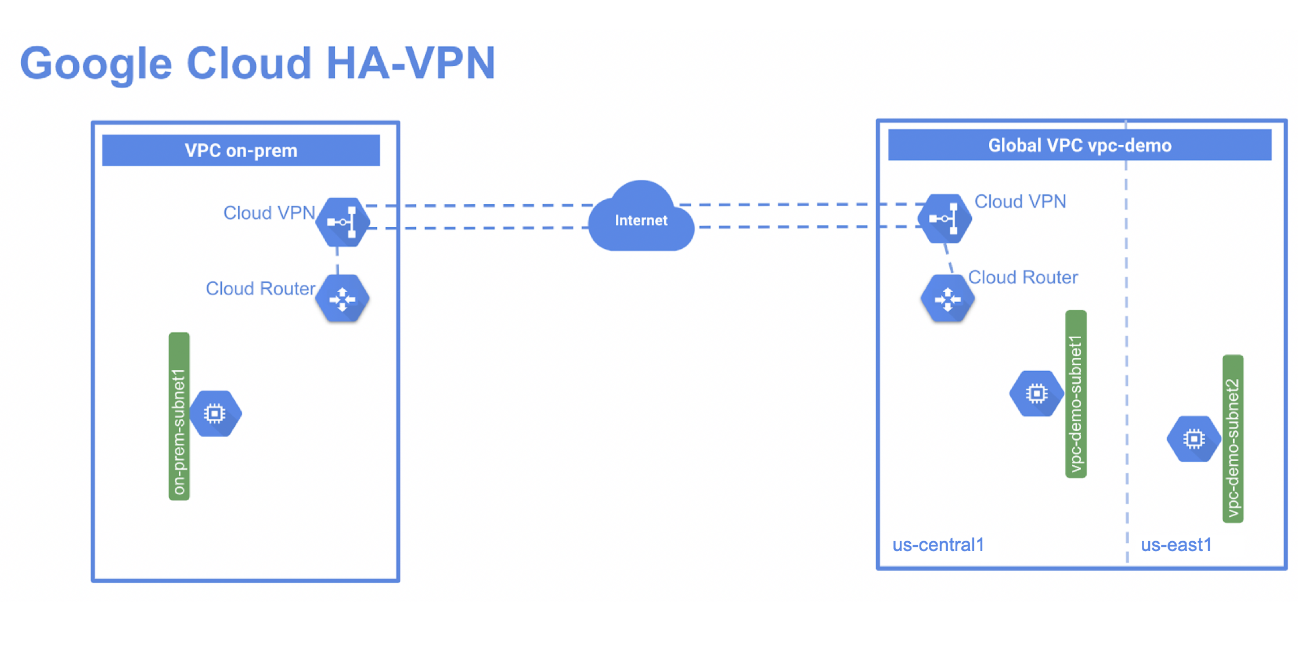
**Overview**

HA VPN is a high-availability (HA) Cloud VPN solution that lets you securely connect your on-premises network to your VPC network through an IPsec VPN connection in a single region. HA VPN provides an SLA of 99.99% service availability.

HA VPN is a regional per VPC, VPN solution. HA VPN gateways have two interfaces, each with its own public IP address. When you create an HA VPN gateway, two public IP addresses are automatically chosen from different address pools. When HA VPN is configured with two tunnels, Cloud VPN offers a 99.99% service availability uptime.

In this lab you create a global VPC called **vpc-demo**, with two custom subnets in **us-east1** and **us-central1**. In this VPC, you add a Compute Engine instance in each region. You then create a second VPC called **on-prem** to simulate a customer's on-premises data center. In this second VPC, you add a subnet in region **us-central1** and a Compute Engine instance running in this region. Finally, you add an HA VPN and a cloud router in each VPC and run two tunnels from each HA VPN gateway before testing the configuration to verify the 99.99% SLA.



**Objectives**

In this lab, you learn how to perform the following tasks:

* Create two VPC networks and instances.
* Configure HA VPN gateways.
* Configure dynamic routing with VPN tunnels.
* Configure global dynamic routing mode.
* Verify and test HA VPN gateway configuration.

**Setup and requirements**

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, 1:15:00), and make sure you can finish within that time.  
   There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.
4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.  
   If you use other credentials, you'll receive errors or **incur charges**.
7. Accept the terms and skip the recovery resource page.

**Note:** Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

**How to start your lab and sign in to the Console**

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



1. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Choose an account** page.

**Note:**Open the tabs in separate windows, side-by-side.

1. On the Choose an account page, click **Use Another Account**. The Sign in page opens.



1. Paste the username that you copied from the Connection Details panel. Then copy and paste the password.

**Note:**You must use the credentials from the Connection Details panel. Do not use your Google Cloud Skills Boost credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

1. Click through the subsequent pages:

* Accept the terms and conditions.
* Do not add recovery options or two-factor authentication (because this is a temporary account).
* Do not sign up for free trials.

After a few moments, the Cloud console opens in this tab.

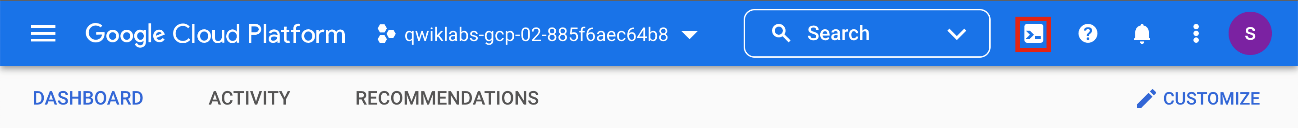
**Note:**You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left. 

Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud.

Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.



1. Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT\_ID*. For example:



**gcloud** is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

* You can list the active account name with this command:

gcloud auth list

Copied!

content\_copy

**Output:**

Credentialed accounts:

- @.com (active)

**Example output:**

Credentialed accounts:

- google1623327\_student@qwiklabs.net

* You can list the project ID with this command:

gcloud config list project

Copied!

content\_copy

**Output:**

[core]

project =

**Example output:**

[core]

project = qwiklabs-gcp-44776a13dea667a6

**Note:**Full documentation of **gcloud** is available in the [gcloud CLI overview guide](https://cloud.google.com/sdk/gcloud).

**Task 1. Set up a Global VPC environment**

In this task you set up a Global VPC with two custom subnets and two VM instances running in each zone.

1. In Cloud Shell, create a VPC network called **vpc-demo**:

gcloud compute networks create vpc-demo --subnet-mode custom

Copied!

content\_copy

The output should look similar to this:

Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/global/networks/vpc-demo].

NAME: vpc-demo

SUBNET\_MODE: CUSTOM

BGP\_ROUTING\_MODE: REGIONAL

IPV4\_RANGE:

GATEWAY\_IPV4:

1. In Cloud Shell, create subnet **vpc-demo-subnet1** in the region **us-central1**:

gcloud compute networks subnets create vpc-demo-subnet1 \

--network vpc-demo --range 10.1.1.0/24 --region us-central1

Copied!

content\_copy

1. Create subnet **vpc-demo-subnet2** in the region **us-east1**:

gcloud compute networks subnets create vpc-demo-subnet2 \

--network vpc-demo --range 10.2.1.0/24 --region us-east1

Copied!

content\_copy

1. Create a firewall rule to allow all custom traffic within the network:

gcloud compute firewall-rules create vpc-demo-allow-custom \

--network vpc-demo \

--allow tcp:0-65535,udp:0-65535,icmp \

--source-ranges 10.0.0.0/8

Copied!

content\_copy

The output should look similar to this:

Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/global/firewalls/vpc-demo-allow-custom].

Creating firewall...done.

NAME: vpc-demo-allow-custom

NETWORK: vpc-demo

DIRECTION: INGRESS

PRIORITY: 1000

ALLOW: tcp:0-65535,udp:0-65535,icmp

DENY:

DISABLED: False

1. Create a firewall rule to allow SSH, ICMP traffic from anywhere:

gcloud compute firewall-rules create vpc-demo-allow-ssh-icmp \

--network vpc-demo \

--allow tcp:22,icmp

Copied!

content\_copy

1. Create a VM instance **vpc-demo-instance1** in zone **us-central1-b**:

gcloud compute instances create vpc-demo-instance1 --zone us-central1-b --subnet vpc-demo-subnet1

Copied!

content\_copy

The output should look similar to this:

Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/zones/us-central1-b/instances/vpc-demo-instance1].

NAME: vpc-demo-instance1

ZONE: us-central1-b

MACHINE\_TYPE: n1-standard-1

PREEMPTIBLE:

INTERNAL\_IP: 10.1.1.2

EXTERNAL\_IP: 34.71.135.218

STATUS: RUNNING

1. Create a VM instance **vpc-demo-instance2** in zone **us-east1-b**:

gcloud compute instances create vpc-demo-instance2 --zone us-east1-b --subnet vpc-demo-subnet2

Copied!

content\_copy

**Task 2. Set up a simulated on-premises environment**

In this task you create a VPC called **on-prem** that simulates an on-premises environment from where a customer connects to the Google cloud environment.

1. in Cloud Shell, create a VPC network called **on-prem**:

gcloud compute networks create on-prem --subnet-mode custom

Copied!

content\_copy

The output should look similar to this:

Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/global/networks/on-prem].

NAME: on-prem

SUBNET\_MODE: CUSTOM

BGP\_ROUTING\_MODE: REGIONAL

IPV4\_RANGE:

GATEWAY\_IPV4:

1. Create a subnet called **on-prem-subnet1**:

gcloud compute networks subnets create on-prem-subnet1 \

--network on-prem --range 192.168.1.0/24 --region us-central1

Copied!

content\_copy

1. Create a firewall rule to allow all custom traffic within the network:

gcloud compute firewall-rules create on-prem-allow-custom \

--network on-prem \

--allow tcp:0-65535,udp:0-65535,icmp \

--source-ranges 192.168.0.0/16

Copied!

content\_copy

1. Create a firewall rule to allow SSH, RDP, HTTP, and ICMP traffic to the instances:

gcloud compute firewall-rules create on-prem-allow-ssh-icmp \

--network on-prem \

--allow tcp:22,icmp

Copied!

content\_copy

1. Create an instance called **on-prem-instance1** in the region **us-central1**:

gcloud compute instances create on-prem-instance1 --zone us-central1-a --subnet on-prem-subnet1

Copied!

content\_copy

**Task 3. Set up an HA VPN gateway**

In this task you create an HA VPN gateway in each VPC network and then create HA VPN tunnels on each Cloud VPN gateway.

1. In Cloud Shell, create an HA VPN in the **vpc-demo network**:

gcloud compute vpn-gateways create vpc-demo-vpn-gw1 --network vpc-demo --region us-central1

Copied!

content\_copy

The output should look similar to this:

Creating VPN Gateway...done.

NAME: vpc-demo-vpn-gw1

INTERFACE0: 35.242.117.95

INTERFACE1: 35.220.73.93

NETWORK: vpc-demo

REGION: us-central1

1. Create an HA VPN in the **on-prem** network:

gcloud compute vpn-gateways create on-prem-vpn-gw1 --network on-prem --region us-central1

Copied!

content\_copy

1. View details of the **vpc-demo-vpn-gw1** gateway to verify its settings:

gcloud compute vpn-gateways describe vpc-demo-vpn-gw1 --region us-central1

Copied!

content\_copy

The output should look similar to this:

creationTimestamp: '2022-01-25T03:02:20.983-08:00'

id: '7306781839576950355'

kind: compute#vpnGateway

labelFingerprint: 42WmSpB8rSM=

name: vpc-demo-vpn-gw1

network: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/global/networks/vpc-demo

region: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1

selfLink: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnGateways/vpc-demo-vpn-gw1

vpnInterfaces:

- id: 0

ipAddress: 35.242.117.95

- id: 1

ipAddress: 35.220.73.93

1. View details of the **on-prem-vpn-gw1** vpn-gateway to verify its settings:

gcloud compute vpn-gateways describe on-prem-vpn-gw1 --region us-central1

Copied!

content\_copy

The output should look similar to this:

creationTimestamp: '2022-01-25T03:03:34.305-08:00'

id: '3697047034868688873'

kind: compute#vpnGateway

labelFingerprint: 42WmSpB8rSM=

name: on-prem-vpn-gw1

network: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/global/networks/on-prem

region: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1

selfLink: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnGateways/on-prem-vpn-gw1

vpnInterfaces:

- id: 0

ipAddress: 35.242.106.234

- id: 1

ipAddress: 35.220.88.140

Create cloud routers

1. Create a cloud router in the **vpc-demo** network:

gcloud compute routers create vpc-demo-router1 \

--region us-central1 \

--network vpc-demo \

--asn 65001

Copied!

content\_copy

The output should look similar to this:

Creating router [vpc-demo-router1]...done.

NAME: vpc-demo-router1

REGION: us-central1

NETWORK: vpc-demo

1. Create a cloud router in the **on-prem** network:

gcloud compute routers create on-prem-router1 \

--region us-central1 \

--network on-prem \

--asn 65002

Copied!

content\_copy

**Task 4. Create two VPN tunnels**

In this task you create VPN tunnels between the two gateways. For HA VPN setup, you add two tunnels from each gateway to the remote setup. You create a tunnel on **interface0** and connect to **interface0** on the remote gateway. Next, you create another tunnel on **interface1** and connect to **interface1** on the remote gateway.

When you run HA VPN tunnels between two Google Cloud VPCs, you need to make sure that the tunnel on **interface0** is connected to **interface0** on the remote VPN gateway. Similarly, the tunnel on **interface1** must be connected to **interface1** on the remote VPN gateway.

**Note:**In your own environment, if you run HA VPN to a remote VPN gateway on-premises for a customer, you can connect in one of the following ways:

* *Two on-premises VPN gateway devices:* Each of the tunnels from each interface on the Cloud VPN gateway must be connected to its own peer gateway.
* *A single on-premises VPN gateway device with two interfaces:* Each of the tunnels from each interface on the Cloud VPN gateway must be connected to its own interface on the peer gateway.
* *A single on-premises VPN gateway device with a single interface:* Both of the tunnels from each interface on the Cloud VPN gateway must be connected to the same interface on the peer gateway.

In this lab you are simulating an on-premises setup with both VPN gateways in Google Cloud. You ensure that **interface0** of one gateway connects to **interface0** of the other and **interface1** connects to **interface1** of the remote gateway.

1. Create the first VPN tunnel in the **vpc-demo** network:

gcloud compute vpn-tunnels create vpc-demo-tunnel0 \

--peer-gcp-gateway on-prem-vpn-gw1 \

--region us-central1 \

--ike-version 2 \

--shared-secret [SHARED\_SECRET] \

--router vpc-demo-router1 \

--vpn-gateway vpc-demo-vpn-gw1 \

--interface 0

Copied!

content\_copy

The output should look similar to this:

Creating VPN tunnel...done.

NAME: vpc-demo-tunnel0

REGION: us-central1

GATEWAY: vpc-demo-vpn-gw1

VPN\_INTERFACE: 0

PEER\_ADDRESS: 35.242.106.234

1. Create the second VPN tunnel in the **vpc-demo** network:

gcloud compute vpn-tunnels create vpc-demo-tunnel1 \

--peer-gcp-gateway on-prem-vpn-gw1 \

--region us-central1 \

--ike-version 2 \

--shared-secret [SHARED\_SECRET] \

--router vpc-demo-router1 \

--vpn-gateway vpc-demo-vpn-gw1 \

--interface 1

Copied!

content\_copy

1. Create the first VPN tunnel in the **on-prem** network:

gcloud compute vpn-tunnels create on-prem-tunnel0 \

--peer-gcp-gateway vpc-demo-vpn-gw1 \

--region us-central1 \

--ike-version 2 \

--shared-secret [SHARED\_SECRET] \

--router on-prem-router1 \

--vpn-gateway on-prem-vpn-gw1 \

--interface 0

Copied!

content\_copy

1. Create the second VPN tunnel in the **on-prem** network:

gcloud compute vpn-tunnels create on-prem-tunnel1 \

--peer-gcp-gateway vpc-demo-vpn-gw1 \

--region us-central1 \

--ike-version 2 \

--shared-secret [SHARED\_SECRET] \

--router on-prem-router1 \

--vpn-gateway on-prem-vpn-gw1 \

--interface 1

Copied!

content\_copy

**Task 5. Create Border Gateway Protocol (BGP) peering for each tunnel**

In this task you configure BGP peering for each VPN tunnel between **vpc-demo** and VPC **on-prem**. HA VPN requires dynamic routing to enable 99.99% availability.

1. Create the router interface for **tunnel0** in network **vpc-demo**:

gcloud compute routers add-interface vpc-demo-router1 \

--interface-name if-tunnel0-to-on-prem \

--ip-address 169.254.0.1 \

--mask-length 30 \

--vpn-tunnel vpc-demo-tunnel0 \

--region us-central1

Copied!

content\_copy

The output should look similar to this:

Updated [https://www.googleapis.com/compute/v1/projects/binal-sandbox/regions/us-central1/routers/vpc-demo-router1].

1. Create the BGP peer for **tunnel0** in network **vpc-demo**:

gcloud compute routers add-bgp-peer vpc-demo-router1 \

--peer-name bgp-on-prem-tunnel0 \

--interface if-tunnel0-to-on-prem \

--peer-ip-address 169.254.0.2 \

--peer-asn 65002 \

--region us-central1

Copied!

content\_copy

The output should look similar to this:

Creating peer [bgp-on-prem-tunnel0] in router [vpc-demo-router1]...done.

1. Create a router interface for **tunnel1** in network **vpc-demo**:

gcloud compute routers add-interface vpc-demo-router1 \

--interface-name if-tunnel1-to-on-prem \

--ip-address 169.254.1.1 \

--mask-length 30 \

--vpn-tunnel vpc-demo-tunnel1 \

--region us-central1

Copied!

content\_copy

1. Create the BGP peer for **tunnel1** in network **vpc-demo**:

gcloud compute routers add-bgp-peer vpc-demo-router1 \

--peer-name bgp-on-prem-tunnel1 \

--interface if-tunnel1-to-on-prem \

--peer-ip-address 169.254.1.2 \

--peer-asn 65002 \

--region us-central1

Copied!

content\_copy

1. Create a router interface for **tunnel0** in network **on-prem**:

gcloud compute routers add-interface on-prem-router1 \

--interface-name if-tunnel0-to-vpc-demo \

--ip-address 169.254.0.2 \

--mask-length 30 \

--vpn-tunnel on-prem-tunnel0 \

--region us-central1

Copied!

content\_copy

1. Create the BGP peer for **tunnel0** in network **on-prem**:

gcloud compute routers add-bgp-peer on-prem-router1 \

--peer-name bgp-vpc-demo-tunnel0 \

--interface if-tunnel0-to-vpc-demo \

--peer-ip-address 169.254.0.1 \

--peer-asn 65001 \

--region us-central1

Copied!

content\_copy

1. Create a router interface for **tunnel1** in network **on-prem**:

gcloud compute routers add-interface on-prem-router1 \

--interface-name if-tunnel1-to-vpc-demo \

--ip-address 169.254.1.2 \

--mask-length 30 \

--vpn-tunnel on-prem-tunnel1 \

--region us-central1

Copied!

content\_copy

1. Create the BGP peer for **tunnel1** in network **on-prem**:

gcloud compute routers add-bgp-peer on-prem-router1 \

--peer-name bgp-vpc-demo-tunnel1 \

--interface if-tunnel1-to-vpc-demo \

--peer-ip-address 169.254.1.1 \

--peer-asn 65001 \

--region us-central1

Copied!

content\_copy

**Task 6. Verify router configurations**

In this task you verify the router configurations in both VPCs. You configure firewall rules to allow traffic between each VPC and verify the status of the tunnels. You also verify private connectivity over VPN between each VPC and enable global routing mode for the VPC.

1. View details of Cloud Router **vpc-demo-router1** to verify its settings:

gcloud compute routers describe vpc-demo-router1 \

--region us-central1

Copied!

content\_copy

The output should look similar to this:

bgp:

advertiseMode: DEFAULT

asn: 65001

keepaliveInterval: 20

bgpPeers:

- bfd:

minReceiveInterval: 1000

minTransmitInterval: 1000

multiplier: 5

sessionInitializationMode: DISABLED

enable: 'TRUE'

interfaceName: if-tunnel0-to-on-prem

ipAddress: 169.254.0.1

name: bgp-on-prem-tunnel0

peerAsn: 65002

peerIpAddress: 169.254.0.2

- bfd:

minReceiveInterval: 1000

minTransmitInterval: 1000

multiplier: 5

sessionInitializationMode: DISABLED

enable: 'TRUE'

interfaceName: if-tunnel1-to-on-prem

ipAddress: 169.254.1.1

name: bgp-on-prem-tunnel1

peerAsn: 65002

peerIpAddress: 169.254.1.2

creationTimestamp: '2022-01-25T03:06:23.370-08:00'

id: '2408056426544129856'

interfaces:

- ipRange: 169.254.0.1/30

linkedVpnTunnel: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnTunnels/vpc-demo-tunnel0

name: if-tunnel0-to-on-prem

- ipRange: 169.254.1.1/30

linkedVpnTunnel: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnTunnels/vpc-demo-tunnel1

name: if-tunnel1-to-on-prem

kind: compute#router

name: vpc-demo-router1

network: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/global/networks/vpc-demo

region: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1

selfLink: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/routers/vpc-demo-router1

1. View details of Cloud Router **on-prem-router1** to verify its settings:

gcloud compute routers describe on-prem-router1 \

--region us-central1

Copied!

content\_copy

The output should look similar to this:

bgp:

advertiseMode: DEFAULT

asn: 65002

keepaliveInterval: 20

bgpPeers:

- bfd:

minReceiveInterval: 1000

minTransmitInterval: 1000

multiplier: 5

sessionInitializationMode: DISABLED

enable: 'TRUE'

interfaceName: if-tunnel0-to-vpc-demo

ipAddress: 169.254.0.2

name: bgp-vpc-demo-tunnel0

peerAsn: 65001

peerIpAddress: 169.254.0.1

- bfd:

minReceiveInterval: 1000

minTransmitInterval: 1000

multiplier: 5

sessionInitializationMode: DISABLED

enable: 'TRUE'

interfaceName: if-tunnel1-to-vpc-demo

ipAddress: 169.254.1.2

name: bgp-vpc-demo-tunnel1

peerAsn: 65001

peerIpAddress: 169.254.1.1

creationTimestamp: '2022-01-25T03:07:40.360-08:00'

id: '3252882979067946771'

interfaces:

- ipRange: 169.254.0.2/30

linkedVpnTunnel: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnTunnels/on-prem-tunnel0

name: if-tunnel0-to-vpc-demo

- ipRange: 169.254.1.2/30

linkedVpnTunnel: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnTunnels/on-prem-tunnel1

name: if-tunnel1-to-vpc-demo

kind: compute#router

name: on-prem-router1

network: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/global/networks/on-prem

region: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1

selfLink: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/routers/on-prem-router1

Configure firewall rules to allow traffic from the remote VPC

Configure firewall rules to allow traffic from the private IP ranges of peer VPN.

1. Allow traffic from network VPC **on-prem** to **vpc-demo**:

gcloud compute firewall-rules create vpc-demo-allow-subnets-from-on-prem \

--network vpc-demo \

--allow tcp,udp,icmp \

--source-ranges 192.168.1.0/24

Copied!

content\_copy

The output should look similar to this:

Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/global/firewalls/vpc-demo-allow-subnets-from-on-prem].

Creating firewall...done.

NAME: vpc-demo-allow-subnets-from-on-prem

NETWORK: vpc-demo

DIRECTION: INGRESS

PRIORITY: 1000

ALLOW: tcp,udp,icmp

DENY:

DISABLED: False

1. Allow traffic from **vpc-demo** to network VPC **on-prem**:

gcloud compute firewall-rules create on-prem-allow-subnets-from-vpc-demo \

--network on-prem \

--allow tcp,udp,icmp \

--source-ranges 10.1.1.0/24,10.2.1.0/24

Copied!

content\_copy

Verify the status of the tunnels

1. List the VPN tunnels you just created:

gcloud compute vpn-tunnels list

Copied!

content\_copy

There should be four VPN tunnels (two tunnels for each VPN gateway). The output should look similar to this:

NAME: on-prem-tunnel0

REGION: us-central1

GATEWAY: on-prem-vpn-gw1

PEER\_ADDRESS: 35.242.117.95

NAME: on-prem-tunnel1

REGION: us-central1

GATEWAY: on-prem-vpn-gw1

PEER\_ADDRESS: 35.220.73.93

NAME: vpc-demo-tunnel0

REGION: us-central1

GATEWAY: vpc-demo-vpn-gw1

PEER\_ADDRESS: 35.242.106.234

NAME: vpc-demo-tunnel1

REGION: us-central1

GATEWAY: vpc-demo-vpn-gw1

PEER\_ADDRESS: 35.220.88.140

1. Verify that **vpc-demo-tunnel0** tunnel is up:

gcloud compute vpn-tunnels describe vpc-demo-tunnel0 \

--region us-central1

Copied!

content\_copy

The tunnel output should show detailed status as *Tunnel is up and running.*

creationTimestamp: '2022-01-25T03:21:05.238-08:00'

description: ''

detailedStatus: Tunnel is up and running.

id: '3268990180169769934'

ikeVersion: 2

kind: compute#vpnTunnel

labelFingerprint: 42WmSpB8rSM=

localTrafficSelector:

- 0.0.0.0/0

name: vpc-demo-tunnel0

peerGcpGateway: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnGateways/on-prem-vpn-gw1

peerIp: 35.242.106.234

region: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1

remoteTrafficSelector:

- 0.0.0.0/0

router: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/routers/vpc-demo-router1

selfLink: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnTunnels/vpc-demo-tunnel0

sharedSecret: '\*\*\*\*\*\*\*\*\*\*\*\*\*'

sharedSecretHash: AOs4oVY4bX91gba6DIeg1DbtzWTj

status: ESTABLISHED

vpnGateway: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnGateways/vpc-demo-vpn-gw1

vpnGatewayInterface: 0

1. Verify that **vpc-demo-tunnel1** tunnel is up:

gcloud compute vpn-tunnels describe vpc-demo-tunnel1 \

--region us-central1

Copied!

content\_copy

The tunnel output should show detailed status as *Tunnel is up and running.*

1. Verify that **on-prem-tunnel0** tunnel is up:

gcloud compute vpn-tunnels describe on-prem-tunnel0 \

--region us-central1

Copied!

content\_copy

The tunnel output should show detailed status as *Tunnel is up and running.*

1. Verify that **on-prem-tunnel1** tunnel is up:

gcloud compute vpn-tunnels describe on-prem-tunnel1 \

--region us-central1

Copied!

content\_copy

The tunnel output should show detailed status as *Tunnel is up and running.*

Verify private connectivity over VPN

1. Open a new Cloud Shell tab and type the following to connect via SSH to the instance **on-prem-instance1**:

gcloud compute ssh on-prem-instance1 --zone us-central1-a

Copied!

content\_copy

1. Type "y" to confirm that you want to continue.
2. Press **Enter** twice to skip creating a password.
3. From the instance **on-prem-instance1** in network **on-prem**, to reach instances in network **vpc-demo**, ping 10.1.1.2:

ping -c 4 10.1.1.2

Copied!

content\_copy

Pings are successful. The output should look similar to this:

PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.

64 bytes from 10.1.1.2: icmp\_seq=1 ttl=62 time=9.65 ms

64 bytes from 10.1.1.2: icmp\_seq=2 ttl=62 time=2.01 ms

64 bytes from 10.1.1.2: icmp\_seq=3 ttl=62 time=1.71 ms

64 bytes from 10.1.1.2: icmp\_seq=4 ttl=62 time=1.77 ms

--- 10.1.1.2 ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 8ms

rtt min/avg/max/mdev = 1.707/3.783/9.653/3.391 ms

Global routing with VPN

HA VPN is a regional resource and cloud router that by default only sees the routes in the region in which it is deployed. To reach instances in a different region than the cloud router, you need to enable global routing mode for the VPC. This allows the cloud router to see and advertise routes from other regions.

1. Open a new Cloud Shell tab and update the **bgp-routing mode** from **vpc-demo** to **GLOBAL**:

gcloud compute networks update vpc-demo --bgp-routing-mode GLOBAL

Copied!

content\_copy

1. Verify the change:

gcloud compute networks describe vpc-demo

Copied!

content\_copy

The output should look similar to this:

autoCreateSubnetworks: false

creationTimestamp: '2022-01-25T02:52:58.553-08:00'

id: '4735939730452146277'

kind: compute#network

name: vpc-demo

routingConfig:

routingMode: GLOBAL

selfLink: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/global/networks/vpc-demo

subnetworks:

- https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/subnetworks/vpc-demo-subnet1

- https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-east1/subnetworks/vpc-demo-subnet2

x\_gcloud\_bgp\_routing\_mode: GLOBAL

x\_gcloud\_subnet\_mode: CUSTOM

1. From the Cloud Shell tab that is currently connected to the instance in network **on-prem** via **ssh**, ping the instance **vpc-demo-instance2** in region us-east1:

ping -c 2 10.2.1.2

Copied!

content\_copy

Pings are successful. The output should look similar to this:

PING 10.2.1.2 (10.2.1.2) 56(84) bytes of data.

64 bytes from 10.2.1.2: icmp\_seq=1 ttl=62 time=34.9 ms

64 bytes from 10.2.1.2: icmp\_seq=2 ttl=62 time=32.2 ms

--- 10.2.1.2 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 2ms

rtt min/avg/max/mdev = 32.189/33.528/34.867/1.339 ms

**Task 7. Verify and test the configuration of HA VPN tunnels**

In this task you will test and verify that the high availability configuration of each HA VPN tunnel is successful.

1. Open a new Cloud Shell tab.
2. Bring **tunnel0** in network **vpc-demo** down:

gcloud compute vpn-tunnels delete vpc-demo-tunnel0 --region us-central1

Copied!

content\_copy

Respond "y" when asked to verify the deletion. The respective **tunnel0** in network **on-prem** will go down.

1. Verify that the tunnel is down:

gcloud compute vpn-tunnels describe on-prem-tunnel0 --region us-central1

Copied!

content\_copy

The detailed status should show as *Handshake\_with\_peer\_broken*.

creationTimestamp: '2022-01-25T03:22:33.581-08:00'

description: ''

detailedStatus: Handshake with peer broken for unknown reason. Trying again soon.

id: '4116279561430393750'

ikeVersion: 2

kind: compute#vpnTunnel

localTrafficSelector:

- 0.0.0.0/0

name: on-prem-tunnel0

peerGcpGateway: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnGateways/vpc-demo-vpn-gw1

peerIp: 35.242.117.95

region: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1

remoteTrafficSelector:

- 0.0.0.0/0

router: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/routers/on-prem-router1

selfLink: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnTunnels/on-prem-tunnel0

sharedSecret: '\*\*\*\*\*\*\*\*\*\*\*\*\*'

sharedSecretHash: AO3jeFtewmjvTMO7JEM5RuyCtqaa

status: FIRST\_HANDSHAKE

vpnGateway: https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-03-cdb29e18d20d/regions/us-central1/vpnGateways/on-prem-vpn-gw1

vpnGatewayInterface: 0

1. Switch to the previous Cloud Shell tab that has the open **ssh** session running, and verify the pings between the instances in network **vpc-demo** and network **on-prem**:

ping -c 3 10.1.1.2

Copied!

content\_copy

The output should look similar to this:

PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.

64 bytes from 10.1.1.2: icmp\_seq=1 ttl=62 time=6.31 ms

64 bytes from 10.1.1.2: icmp\_seq=2 ttl=62 time=1.13 ms

64 bytes from 10.1.1.2: icmp\_seq=3 ttl=62 time=1.20 ms

--- 10.1.1.2 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 5ms

rtt min/avg/max/mdev = 1.132/2.882/6.312/2.425 ms

Pings are still successful because the traffic is now sent over the second tunnel. You have successfully configured HA VPN tunnels.

**Task 8. (Optional) Clean up lab environment**

In this task you clean up the resources you have used. This task is optional. When you end the lab, all your resources and your project are cleaned up and discarded for you automatically. However, you should know how to clean up resources yourself in your own environment to save on costs and reduce resource usage.

Delete VPN tunnels

* From Cloud Shell, type the following commands to delete the remaining tunnels. Type "y" to confirm each action when asked:

gcloud compute vpn-tunnels delete on-prem-tunnel0 --region us-central1

Copied!

content\_copy

gcloud compute vpn-tunnels delete vpc-demo-tunnel1 --region us-central1

Copied!

content\_copy

gcloud compute vpn-tunnels delete on-prem-tunnel1 --region us-central1

Copied!

content\_copy

Remove BGP peering

* Type the following commands from each BGP peer to remove peering:

gcloud compute routers remove-bgp-peer vpc-demo-router1 --peer-name bgp-on-prem-tunnel0 --region us-central1

Copied!

content\_copy

gcloud compute routers remove-bgp-peer vpc-demo-router1 --peer-name bgp-on-prem-tunnel1 --region us-central1

Copied!

content\_copy

gcloud compute routers remove-bgp-peer on-prem-router1 --peer-name bgp-vpc-demo-tunnel0 --region us-central1

Copied!

content\_copy

gcloud compute routers remove-bgp-peer on-prem-router1 --peer-name bgp-vpc-demo-tunnel1 --region us-central1

Copied!

content\_copy

Delete cloud routers

* Type each command to delete the routers. Type "y" to confirm each action when asked:

gcloud compute routers delete on-prem-router1 --region us-central1

Copied!

content\_copy

gcloud compute routers delete vpc-demo-router1 --region us-central1

Copied!

content\_copy

Delete VPN gateways

* Type each command to delete the VPN gateways. Type "y" to confirm each action when asked:

gcloud compute vpn-gateways delete vpc-demo-vpn-gw1 --region us-central1

Copied!

content\_copy

gcloud compute vpn-gateways delete on-prem-vpn-gw1 --region us-central1

Copied!

content\_copy

Delete instances

* Type the following commands to delete each instance. Type "y" to confirm each action when asked:

gcloud compute instances delete vpc-demo-instance1 --zone us-central1-b

Copied!

content\_copy

gcloud compute instances delete vpc-demo-instance2 --zone us-east1-b

Copied!

content\_copy

gcloud compute instances delete on-prem-instance1 --zone us-central1-a

Copied!

content\_copy

Delete firewall rules

* Type the following to delete the firewall rules. Type "y" to confirm each action when asked:

gcloud compute firewall-rules delete vpc-demo-allow-custom

Copied!

content\_copy

gcloud compute firewall-rules delete on-prem-allow-subnets-from-vpc-demo

Copied!

content\_copy

gcloud compute firewall-rules delete on-prem-allow-ssh-icmp

Copied!

content\_copy

gcloud compute firewall-rules delete on-prem-allow-custom

Copied!

content\_copy

gcloud compute firewall-rules delete vpc-demo-allow-subnets-from-on-prem

Copied!

content\_copy

gcloud compute firewall-rules delete vpc-demo-allow-ssh-icmp

Copied!

content\_copy

Delete subnets

* Type the following to delete the subnets. Type "y" to confirm each action when asked:

gcloud compute networks subnets delete vpc-demo-subnet1 --region us-central1

Copied!

content\_copy

gcloud compute networks subnets delete vpc-demo-subnet2 --region us-east1

Copied!

content\_copy

gcloud compute networks subnets delete on-prem-subnet1 --region us-central1

Copied!

content\_copy

Delete VPC

* Type these commands to delete the VPCs. Type "y" to confirm each action when asked:

gcloud compute networks delete vpc-demo

Copied!

content\_copy

gcloud compute networks delete on-prem

Copied!

content\_copy

**Task 9: Review**

In this lab you configured HA VPN gateways. You also configured dynamic routing with VPN tunnels and configured global dynamic routing mode. Finally you verified that HA VPN is configured and functioning correctly.

**End your lab**