.

  Within the `cloudtemplateExtNetwork` area are the following fields:

  - **vrfName**: This property represents the VRF used for the external network (for example, a transport VRF). Multiple remote sites can use the same transport VRF, which means that all of these remotes sites are treated as one VRF on the cloud and all of the remote sites receive the same routes from the cloud.
  - **hubNetworkName**: This property represents the name of the hub network used by this external network. This name refers to one of the hub networks created in the `cloudtemplateHubNetworkName` area.
  - **vpnRouterName**: This property represents the name of the VPN router used by this external network. This name refers to the VPN router created by `cloudtemplateVpnRouter`.

  In addition, an external network can be deployed in multiple regions, and a router used on the external network should be deployed in those regions (in other words, `hubNetworkName` and `vpnRouterName` should exist in those regions).

- **cloudtemplateVpnNetwork**: This MO represents a remote site.

  Within the `cloudtemplateVpnNetwork` area is the **remoteSiteId** field. This property represents the remote site ID.

- **cloudtemplateVpnRouter**: This MO translates to a Google Cloud VPN gateway. Only one `cloudtemplateVpnRouter` is allowed, with the name `default`.
- **cloudtemplateIpSecTunnel**: This MO represents a remote peer.
- **cloudtemplateBgpIpv4**: This MO represents a remote site IPv4 BGP peer.

  If the `peeraddr` entry under `cloudtemplateBgpIpv4` has the default address (0.0.0.0/0), then the remote BGP peer is assumed to be the inner address of the tunnel on the remote device.

Note that the model above supports the following:

- Both ikev1 and ikev2 to an external device.

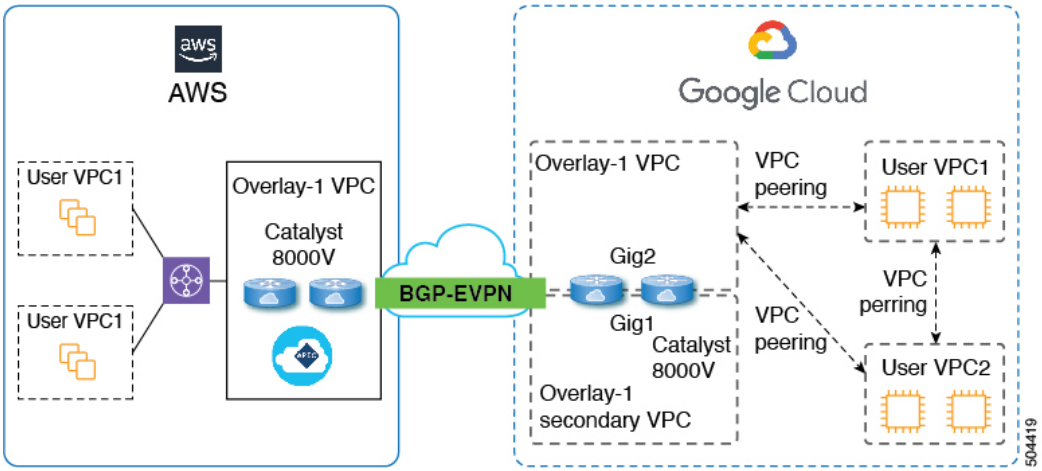
- Multiple `cloudtemplateIpSecTunnelSubnetPool` subnet pools. The allowed IP ranges in the `cloudtemplateIpSecTunnelSubnetPool` subnet pools is dependent on the cloud provider and use case. For example, 169.254.0.0/16 or a lesser subnet of it is supported for Google Cloud VPN connections.

Inter-Site Connectivity Using BGP-EVPN

For inter-site use cases, support is available for configuring a BGP-EVPN connection for inter-site connectivity in these scenarios:

- Cloud site-to-cloud site:
  - Google Cloud site-to-Google Cloud site
  - Google Cloud site-to-AWS site
  - Google Cloud site-to-Azure site
- Google Cloud site-to-ACI on-premises site

In each of these scenarios, Cisco Catalyst 8000Vs are used for the BGP-EVPN connection.



The following sections provide more information on the components that allow for inter-site connectivity using BGP-EVPN:

- Characteristics of Inter-Site Connectivity Using BGP-EVPN, on page 8
- VPC Peering, on page 9

Characteristics of Inter-Site Connectivity Using BGP-EVPN

Based on Google Cloud behavior, each network interface of a VM or instance must be associated with a different VPC. Because the Cisco Catalyst 8000V is also a VM, this means that each network interface for a given Cisco Catalyst 8000V has to be associated with a different VPC. Two gigabit network interfaces in the Cisco Catalyst 8000V are therefore used in the following ways:

- The gig1 interface is associated with the overlay-1 secondary VPC. In addition, the gig1 interface is used as the management interface.

- The gig2 interface is associated with the overlay-1 VPC. In addition, the gig2 interface is used as the routing interface.

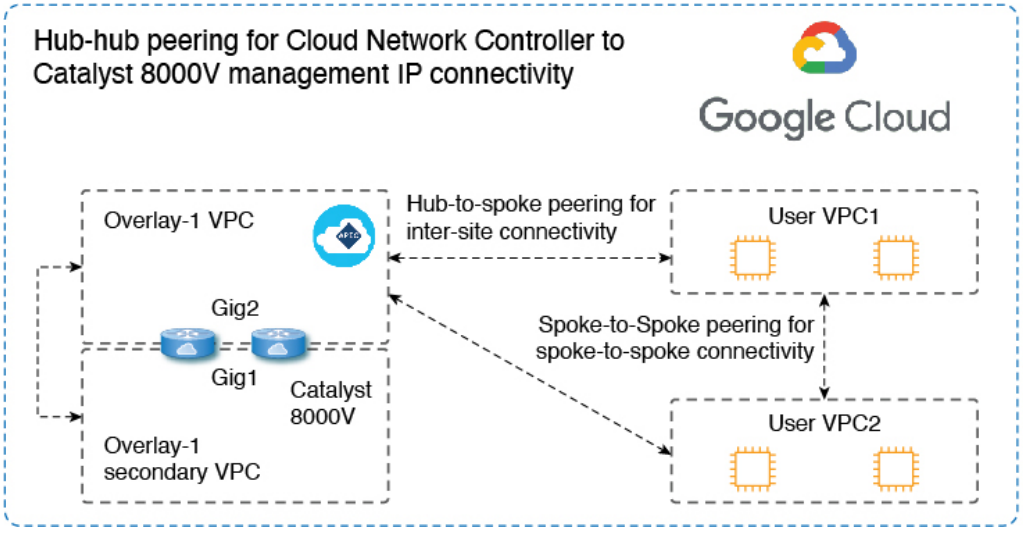
VPC Peering

In order to have communcation from the spoke VPC to an on-premises network, the spoke VPC must have peering enabled to the hub VPCs. The peering is automated by intent from Cisco Cloud Network Controller. VPC peering for Cisco Cloud Network Controller with Google Cloud employs a hub-spoke topology, as shown in the following figure.

Cisco Cloud Network Controller with Google Cloud uses three types of VPC peering:

- Spoke-to-spoke VPC peering: This is used for spoke-to-spoke intra-site communication.
- Hub-to-spoke VPC peering: This is used for inter-site communication that goes through the Cisco Catalyst 8000V routers using BGP-EVPN.
- Hub-to-hub VPC peering: This is used for communication between the Cisco Cloud Network Controller in the overlay-1 VPC and the Cisco Catalyst 8000V routers management interfaces in the overlay-1 secondary VPC.

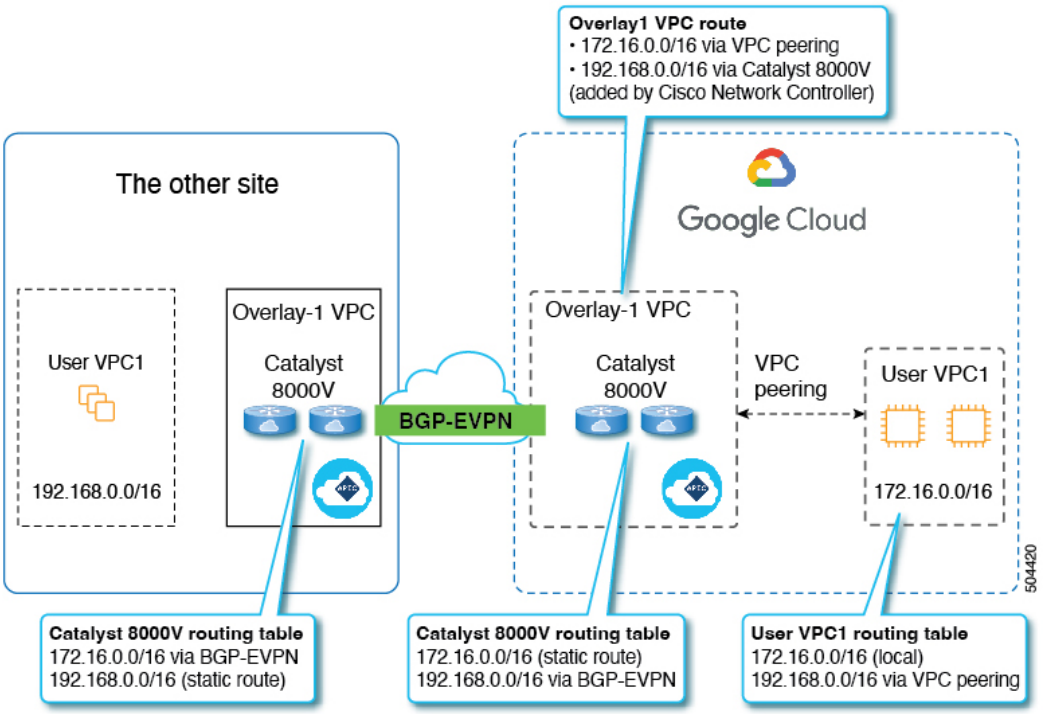
Note that the overlay-1 secondary VPC is not involved in the data path for either spoke-to-spoke or inter-site traffic.



Cisco Cloud Network Controller automates configurations to exchange the routes between cloud sites in the following situations:

- Overlay-1 VPC to the destination in the same site: The overlay-1 VPC has the route to the spoke VPC in the same site through VPC peering.
- Spoke VPCs to the destination in another site: The routes for the subnets in the other site are added to the overlay-1 VPC by Cisco Cloud Network Controller and the routes are exported to the spoke VPCs. In this way, the spoke VPCs have the routes to reach the destination subnets in the other site.
- Between Cisco Catalyst 8000Vs in different sites: The static route for the spoke VPC CIDRs are added to the Cisco Catalyst 8000V routers in the same site. The static routes are redistributed to the Catalyst

8000V routers in the other site through BGP EVPN. In this way, the Catalyst 8000Vs have the routes to reach the destination subnets in the other site, as shown in the following figure.



In this scenario, a static route to the remote CIDR is programmed in the hub VPC with the next hop as the Cisco Catalyst 8000V. These routes are learned by the spoke VPC using peering.

Configuring Routing and Security Policies Separately

To allow communication between two endpoints in different VRFs, you need to establish routing and security policies separately:

- **Routing policies**: Policies used to define routes to establish traffic flow
- **Security policies**: Rules used for security purposes, such as zoning rules, security-group rules, ACLs, and so on

For Google Cloud, routing must be configured independent of security. In other words, for Google Cloud, "contracts" are used only for security. To configure routing, you must configure route-maps.

Configuring Routing Policies

Using inter-VRF routing, you can configure an independent routing policy to specify which routes to leak between a pair of VRFs. To establish routing, you must configure route maps between a pair of VRFs.

For situations where you can use route maps to set which routes to leak between a pair of VRFs, the following types of VRFs are used for inter-VRF routing:

- **External VRF** is a VRF that is associated with one or more external networks.

- **Internal VRF** is a VRF that has one or more cloud context profiles or cloud subnets associated with it.

When configuring inter-VRF routing with these types of VRFs:

- Between a pair of internal VRFs, you must always leak all routes.
- From an internal VRF to an external VRF, you can leak specific routes or all routes.
- From an external VRF to an internal VRF, you must leak all routes.

Guidelines and Restrictions

The following guidelines apply when using inter-VRF routing to leak routes between a pair of VRFs using route maps:

- Routes are always leaked bi-directionally between two VRFs. For every route leak entry from one tenant/VRF under another tenant/VRF, there must be a corresponding route leak entry going in the opposite direction.

  For example, assume there are two tenants (`t1` and `t2`) and two corresponding VRFs (`v1` and `v2`). For every route leak entry `t1:v1` under the VRF `t2:v2`, there must be a corresponding route leak entry `t2:v2` under the VRF `t1:v1`.

- Once you associate an external VRF with an external network, if you want to change the external VRF, you need to delete the external network and then recreate the external network with the new external VRF.
- You cannot configure "smaller" prefixes to be leaked while a "larger" prefix is already being leaked. For example, configuring the 10.10.10.0/24 prefix will be rejected if you already have the 10.10.0.0/16 prefix configured to be leaked. Similarly, if you configure the 0.0.0.0/0 (leak all) prefix, no other prefix will be allowed to be configured.

Configuring Security Policies

While an EPG in Cisco Cloud Network Controller corresponds to security groups in AWS and Azure, there is no equivalent corresponding component in Google Cloud for an EPG. The closest equivalent in Google Cloud is a combination of firewall rules and network tags.

The firewall resource in Google Cloud is global to the project (tenant). Firewall rules are associated with a single VPC and their scope applies to the entire VPC globally. The scope of the firewall rule is further defined by the Target parameter. In other words, the set of instances that a rule is applied to can be selected by one or more of the following Target types:

- **Network tags**: Network tags are key strings that drive the VM's firewall and routing configuration on Google Cloud. Instances (for example, VMs) can be tagged with unique strings. Firewall rules are applied to all instances with equal tags. Multiple tag values act as a logical 'or' operator, where the firewall rule is applied as long as at least one tag matches.
- **All instances in the network**: The firewall rule applies to all instances in the VPC.

Firewall rules also identify the source and destination of the traffic. Depending on whether the rule is for ingress traffic (going to a VM) or egress traffic (leaving a VM), the source and destination fields accept different values. The following list provides more information on those values:

- **Ingress rules**:

- **Source**: Can be identified using:
  - Network tags
  - IP addresses
  - A combination of IP addresses and network tags with a logical ‘or’ operator
- **Destination**: The Target parameter identifies the destination instances

- **Egress rules**:
  - **Source**: The Target parameter identifies the source instances
  - **Destination**: Can be identified using only IP addresses (not network tags)

How Cisco Cloud Network Controller Implements Firewall Rules With Google Cloud

The following list describes how Cisco Cloud Network Controller implements firewall rules with Google Cloud:

- **Global resources**: VPCs and firewalls in Google Cloud are global resources, so Cisco Cloud Network Controller does not have to program firewall rules for endpoints that span multiple regions. The same firewall rules apply for any region where the endpoint resides.
- **Firewall egress rules and network tags**: Firewall egress rules do not support network tags as a destination field, so you must list individual IP addresses for endpoints.
- **Source tags in firewall ingress rules and alias IP ranges**: Firewall ingress rules do not include the alias IP ranges of VMs matching the network tags used in the source field.
- **Priority fields in firewall rules**: Google Cloud evaluates firewall rules following their priority values.

Given that Google Cloud firewall rules follow a priority list, Cisco Cloud Network Controller configures a pair of low-priority deny-all ingress and egress rules when the VPC is created. Afterwards, Cisco Cloud Network Controller configures rules that open traffic according to the EPG’s contracts with higher priority. Therefore, if there is no explicit rule that allows certain traffic as a result of an EPG contract, the low-priority rule matches and the default behavior is deny-all.

Endpoints and Endpoint Selectors

On the Cisco Cloud Network Controller, a cloud EPG is a collection of endpoints that share the same security policy. Cloud EPGs can have endpoints in one or more subnets and are tied to a VRF.

The Cisco Cloud Network Controller has a feature called endpoint selector, which is used to assign an endpoint to a Cloud EPG. The endpoint selector is essentially a set of rules run against the cloud instances assigned to the Google Cloud VPC managed by Cisco ACI. Any endpoint selector rules that match endpoint instances will assign that endpoint to the Cloud EPG. The endpoint selector is similar to the attribute-based microsegmentation available in Cisco ACI.

Following are the types of endpoint selectors available for the two types of cloud EPGs:

- **Application EPGs**:
  - **IP**: Used to select by the IP address or subnet.
  - **Region**: Used to select by the region of the endpoint.

- **Custom**: Used to select by a custom tag or label. For example, if you added a Location tag in Google Cloud, you might create the custom tag Location in this field to match the Location tag that you added in Google Cloud earlier.

- **External EPGs**:

  Subnet: The subnet selector is a type of endpoint selector where the match expression uses the IP address of a subnet, so an entire subnet is assigned as being part of the EPG. Essentially, when you use the subnet selector as the endpoint selector, all of the endpoints within that subnet belongs to the associated EPG.

When using Cisco Cloud Network Controller endpoint selectors with Google Cloud, a network tag is applied that associates the EPG to the matching VM in Google Cloud. Once the network tag is configured in the VM, Google Cloud applies the firewall rules for the VM's traffic.

VMs on Google Cloud also support labels. Labels are key-value pairs that are meant to be an organizational tool. The custom endpoint selector in Cisco Cloud Network Controller recognizes the labels assigned to the VMs in Google Cloud.

Cisco Cloud Network Controller reserves a unique network tag string for each EPG. In Google Cloud, this value is used as the target field in the firewall rules created for the EPG. When a new VM matches an endpoint selector of the EPG, Cisco Cloud Network Controller appends this value to the existing VM's network tags. In addition, the EPG's network tag is used in the source field of the Google Cloud firewall rules.

For example, consider the sample configuration below:

```
<cloudEPg  name="epg1" >
    <cloudRsCloudEPgCtx tnFvCtxName="v1"/>
    <fvRsProv tnVzBrCPName="httpSSHFamily"/>
    <cloudEPSelector name="web-selector" matchExpression="custom:server=='web'"/>
    <cloudEPSelector name="web-selector" matchExpression="custom:server==backend"/>
</cloudEPg>
<cloudEPg  name="epg2" status="">
    <cloudRsCloudEPgCtx tnFvCtxName="v1"/>
    <fvRsCons tnVzBrCPName="httpSSHFamily"/>
   <cloudEPSelector name="database-selector" matchExpression="custom:server=='database'"/>
</cloudEPg>
```

Assuming there are three endpoints in the VPC with the following configuration, Cisco Cloud Network Controller configures the following network tags, where the Cisco Cloud Network Controller-configured network tags are in the following format:

`capic-<app-profile-name>-<epg-name>`

Endpoint	Application Profile	EPG	Primary IP	Labels	Cisco Cloud Network Controller-Configured Network Tags
EP1	First application profile (app01)	First EPG (epg01)	10.0.0.1	server:web	capic-app01-epg01
EP2	Second application profile (app02)	Second EPG (epg02)	20.0.0.1	server:backend	capic-app02-epg02
EP3	Second application profile (app02)	Third EPG (epg03)	30.0.0.1	server:database	capic-app02-epg03

Cisco Cloud Network Controller needs admin permission over the VMs in order to set their network tags. This permission is granted by the *Compute Instance Admin* role.

There might be cases where Cisco Cloud Network Controller does not have this permission and cannot manage the VM's tags. In those scenarios, you can configure the network tags in your VMs first and then provide the proper endpoint selector configuration to Cisco Cloud Network Controller later on.

To see firewall rules:

- **In Google Cloud**: In your Google Cloud account, navigate to **VPC Network** > **Firewall**.
  - If the VM is part of an EPG, you can find the endpoints by expanding a firewall rule and then viewing the multiple entries shown in the **Filters** column. which are the endpoints.
  - Use the entry in the **Type** column to determine if a particular firewall rule is an ingress or an egress firewall rule.
    - If the firewall rule is an ingress type, then traffic is being sent to these endpoints.
    - If the firewall rule is an egress type, then these entries show where it can receive the traffic.
- **In Cisco Cloud Network Controller**: Firewall rules are associated with VPCs, so navigate to **Cloud Resources** > **VPCs**, then double-click on a VPC to get the detail screen. Then click on the **Cloud Resources** tab; there you will see the ingress and egress rules.

Understanding VPCs and Subnets Under Google Cloud and Cloud Context Profiles Under Cisco Cloud Network Controller

In Google Cloud, a VPC is a global resource, whereas a subnet is regional and spans every availability zone in the region, but a subnet cannot overlap with other subnets in the same VPC or in peered VPCs.

Each subnet must have exactly one primary CIDR block (IP range) and can have up to 30 secondary CIDR blocks. There can be up to 300 primary and secondary CIDRs in a VPC. The NIC for each VM gets its primary internal IP address from the primary CIDR block, whereas secondary IP ranges can only be used for **alias IP ranges**, which is a Google Cloud organizational tool to assign address pools to containers or applications running inside the VM.

The following provides more information on the associations between Cisco Cloud Network Controller objects and Google Cloud objects:

- **One-to-one mapping of Google Cloud VPC to Cisco Cloud Network Controller VRF**: A Google Cloud VPC is deployed for each Cisco Cloud Network Controller VRF (`fvCtx` object). Cloud context profiles (`cloudCtxProfile` object) define the set of regional subnets to deploy. Every cloud context profile in the same VRF maps to the same VPC.
- **Google Cloud subnets and their secondary IP ranges**: Cisco Cloud Network Controller deploys a subnet with primary and secondary IP ranges using Cisco Cloud Network Controller CIDR and subnet objects. The Cisco Cloud Network Controller subnet object is used to represent an IP range and the Cisco Cloud Network Controller CIDRs's primary property tells whether it is primary or secondary. Secondary Cisco Cloud Network Controller subnet objects are associated with the corresponding primary one, because only the latter deploys the actual subnet in Google Cloud.

Understanding VPC Groups

The cloud context profile is used within Cisco Cloud Network Controller as a mapping tool for a VPC, where one cloud context profile is associated with one VPC. The cloud context profile also contains information on the region association, where the cloud context profile is used to determine which region a VPC gets deployed to.

In Google Cloud, when you want to create a VPC, you might have to create multiple cloud context profiles through Cisco Cloud Network Controller if you want to deploy subnets in multiple regions. However, VPCs are global in nature with Google Cloud, where a VPC spans all the regions.

Therefore, a property called **VPC group** (`vpcGroup`) is available within the cloud context profile that allows Cisco Cloud Network Controller to group multiple cloud context profiles together to form one VPC. Multiple cloud context profiles that are associated with each other using the VPC group feature form the VPC construct within Google Cloud, where the VPC group name is shown in Google Cloud.

Since only one Google Cloud VPC is allowed within one Cisco Cloud Network Controller VRF, you must use the same name for the VPC group property for each cloud context profile listed in a VRF. Profiles having the same VPC group name reside in the same VPC.

The scope of this matching mechanism is at the tenant level. The same values can be reused across tenants, but they implicitly define different groups, since they are also part of different Google Cloud Projects.

Cisco Cloud Network Controller deploys a VPC for each distinct `fvCtx`, `cloudRsToCtx` and `vpcGroup` tuple, as long as there is at least one `cloudSubnet` defined. The cloud context profile becomes a container of regional resources, such as subnets, associated to a VRF, and it no longer maps to a VPC.

The example below defines two context profiles (c1 and c2) inside the same VRF (v1) with one VPC group (vpc-1). This configuration deploys one VPC, where the subnets defined in profiles c1 and c2 are deployed in that VPC because they are part of the same VPC group.

```
<fvTenant name="t1">
    <fvCtx name="v1"/>
    <cloudCtxProfile name="c1" vpcGroup="vpc-1">
        <cloudRsCtxProfileToRegion tDn="uni/clouddomp/provp-gcp/region-us-west1" />
        <cloudRsToCtx tnFvCtxName="v1"/>
        <cloudCidr addr="10.0.0.0/16" primary="yes" >
            <cloudSubnet ip="10.0.1.0/24">
                <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-west1/zone-default"/>
            </cloudSubnet>
        </cloudCidr>
    </cloudCtxProfile>
    <cloudCtxProfile name="c2" vpcGroup="vpc-1">
        <cloudRsCtxProfileToRegion tDn="uni/clouddomp/provp-gcp/region-us-east1" />
        <cloudRsToCtx tnFvCtxName="v1"/>
        <cloudCidr addr="20.0.0.0/16" primary="yes" >
            <cloudSubnet ip="20.0.1.0/24">
                <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-east1/zone-default"/>
            </cloudSubnet>
        </cloudCidr>
    </cloudCtxProfile>
</fvTenant>
```

Understanding Primary and Secondary Subnets and Subnet Groups

Cisco Cloud Network Controller deploys every subnet (`cloudSubnet`) in the VPC (which is identified by the tuple `fvCtx`, `cloudRsToCtx`, and `vpcGroup`) in the region that is pointed to by the `cloudRsCtxProfileToRegion` relation.

In Google Cloud, there is no concept of a primary CIDR for the VPC, but the **primary** flag in the CIDR (`cloudCidr`) field in the cloud context profile is available for Cisco Cloud Network Controller to support secondary IP ranges. Every subnet configured under a primary CIDR will be deployed as an actual Google Cloud subnet with the specified primary IP range (named *primary subnets*). Having multiple CIDRs set as primary under a given cloud context profile (`cloudCtxProfile`) is supported. Therefore, you can have more than one primary CIDR under a given cloud context profile with multiple primary subnets.

The following POST shows an example where one VPC and three subnets are deployed in Google Cloud.

```
<polUni>
    <fvTenant name="t1">
        <fvCtx name="v1"/>
        <cloudCtxProfile name="c1" vpcGroup="vpc-1">
            <cloudRsCtxProfileToRegion tDn="uni/clouddomp/provp-gcp/region-us-west1" />
            <cloudRsToCtx tnFvCtxName="v1"/>
            <cloudCidr addr="10.0.0.0/16" primary="yes" >
                <cloudSubnet ip="10.0.1.0/24">
                    <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-west1/zone-default"/>
                </cloudSubnet>
                <cloudSubnet ip="10.0.2.0/24">
                    <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-west/zone-default"/>
                </cloudSubnet>
            </cloudCidr>
            <cloudCidr addr="20.0.0.0/16" primary="yes" >
                <cloudSubnet ip="20.0.1.0/24">
                    <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-west/zone-default"/>
                </cloudSubnet>
            </cloudCidr>
        </cloudCtxProfile>
</polUni>
```

In the example above, one VPC is configured for the VRF `v1` with three primary subnets (10.0.1.0/24, 10.0.2.0/24, and 20.0.1.0/24) deployed in the us-west region.

A secondary CIDR contains the secondary IP ranges (called *secondary subnets*) that are configured in the existing primary subnets. When designating a CIDR as either primary or secondary, it's helpful to consider these differences between the two:

- The primary CIDR is normally the VM.
- The secondary CIDR is more of a container used for the application.

You can group together primary and secondary subnets into a **subnet group**. This grouping mechanism assigns secondary subnets (for example, IP ranges) to a primary subnet, which is mapped to an actual Google Cloud subnet. The scope of the subnet group is at the cloud context profile level. While you can have multiple cloud context profiles within the same tenant, subnets are part of a subnet group only within the same cloud context profile.

You will use the **subnet group label** to assign a unique label to a specific subnet group. If you have multiple subnets that have the same subnet group label, then those subnets all belong to the same subnet group as long

as they are all within the same cloud context profile. Note that while the subnet group label is used within Cisco Cloud Network Controller to group primary and secondary subnets, it is not used in Google Cloud.

Note the following guidelines for the primary and secondary CIDRs:

- **Primary CIDR**:
  - Any subnet group can have at maximum of only one subnet from the primary CIDR.
  - You can have multiple subnets in the primary CIDR, but all of the subnets must be in a separate subnet group.
- **Secondary CIDR**: You can have multiple subnets from the secondary CIDR in the same subnet group.

The following POST shows an example where two VPCs with two subnets each in different regions and having secondary CIDRs are deployed in Google Cloud.

```
<polUni>
    <fvTenant name="t1">
        <fvCtx name="v1"/>
        <fvCtx name="v2"/>
        <cloudCtxProfile name="c1" vpcGroup="vpc-1">
            <cloudRsCtxProfileToRegion tDn="uni/clouddomp/provp-gcp/region-us-west1" />
            <cloudRsToCtx tnFvCtxName="v1"/>
            <cloudCidr addr="10.0.0.0/16" primary="yes" >
                <cloudSubnet ip="10.0.1.0/24" subnetGroup="subnet-1">
                    <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-west1/zone-default"/>
                </cloudSubnet>
                <cloudSubnet ip="10.0.2.0/24" subnetGroup="subnet-2">
                    <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-west1/zone-default"/>
                </cloudSubnet>
            </cloudCidr>
            <cloudCidr addr="40.0.0.0/16" primary="no">
                <cloudSubnet ip="40.0.1.0/24" subnetGroup="subnet-1">
                    <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-west1/zone-default"/>
                </cloudSubnet>
            </cloudCidr>
        </cloudCtxProfile>
        <cloudCtxProfile name="c2" vpcGroup="vpc-2">
            <cloudRsCtxProfileToRegion tDn="uni/clouddomp/provp-gcp/region-us-east1" />
            <cloudRsToCtx tnFvCtxName="v2"/>
            <cloudCidr addr="20.0.0.0/16" primary="yes">
                <cloudSubnet ip="20.0.1.0/24" subnetGroup="subnet-1">
                    <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-east1/zone-default"/>
                </cloudSubnet>
            </cloudCidr>
            <cloudCidr addr="30.0.0.0/16" primary="no">
                <cloudSubnet ip="30.0.1.0/24" subnetGroup="subnet-1">
                    <cloudRsZoneAttach
tDn="uni/clouddomp/provp-gcp/region-us-east1/zone-default"/>
                </cloudSubnet>
            </cloudCidr>
        </cloudCtxProfile>
    </fvTenant>
</polUni>
```

Note that the subnet group `subnet-1` in the cloud context profile `c2` is not the same subnet group in the cloud context profile `c1`, because the scope of the subnet group is at the cloud context profile level.

The intent of the example above is summarized as follows:

- Tenant `t1` defines VRF `v1` and `v2`.
- Cloud context profile `c1` defines the subnets in region us-west1 for VRF `v1` and VPC group `vpc-1`. This deploys VPC `vpc-1`.
- Cloud context profile `c2` defines the subnets in region us-east1 for VRF `v2` and VPC group `vpc-2`. This deploys VPC `vpc-2`.
- The following subnets are deployed in VPC `vpc-1` in region us-west1:
  - `Subnet-1` subnet group:
    - Primary IP range: 10.0.1.0/24
    - Secondary IP ranges: 40.0.1.0/24
  - `Subnet-2` subnet group:
    - Primary IP range: 10.0.2.0/24
- The following subnets are deployed in VPC `vpc-2` in region us-east1:
  - Subnet-1:
    - Primary IP range: 20.0.1.0/24
    - Secondary IP ranges: 30.0.1.0/24

Guidelines and Limitations For Configuring Cisco Cloud Network Controller with Google Cloud

Following are the guidelines and limitations when configuring Cisco Cloud Network Controller with Google Cloud:

- For releases prior to release 25.0(5), Google Cloud does not support routing based on contracts. For more information, see Inter-Site Connectivity Using BGP-EVPN, on page 8.
- External connectivity between two Google Cloud sites is not supported.
- The external VRF can be configured only in the infra tenant in Cisco Cloud Network Controller.
- The tenant `common` in Cisco Cloud Network Controller cannot be associated with any Google Cloud project.
- In Google Cloud, the infra VPC and spoke VPCs are connected through VPC peering.
- In order to configure connectivity between the on-premises data center and the public cloud, you must manually configure the remote device by downloading the external device configuration files and manually enabling connectivity between Google Cloud and the external devices.

The external device configuration files that you download are not final configurations. Instead, the external device configuration files are provided more as a guidance. You must manually modify the information in the configuration files to configure the Google Cloud Router with IPSec, which is used to create connectivity between the on-premises data center and the public cloud, where:

- The Google Cloud Router and tunnels are deployed in the infra (hub) VPC.
- One cloud router per region is supported. Cloud routers can be deployed in a maximum of four regions.
- Spoke VPCs peer with the infra VPC to share the VPN connections to external sites, such as the on-premises data center.

Naming Length Restrictions Imposed By Google Cloud Firewall Rules

Google Cloud firewall rules are named resources, and Cisco Cloud Network Controller derives a name from the internal policy and uses that to deploy the Google Cloud firewall rules. Cisco Cloud Network Controller uses the following naming scheme for the internal policy:

```
{VPC-name}-{in/eg}-{target App-name}-{target EPG-name}-{contract-name}
```

The maximum length for a Google Cloud firewall rule name is 62 characters. This imposes a restriction on the names that you can use when configuring the following Cisco Cloud Network Controller components whose names are used in the Google Cloud firewall rule name:

- VPC group
- Application profile
- Application EPG or external EPG
- Contract

Knowing that the maximum number of characters is 62 for a Google Cloud firewall rule name, and taking into account the fixed areas in the string that makes up the Google Cloud firewall rule name:

- Hyphens (4 characters total)
- `in` (ingress) or `eg` (egress) value (2 characters)

That means that the total number of characters available for the combined names of all of the individual Cisco Cloud Network Controller components cannot exceed 56:

62- 4 (number of hypens)- 2 (`in` or `eg` characters) = 56 characters

So, the sum of the lengths of the names of the VPC group, application profile, application EPG or external EPG, and contract must be smaller than 56 characters. On average, this allows for roughly 14 characters for the name of each component.