This page describes special networking configurations of Compute Engine virtual machine (VM) instances, such as the following:

- Setting up an external HTTP connection to a VM
- Configuring a VM as a network proxy
- Configuring a VM as a VPN gateway
- Configuring a VM as a NAT gateway
- Building high availability and high bandwidth NAT gateways

The default firewall rules do not allow HTTP or HTTPS connections to your instances. However, it is fairly simple to add a rule that does allow them. Note that a VM must have an external (static or ephemeral) IP address (/compute/docs/ip-addresses#externaladdresses) before it can receive traffic from outside its Virtual Private Cloud (VPC) network.

You can add a firewall rule to allow HTTP or HTTPS connections using the `gcloud` command-line tool or the Google Cloud Console (https://console.cloud.google.com/). You can also add a firewall rule through the API (/compute/docs/reference/latest/firewalls/insert).

You can design your VPC network so that only one instance has external access, and all other instances in the VPC network use that instance as a proxy server to the outside world. This is useful if you want to control access into or out of your VPC network, or reduce the cost of paying for multiple external IP addresses.

This particular example discusses how to set up a network proxy on VM instances that use a Debian image. It uses a gateway instance as a Squid proxy server (http://www.squid-cache.org/) but this is only one way of setting up a proxy server.

To set up a Squid proxy server:

1. Set up one instance with an external (static or ephemeral) IP address (/compute/docs/ip-addresses#externaladdresses). For this example, name your instance **`gateway-instance`**.
2. Set up one or more instances without external IP addresses by specifying **`gcloud compute instances create ... --no-address`**. For this example, call this instance **`hidden-instance`**.
3. Learn how to connect from one instance to another (/compute/docs/instances/connecting-advanced#sshbetweeninstances) because you will not be able to connect directly into your internal-only instances.
4. Add a firewall to allow tcp traffic on port 3128:
5. Install Squid (http://www.squid-cache.org/) on **`gateway-instance`**, and configure it to allow access from any machines on the VPC network (RFC 1918 (https://tools.ietf.org/html/rfc1918), RFC 4193 (https://tools.ietf.org/html/rfc4193), and RFC 4291 (https://tools.ietf.org/html/rfc4291) IP spaces). This assumes that **`gateway-instance`** and **`hidden-instance`** are both connected to the same VPC network, which enables them to connect to each other.

   Enable any machine on the local network to use the Squid3 server. The following **`sed`** commands uncomment and enable the **`acl localnet src`** entries in the Squid config files for local networks and machines.

6. Configure `hidden-instance` to use `gateway-instance` as its proxy. Use ssh to connect into `hidden-instance` and define its proxy URL addresses to point to `gateway-instance` on port 3128 (the default Squid configuration) as shown here:

Update sudoers to pass these env variables through.

★ **Note:** VM instances that use a proxy server won't be able to access the metadata server by default, as all requests to the metadata server will be forwarded to the proxy.

7. Exit `sudo`, load the variables, and run `apt-get` on `hidden-instance`. It should now work using gateway as a proxy. If gateway were not serving as a proxy, `apt-get` would not work because `hidden-instance` has no direct connection to the Internet.

You can use Strongswan VPN software to set up a VPN gateway on one of your instances. For most users, we recommend that you use Cloud VPN (/vpn/docs/concepts/overview) instead of Strongswan. With Cloud VPN, you don't need to create and configure a VM to run VPN software. Use Strongswan in cases where Cloud VPN doesn't provide required functionality.

The following setup is only an example of how you might set up your own VPN gateways. We do not validate or endo
arty software and do not guarantee that the following example works for all scenarios.

1. Create a VPC network that your on-premises network will connect to.

2. Create a subnet with an IP range that doesn't overlap with your on-premises subnet.

3. Create a VM instance in the `vpn-subnet` subnet. This instance will be your VPN gateway.

4. Look up and record your VPN gateway's internal and external IP address.

The external IP address is the value of the `natIP` field. The internal IP address is the value of the `networkIP` field, such as 10.0.0.2.

5. Create a VM instance that communicates with clients in your on-premises network through the VPN gateway.

6. Create a route in the `vpn-network` network to route traffic through vpn-gateway if it is destined for your on-premises network.

   The `[VPN_GATEWAY_INTERNAL_IP]` value is the internal IP address of your VPN gateway (the value of the `networkIP` field).

7. Add the following firewall rules to your VPC network to accept incoming traffic.

Create firewall rules in your on-premises network to accept incoming traffic from the VPC network.

8. Connect (/compute/docs/instances/connecting-to-instance) to your VPN gateway instance.
9. Install and configure Strongswan, the VPN software.

   From the home directory, create a file named `ipsec.conf`. Populate it with the following contents, replacing the placeholders with your environment's values:

   Then, run the following commands, replacing `[secret-key]` with a secret key (a string value):

You must also configure your on-premises VPN gateway to successfully establish a VPN tunnel.

If your on-premises gateway machine is running a Debian-based operating system, you can use the same steps to install and configure Strongswan. For example, make a copy of your `ipsec.conf` file and switch the left and right IDs and subnets.

10. Test your VPN tunnel by pinging an on-premises machine from the `test-vpn` instance:

If you are experiencing issues with your VPN setup based on the instructions above, try these tips to troubleshoot your setup:

1. Check the status of your connection:

   If `myconn` isn't listed, start up the connection:

2. Determine whether the two VPN endpoints are able to communicate at all.

   Use netcat (http://netcat.sourceforge.net/) to send VPN-like traffic (UDP, port 4500). Run the following command on your local VPN endpoint:

   Run `tcpdump` on the receiving end to determine that your VM instance can receive the packet on port 4500:

3. Turn on more verbose logging by adding the following lines to your `ipsec.conf` files:

   Next, retry your connection. Although the connection should still fail, you can check the log for errors. The log file should be located at `/var/log/charon.log` on your VM instance.

Consider using Cloud NAT (/nat/docs/) instead of configuring a VM as a NAT gateway. If you already have an instan
NAT gateway configured, you can migrate your traffic by following the Migrate an instance-based NAT gateway to C
‡migrate-nat) procedure. If you require an instance-based NAT gateway, consider creating an internal TCP/UDP load
cer and using it as a next hop (/load-balancing/docs/internal/ilb-next-hop-overview#specifications) instead of follow
ections in this section. An internal TCP/UDP load balancer supports health checks that use TCP, allowing you to verif
of backend instances so that you can control how new connections are routed
-balancing/docs/internal/#traffic_distribution).

You can create more complicated networking scenarios by making changes to the routes collection. This section describes how you can set up an internal address translation (NAT) gateway instance that can route traffic from internal-only virtual machine instances to the Internet. This allows you to use one external IP address to send traffic from multiple virtual machine instances but only expose a single virtual machine to the Internet.

1. To start, create a VPC network (/vpc/docs/vpc) to host your virtual machine instances for this scenario.

2. Create subnet for the `us-central1` region.

3. Create firewall rules to allow ssh connections in the new network you just created.

4. Create a virtual machine to act as a NAT gateway on `custom-network1`.

5. Tag (/vpc/docs/add-remove-network-tags) any virtual machine instances without an external IP address that will use the gateway instance with the tag `no-ip`, or create a new virtual machine without an external IP address and tag the instance with the `no-ip` tag.

   - Add tags to an existing instance.

   - Alternatively, create a new virtual machine without an external IP address.

6. Create a route to send traffic destined to the Internet through your gateway instance.

Setting the priority of this route ensures that this route wins if there are any other conflicting routes. 1000 is the default priority and a value lower than 1000 will take precedent.

7. Next, log on to your gateway instance and configure iptables to NAT internal traffic to the Internet.

★ **Note:** These examples assume the interface is called `eth0`. Different Linux distributions use different names for interfaces. Modify the name of the interface in commands to match your distribution.

On your instance, configure iptables:

The first `sudo` command tells the kernel that you want to allow IP forwarding. The second `sudo` command masquerades packets received from internal instances as if they were sent from the NAT gateway instance.

To inspect your iptables NAT rules, use the list option:

Check that the output is similar to the following example:

8. (Optional) If you want these settings to persist across future reboots:

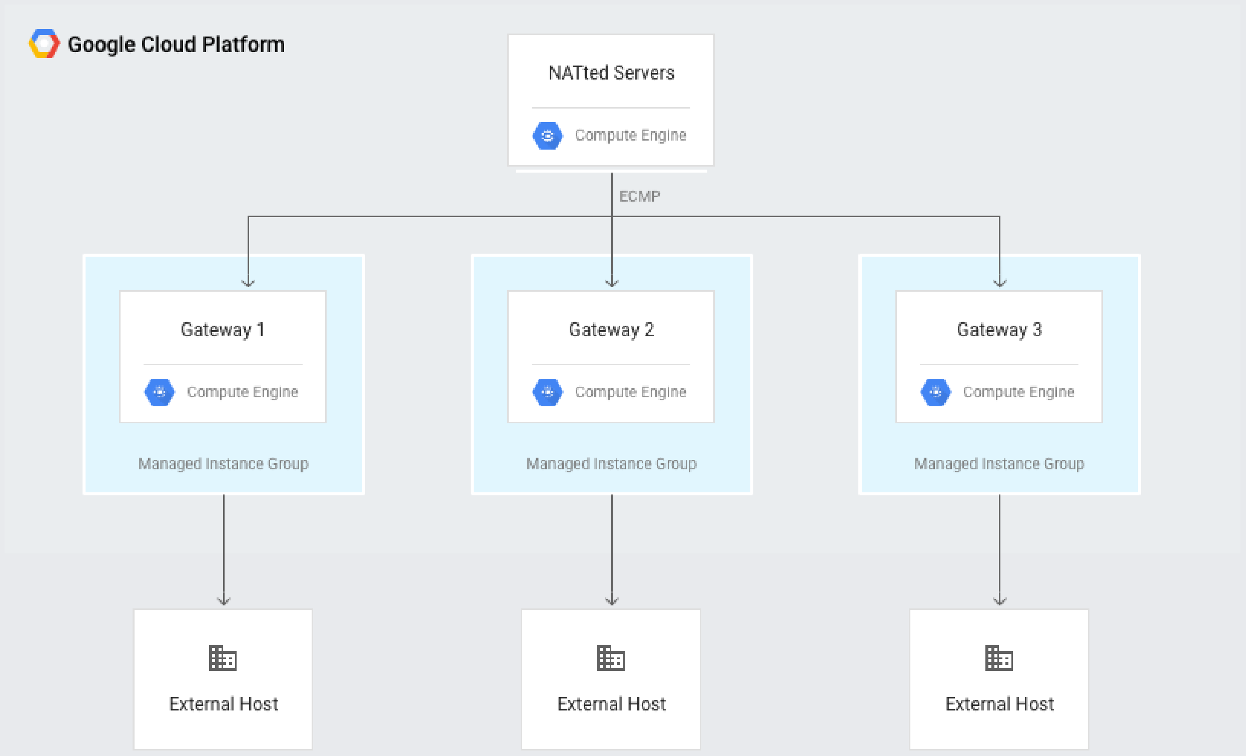
Consider using Cloud NAT (/nat/docs/) instead of configuring a VM as a NAT gateway. If you already have an instan
NAT gateway configured, you can migrate your traffic by following the Migrate an instance-based NAT gateway to C
‡migrate-nat) procedure. If you require an instance-based NAT gateway, consider creating an internal TCP/UDP load
cer and using it as a next hop (/load-balancing/docs/internal/ilb-next-hop-overview#specifications) instead of follow
ections in this section. An internal TCP/UDP load balancer supports health checks that use TCP, allowing you to verif
of backend instances so that you can control how new connections are routed
-balancing/docs/internal/#traffic_distribution).

This section describes how to set up multiple NAT gateways with Equal Cost Multi-Path (ECMP) routing and autohealing enabled for a more resilient and high-bandwidth deployment.

Google Cloud uses RFC 1918 private IP addresses for virtual machines. If these VMs need access to resources on the public internet, a NAT is required. A single NAT gateway architecture (#natgateway) is sufficient for simple scenarios. However, higher throughput or higher availability requires a more resilient architecture.

In instances where multiple routes have the same priority, Google Cloud uses ECMP routing to distribute traffic. In this case, you create several NAT gateways to receive parts of the traffic through ECMP. The NAT gateways then forward the traffic to external hosts with their public IP addresses.

The following diagram shows this configuration.



For higher resiliency, you place each gateway in a separate managed instance group with size 1 and attach a simple health check to ensure that the gateways will automatically restart if they fail. The gateways are in separate instance groups so they'll have a static external IP attached to the instance template. You provision three `n1-standard-2` NAT gateways in this example, but you can use any other number or size of gateway that you want. For example, `n1-standard-2` instances are capped at 4 Gbps of network traffic; if you need to handle a higher volume of traffic, you might choose `n1-standard-8`.

1. Create a VPC network (if needed). If you're not adding these gateways to an existing VPC, create a VPC network and subnet for them. If you are adding them to an existing VPC, skip to the second step and modify the regions as appropriate for your environment.

    a. Using Cloud Shell, create a custom VPC associated with your Google Cloud project. This VPC allows you to use non-default IP addressing, but does not include any default firewall rules:

b. Create a subnet within this VPC, and specify a region and IP range. For this tutorial, use `10.0.1.0/24` and the `us-east1` region:

2. Reserve and store three static IP addresses.

   a. Reserve and store an address named `nat-1` in the `us-east1` region:

   b. Reserve and store an address named `nat-2` in `us-east1`:

   c. Reserve and store an address named `nat-3` in `us-east1`:

3. Create three instance templates with reserved IPs.

   a. Copy the startup config:

If you cannot access the startup script, copy it from the Startup script section (#startup).

b. Create a `nat-1` instance template:

c. Create a `nat-2` instance template:

d. Create a `nat-3` instance template:

The `n1-standard-2` machine type has two vCPUs and can use up to 4 Gbps of network bandwidth. If you need more bandwidth, you might want to choose a different host. Bandwidth scales at 2 Gbps per vCPU, up to 16 Gbps on an 8vCPU host.

4. Create a health check to monitor responsiveness:

If a system fails and can't respond to HTTP traffic, it is restarted. In this case, since you need a project, you can use an existing project or create a new one.

5. Create a VM instance group for each NAT gateway:

6. Set up autohealing to restart unresponsive NAT gateways:

7. Add default routes to your VPC network, which apply to instances that use the NAT:

8. Tag the instances that you want to use the NAT:

9. Test NAT functionality. With your gateways configured and your guest VMs tagged, ping external hosts without giving your VMs external IPs, as in this example:

Example output:

This configuration provides three NAT gateways in the us-east1 region, each capable of 2 Gbps. ECMP load balancing isn't perfect, though, and an individual flow is not spread across multiple links.

- A Terraform module for this configuration is also available (https://www.google.com/url?q=https://github.com/GoogleCloudPlatform/terraform-google-nat-gateway/tree/master/examples/ha-nat-gateway&sa=D&ust=1512500207728000&usg=AFQjCNGMCkIAu9IwkUzNEF4oTzNw8zQ93Q) for automating deployments.
- This configuration is best for ephemeral or non-stateful outbound links. If the size of the NAT gateway pool changes, TCP connections might be rebalanced, which could result in resetting an established connection.
- The nodes are not automatically updated, so if a Debian default installation presents a threat, you will need to update manually.
- These instances are all in the us-east1 region. If your VMs are in other zones, you might get better performance by moving gateways closer to those zones.
- Bandwidth per gateway is up to 2 Gbps per core unidirectional. During a gateway failure, traffic is distributed to the remaining gateways, but because running flows are not reprogrammed, traffic does not immediately resettle when the gateway comes back online. So make sure you allow enough overhead when sizing.
- To be alerted of unexpected results, use Stackdriver to monitor the managed instance groups and network traffic.

This example assumes the interface is called **`eth0`**. Different Linux distributions use different names for interfaces. M
me of the interface in commands to match your distribution.

Startup script referenced in step 3a (#step-3a):

If you have an instance-based NAT gateway but would like to migrate to Cloud NAT, do the following:

1. Configure Cloud NAT (/nat/docs) in the same region where you have the instance-based gateway.
2. Delete the static route or routes (/vpc/docs/using-routes#deletingaroute) sending packets the instance-based NAT gateway. Note that you still should have a default gateway route for your network.
    - If you used the single gateway example (#natgateway) above, the route is called `no-ip-internet-route`.
    - If you used the high-availability and high-bandwidth example (#multiple-natgateways) above, the routes are called `natroute1`, `natroute2`, and `natroute3`.
3. Traffic should start flowing through Cloud NAT.
4. Once you have confirmed that everything is working, delete your instance-based NAT gateways.

- To learn more about VPC networks, see VPC network overview (/vpc/docs/vpc).
- To create, modify, or delete VPC networks, see Using VPC networks (/vpc/docs/using-vpc).