Setting Up Internal TCP/UDP Load Balancing

This guide uses an example to teach the fundamentals of Google Cloud Internal TCP/UDP Load Balancing. Before following this guide, familiarize yourself with the following:

- Internal TCP/UDP Load Balancing concepts (https://cloud.google.com/load-balancing/docs/internal/index)
- Firewall rules overview (https://cloud.google.com/vpc/docs/firewalls)
- Health check concepts (https://cloud.google.com/load-balancing/docs/health-check-concepts)

Permissions

To follow this guide, you need to create instances and modify a network in a project. You should be either a project owner or editor (https://cloud.google.com/iam/docs/understanding-roles#primitive_roles), or you should have all of the following Compute Engine IAM roles (https://cloud.google.com/compute/docs/access/iam):

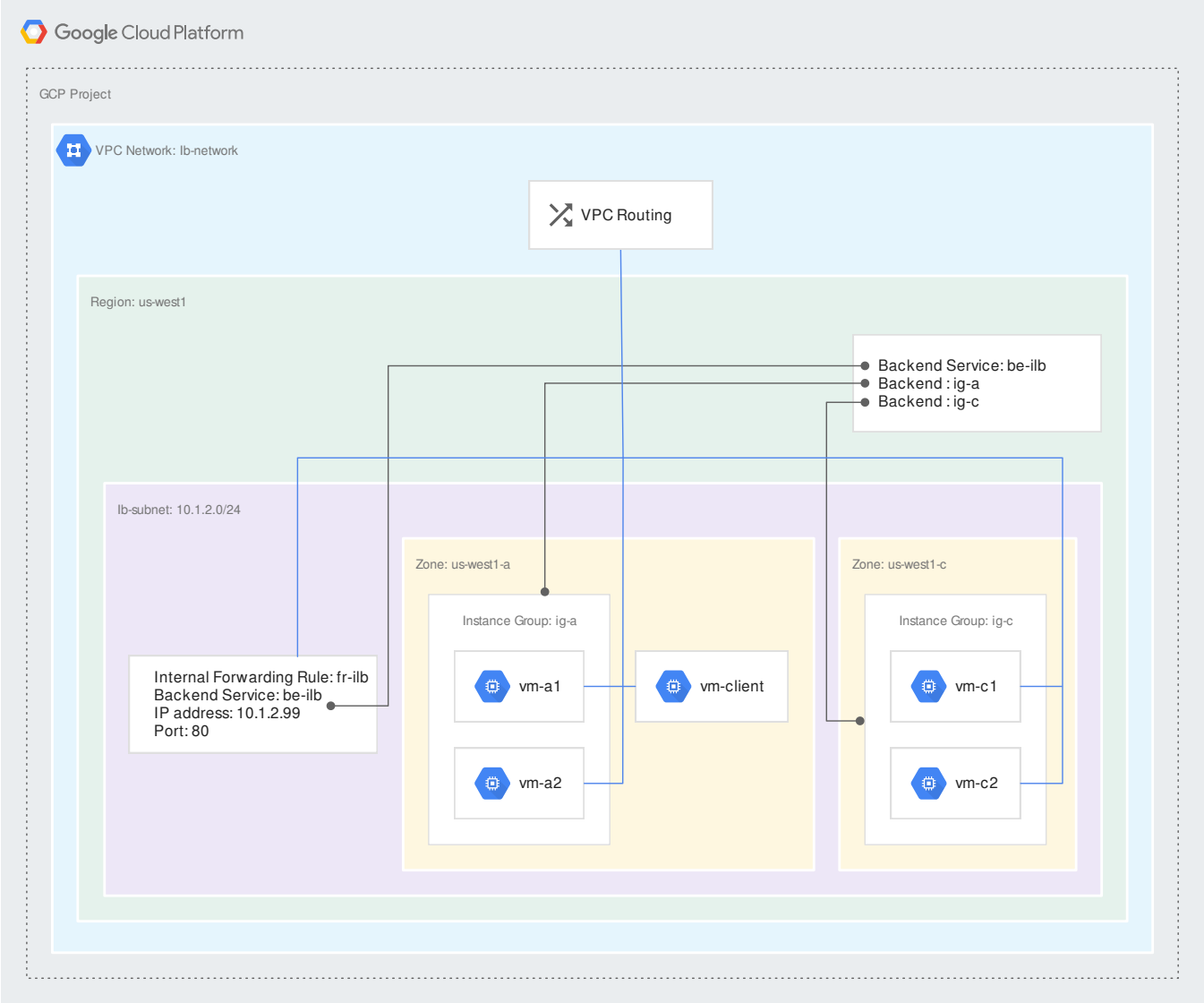
Task	Required Role
Create networks, subnets, and load balancer components	Network Admin (https://cloud.google.com/compute/docs/access/iam#compute.networkAdmin)
Add and remove firewall rules	Security Admin (https://cloud.google.com/compute/docs/access/iam#compute.securityAdmin)
Create instances	Instance Admin (https://cloud.google.com/compute/docs/access/iam#compute.instanceAdmin)

Setup

This guide shows you how to configure and test an internal TCP/UDP load balancer. The steps in this section describe how to configure the following:

1. A sample VPC network with custom subnets
2. Firewall rules that allow incoming connections to backend VMs
3. Four backend VMs:
    - Two VMs in an unmanaged instance group in zone `us-west1-a`
    - Two VMs in an unmanaged instance group in zone `us-west1-c`
4. One client VM to test connections
5. The following internal TCP/UDP load balancer components:
    - A health check for the backend service
    - An internal backend service in the `us-west1` region to manage connection distribution to the two zonal instance groups
    - An internal forwarding rule and internal IP address for the frontend of the load balancer

The architecture for this example looks like this:



(https://cloud.google.com/load-balancing/images/ilb-example.svg)
Internal TCP/UDP Load Balancing Example Configuration (click to enlarge)

Note: For simplicity, this example uses unmanaged instance groups. In production, consider using a managed instance group, as illustrated in Managed instance groups (#internal_load_balancing_with_regional_instance_groups).

Configuring a network, region, and subnet

You need a VPC network with at least one subnet. An internal TCP/UDP load balancer is regional. Traffic within the VPC network is routed to the load balancer if its source is from a subnet in the same region as the load balancer.

This example uses the following VPC network, region, and subnet:

- Network: The network is a custom mode VPC network (https://cloud.google.com/vpc/docs/vpc#subnet-ranges) named `lb-network`.
- Region: The region is `us-west1`.
- Subnet: The subnet, `lb-subnet`, uses the `10.1.2.0/24` IP range.

Note: You can change the name of the network, the region, and the parameters for the subnet; however, subsequent steps in this guide use the network, region, and subnet parameters as outlined above.

To create the example network and subnet, follow these steps.

CONSOLE GCLOUD API

1. Go to the VPC networks page in the Google Cloud Console.
   GO TO THE VPC NETWORK PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/NETWORKING/NETW
2. Click **Create VPC network**.
3. Enter a **Name** of `lb-network`.
4. In the **Subnets** section:
   - Set the **Subnet creation mode** to **Custom**.
   - In the **New subnet** section, enter the following information:
     - **Name**: `lb-subnet`
     - **Region**: `us-west1`
     - **IP address range**: `10.1.2.0/24`
     - Click **Done**.
5. Click **Create**.

Configuring firewall rules

This example uses the following firewall rules:

- `fw-allow-lb-subnet`: An ingress rule, applicable to all targets in the VPC network, allowing traffic from sources in the `10.1.2.0/24` range. This rules allows incoming traffic from any source within the `lb-subnet` to the instances (VMs) being load balanced.

- `fw-allow-ssh`: An ingress rule, applicable to the instances being load balanced, that allows incoming SSH connectivity on TCP port 22 from any address. You can choose a more restrictive source IP range for this rule; for example, you can specify just the IP ranges of the system from which you will initiating SSH sessions. This example uses the target tag `allow-ssh` to identify the VMs to which it should apply.
- `fw-allow-health-check`: An ingress rule, applicable to the instances being load balanced, that allows traffic from the Google Cloud health checking systems (`130.211.0.0/22` and `35.191.0.0/16`). This example uses the target tag `allow-health-check` to identify the instances to which it should apply.

Without these firewall rules, the default deny ingress (https://cloud.google.com/vpc/docs/firewalls#default_firewall_rules) rule blocks incoming traffic to the backend instances.

Note: You must create a firewall rule to allow health checks from the IP ranges of Google Cloud probe systems. See probe IP ranges (https://cloud.google.com/load-balancing/docs/health-check-concepts) for more information.

CONSOLE GCLOUD API

1. Go to the Firewall rules page in the Google Cloud Console.
   GO TO THE FIREWALL RULES PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/NETWORKING/FIRE
2. Click **Create firewall rule** and enter the following information to create the rule to allow subnet traffic:
   - **Name**: `fw-allow-lb-subnet`
   - **Network**: `lb-network`
   - **Priority**: `1000`
   - **Direction of traffic**: ingress
   - **Action on match**: allow
   - **Targets**: All instances in the network
   - **Source filter**: `IP ranges`
   - **Source IP ranges**: `10.1.2.0/24`
   - **Protocols and ports**: Allow all
3. Click **Create**.

4. Click **Create firewall rule** again to create the rule to allow incoming SSH connections:
    - **Name**: `fw-allow-ssh`
    - **Network**: `lb-network`
    - **Priority**: `1000`
    - **Direction of traffic**: ingress
    - **Action on match**: allow
    - **Targets**: Specified target tags
    - **Target tags**: `allow-ssh`
    - **Source filter**: `IP ranges`
    - **Source IP ranges**: `0.0.0.0/0`
    - **Protocols and ports**: Choose *Specified protocols and ports* then type: `tcp:22`
5. Click **Create**.
6. Click **Create firewall rule** a third time to create the rule to allow Google Cloud health checks:
    - **Name**: `fw-allow-health-check`
    - **Network**: `lb-network`
    - **Priority**: `1000`
    - **Direction of traffic**: ingress
    - **Action on match**: allow
    - **Targets**: Specified target tags
    - **Target tags**: `allow-health-check`
    - **Source filter**: `IP ranges`
    - **Source IP ranges**: `130.211.0.0/22` and `35.191.0.0/16`
    - **Protocols and ports**: Allow all
7. Click **Create**.

Creating backend VMs and instance groups

This example uses two *unmanaged* instance groups each having two backend (server) VMs. To demonstrate the regional nature of Internal TCP/UDP Load Balancing, the two instance groups are placed in separate zones, `us-west1-a` and `us-west1-c`.

- Instance group `ig-a` contains these two VMs:
    - `vm-a1`

    - `vm-a2`
- Instance group `ig-c` contains these two VMs:
    - `vm-c1`
    - `vm-c2`

Traffic to all four of the backend VMs is load balanced. Each of the four run an Apache web server on TCP ports 80 and 443. Each is assigned an internal IP address in the `lb-subnet` and an ephemeral external (public) IP address. You can remove the external IP addresses later (#remove_external_ip).

External IP address for the backend VMs are not required; however, they are useful for this example because they permit the backend VMs to download Apache from the internet, and they make it easy to connect via SSH (https://cloud.google.com/compute/docs/instances/connecting-to-instance#gcetools).

By default, Apache is configured to bind to any IP address. Internal TCP/UDP load balancers deliver packets by preserving the destination IP. Ensure that server software running on your backend VMs is listening on the IP address of the load balancer's internal forwarding rule. If you configure multiple internal forwarding rules (https://cloud.google.com/load-balancing/docs/internal/index#multiple_forwarding_rule), ensure that your software listens to the internal IP address associated with each one. The destination IP address of a packet delivered to a backend VM by an internal TCP/UDP load balancer is the internal IP address of the forwarding rule.

For instructional simplicity, these backend VMs run Debian GNU/Linux 9.

CONSOLE GCLOUD API

Create backend VMs

1. Go to the VM instances page in the Google Cloud Console.
   GO TO THE VM INSTANCES PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/COMPUTE/INSTANCE
2. Repeat the following steps to create four VMs, using the following name and zone combinations.
    - Name: `vm-a1`, zone: `us-west1-a`
    - Name: `vm-a2`, zone: `us-west1-a`
    - Name: `vm-c1`, zone: `us-west1-c`
    - Name: `vm-c2`, zone: `us-west1-c`

3. Click **Create instance**.
4. Set the **Name** as indicated in step 2.
5. For the **Region**, choose `us-west1`, and choose a **Zone** as indicated in step 2.
6. In the **Boot disk** section, ensure that the selected image is *Debian GNU/Linux 9 Stretch*. Click **Choose** to change the image if necessary.
7. Click **Management, security, disks, networking, sole tenancy** and make the following changes:
   - Click **Networking** and add the following **Network tags**: `allow-ssh` and `allow-health-check`
   - Click the edit button under **Network interfaces** and make the following changes then click **Done**:
     - **Network**: `lb-network`
     - **Subnet**: `lb-subnet`
     - **Primary internal IP**: Ephemeral (automatic)
     - **External IP**: Ephemeral
   - Click **Management**. In the **Startup script** field, copy and paste the following script contents. The script contents are identical for all four VMs:

```
#! /bin/bash
apt-get update
apt-get install apache2 -y
a2ensite default-ssl
a2enmod ssl
vm_hostname="$(curl -H "Metadata-Flavor:Google" \
http://169.254.169.254/computeMetadata/v1/instance/name)"
echo "Page served from: $vm_hostname" | \
tee /var/www/html/index.html
systemctl restart apache2
```

8. Click **Create**.

Create instance groups

1. Go to the Instance groups page in the Google Cloud Console.

   GO TO THE INSTANCE GROUPS PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/COMPUTE/INSTA

2. Repeat the following steps to create two unmanaged instance groups each with two VMs in them, using these combinations.
   - Instance group: `ig-a`, zone: `us-west1-a`, VMs: `vm-a1` and `vm-a2`
   - Instance group: `ig-c`, zone: `us-west1-c`, VMs: `vm-c1` and `vm-c2`

3. Click **Create instance group**.
4. Set **Name** as indicated in step 2.
5. In the **Location** section, select **Single-zone**, choose `us-west1` for the **Region**, and then choose a **Zone** as indicated in step 2.
6. In the **Group type** section, select **Unmanaged instance group**.
7. For **Network**, enter `lb-network`.
8. For **Subnetwork**, enter `lb-subnet`.
9. In the **VM instances** section, add the VMs as indicated in step 2.
10. Click **Create**.

Creating a client VM

This example creates a client VM (`vm-client`) in the same region as the backend (server) VMs. The client is used to validate the load balancer's configuration and demonstrate expected behavior as described in the testing (#test-your-load-balancer) section.

CONSOLE GCLOUD API

1. Go to the VM instances page in the Google Cloud Console.
   GO TO THE VM INSTANCES PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/COMPUTE/INSTANCE
2. Click **Create instance**.
3. Set the **Name** to `vm-client`.
4. Set the **Zone** to `us-west1-a`.
5. Click **Management, security, disks, networking, sole tenancy** and make the following changes:
   - Click **Networking** and add the `allow-ssh` to **Network tags**.
   - Click the edit button under **Network interfaces** and make the following changes then click **Done**:
     - **Network**: `lb-network`
     - **Subnet**: `lb-subnet`
     - **Primary internal IP**: Ephemeral (automatic)
     - **External IP**: Ephemeral
6. Click **Create**.

Configuring load balancer components

These steps configure all of the internal TCP/UDP load balancer components (https://cloud.google.com/load-balancing/docs/internal/#components) starting with the health check and backend service, and then the frontend components:

- **Health check**: In this example, we use an HTTP health check that simply checks for an HTTP `200` (OK) response. For more information, see the health checks section of the Internal TCP/UDP Load Balancing overview (https://cloud.google.com/load-balancing/docs/internal/#health-checking).
- **Backend service**: Because we need to pass HTTP traffic through the internal load balancer, we need to use TCP, not UDP.
- **Forwarding rule**: This example creates a single internal forwarding rule.
- **Internal IP address**: In this example, we specify an internal IP address, `10.1.2.99`, when we create the forwarding rule.

CONSOLE GCLOUD API

Create the load balancer and configure a backend service

1. Go to the Load balancing page in the Google Cloud Console.
   GO TO THE LOAD BALANCING PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/NETWORKING/LOA
2. Click **Create load balancer**.
3. Under **TCP load balancing**, click **Start configuration**.
4. Under **Internet facing or internal only** select **Only between my VMs**.
5. Click **Continue**.
6. Set the **Name** to `be-ilb`.
7. Click **Backend configuration** and make the following changes:
   a. **Region**: `us-west1`
   b. **Network**: `lb-network`
   c. Under **Backends**, in the **New item** section, select the `ig-a` instance group and click **Done**.
   d. Click **Add backend**. In the **New item** section that appears, select the `ig-c` instance group and click **Done** again.
   e. Under **Health check**, choose **Create another health check**, enter the following information, and click **Save and continue**:

- **Name**: `hc-http-80`
- **Protocol**: `HTTP`
- **Port**: `80`
- **Proxy protocol**: `NONE`
- **Request path**: `/`

f. Verify that there is a blue check mark next to **Backend configuration** before continuing. Review this step if not.

8. Click **Frontend configuration**. In the **New Frontend IP and port** section, make the following changes:
   a. **Name**: `fr-ilb`
   b. **Subnetwork**: `lb-subnet`
   c. From **Internal IP**, choose **Reserve a static internal IP address**, enter the following information, and click **Reserve**:
      - **Name**: `ip-ilb`
      - **Static IP address**: *Let me choose*
      - **Custom IP address**: `10.1.2.99`
   d. **Ports**: Choose **Single**, and enter `80` for the **Port number**.
   e. Verify that there is a blue check mark next to **Frontend configuration** before continuing. Review this step if not.
9. Click **Review and finalize**. Double-check your settings.
10. Click **Create**.

Testing

These tests show how to validate your load balancer configuration and learn about its expected behavior.

Testing load balancing

This test contacts the load balancer from a separate client VM; that is, not from a backend VM of the load balancer. The expected behavior is for traffic to be distributed among the four backend VMs because no session affinity has been configured (https://cloud.google.com/load-balancing/docs/internal/index#traffic_distribution).

1. Connect to the client VM instance.

```
gcloud compute ssh vm-client --zone=us-west1-a
```

2. Make a web request to the load balancer using `curl` to contact its IP address. Repeat the request so you can see that responses come from different backend VMs. The name of the VM generating the response is displayed in the text in the HTML response, by virtue of the contents of `/var/www/html/index.html` on each backend VM. Expected responses look like: `Page served from: vm-a1`, `Page served from: vm-a2`, and so on.

```
curl http://10.1.2.99
```

   - If you add a service label (https://cloud.google.com/load-balancing/docs/dns-names) to the internal forwarding rule, you can use internal DNS (https://cloud.google.com/compute/docs/internal-dns) to contact the load balancer using its service name.

```
curl http://web-test.fr-ilb.il4.us-west1.lb.[PROJECT_ID].internal
```

Pinging the load balancer's IP address

This test demonstrates an expected behavior: You cannot ping the IP address of the load balancer. This is because internal TCP/UDP load balancers are implemented in virtual network programming — they are not separate devices (https://cloud.google.com/load-balancing/docs/internal/#how_ilb_works).

1. Connect to the client VM instance.

```
gcloud compute ssh vm-client --zone=us-west1-a
```

2. Attempt to ping the IP address of the load balancer. Notice that you do not get a response and that the `ping` command times out after 10 seconds in this example.

```
timeout 10 ping 10.1.2.99
```

Sending requests from load balanced VMs

This test demonstrates that requests to the load balancer that originate from any of the backend VMs — the server VMs being load balanced — are always answered by the same VM

making the request.

Internal TCP/UDP Load Balancing is implemented using virtual network programming and VM configuration in the guest OS. On Linux VMs, the Linux Guest Environment (https://cloud.google.com/compute/docs/instances/linux-guest-environment) performs the local configuration by installing a route in the guest OS routing table. Because of this local route, traffic to the IP address of the load balancer stays on the load balanced VM itself. (This local route is *different* from the routes in the VPC network.)

Caution: Don't rely on making requests to an internal TCP/UDP load balancer from a VM being load balanced (in the backend service for that load balancer). A request is always sent to the VM that makes the request, and health check information is ignored. Further, the backend can respond to traffic sent using protocols and destination ports other than those configured on the load balancer's internal forwarding rule.

1. Connect to a backend VM, such as `vm-a1`:

```
gcloud compute ssh vm-a1 --zone=us-west1-a
```

★ **Important:** You won't be able to connect to a backend VM in this way if you have removed its external IP address (#remove_external_ip). See connecting to instances that do not have external IP addresses (https://cloud.google.com/compute/docs/instances/connecting-advanced#sshbetweeninstances) for connection options if your backend VMs do not have external IP addresses.

2. Make a web request to the load balancer (by IP address or service name) using `curl`. Repeat the request, and note that the response is sent from the backend VM that makes the request. The expected response when testing from `vm-a1` is always: `Page served from: vm-a1`.

```
curl http://10.1.2.99
```

3. Inspect the local routing table, looking for a destination that matches the IP address of the load balancer itself, `10.1.2.99`. This route is a necessary part of Internal TCP/UDP Load Balancing, but it also demonstrates why a request from a VM behind the load balancer is always responded to by the same VM.

```
ip route show table local | grep 10.1.2.99
```

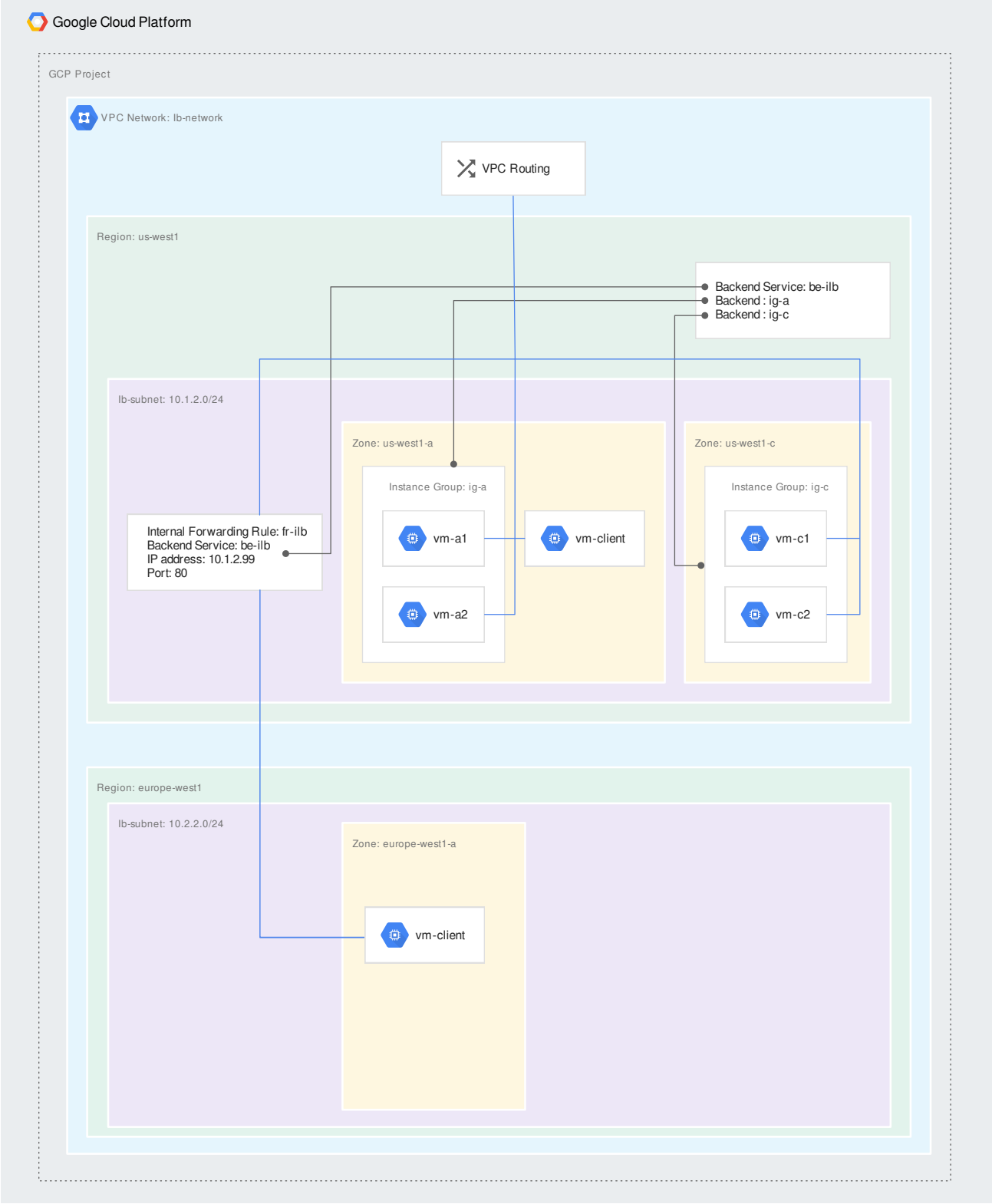
Additional configuration options

This section expands on the configuration example to provide alternative and additional configuration options. All of the tasks are optional. You can perform them in any order.

Enabling global access

With global access, client VM instances from any region can access your internal TCP/UDP load balancers.

In the following example, a client VM from the `europe-west1` region accesses the internal TCP/UDP load balancer in the `us-west1` region.



(https://cloud.google.com/load-balancing/images/ilb-global-access.svg)

Internal TCP/UDP Load Balancing with Global Access (click to enlarge)

By default, client VMs originating traffic in your VPC network must be in the same region as the IP address of the internal forwarding rule and the associated backends. You can override this default behavior by enabling global access, which allows clients from any region to access the load balancer's backends. The backend instances must still be located in a single region and in the same region as the load balancer.

Global access also expands the deployment possibilities when you access an internal TCP/UDP load balancer from VPC networks that are connected using VPC Network Peering. For more information, see Internal TCP/UDP load balancing and connected networks (https://cloud.google.com/load-balancing/docs/internal/internal-lb-and-other-networks).

To configure global access, run the following commands.

GCLOUD

1. Update your existing forwarding rule, `fr-ilb` to include the `--allow-global-access` flag.

```
gcloud beta compute forwarding-rules update fr-ilb \
   --allow-global-access
```

2. Make sure that the forwarding rule allows global access.

```
gcloud beta compute forwarding-rules describe fr-ilb
```

   The output should include: `allowGlobalAccess: true`.

3. Create a client VM in a different region than the load balancer. In this example, the client is in the `europe-west1` region.

```
gcloud compute instances create vm-client2 \
   --zone=europe-west1-a \
   --image-family=debian-9 \
   --image-project=debian-cloud \
   --tags=allow-ssh \
   --subnet=lb-subnet
```

4. Connect to the client VM instance and test connectivity.

```
gcloud compute ssh vm-client2 --zone=europe-west1-a
```

```
curl http://10.1.2.99
```

Configuring managed instance groups

The example configuration created two unmanaged instance groups (#configure_instances_and_instance_groups). You can instead use managed instance groups (https://cloud.google.com/compute/docs/instance-groups/#types_of_managed_instance_groups), including zonal and regional managed instance groups, as backends for Internal TCP/UDP Load Balancing.

Managed instance groups require that you create an instance template. This procedure demonstrates how to replace the two unmanaged zonal instance groups from the example with a single, regional managed instance group. A regional managed instance group automatically creates VMs in multiple zones of the region, making it simpler to distribute production traffic among zones.

Managed instance groups also support autoscaling (https://cloud.google.com/sdk/gcloud/reference/compute/instance-groups/managed/set-autoscaling) and autohealing (https://cloud.google.com/sdk/gcloud/reference/compute/instance-groups/managed/update). If you use autoscaling with Internal TCP/UDP Load Balancing, you cannot scale based on load balancing.

This procedure shows you how to modify the backend service for the example internal TCP/UDP load balancer so that it uses a regional managed instance group.

CONSOLE GCLOUD

Instance template

1. Go to the VM instance templates page in the Google Cloud Console.
   GO TO THE VM INSTANCE TEMPLATES PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/COMPUTE
2. Click **Create instance template**.
3. Set the **Name** to `template-vm-ilb`.
4. Choose a machine type (https://cloud.google.com/compute/docs/machine-types).
5. For **Boot disk**, click **Change**, choose *Debian GNU/Linux 9 Stretch*, then click **Select**.
6. Click **Management, security, disks, networking, sole tenancy**.
   - Click **Networking** and make the following changes:

- **Network**: `lb-network`
- **Subnet**: `lb-subnet`
- **Network tags**: `allow-ssh` and `allow-health-check`

- Click **Management**. In the **Startup script** field, copy and paste the following script contents:

```
#! /bin/bash
apt-get update
apt-get install apache2 -y
a2ensite default-ssl
a2enmod ssl
vm_hostname="$(curl -H "Metadata-Flavor:Google"

http://169.254.169.254/computeMetadata/v1/instance/name)"
echo "Page served from: $vm_hostname" |

tee /var/www/html/index.html
systemctl restart apache2
```

7. Click **Create**.

Managed instance group

1. Go to the VM instance groups page in the Google Cloud Console.

   GO TO THE VM INSTANCE GROUPS PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/COMPUTE/IN

2. Click **Create instance group**.
3. Set the **Name** to `ig-ilb`.
4. For **Location**, choose **Multi-zone**, and set the **Region** to `us-west1`.
5. Set the **Instance template** to `template-vm-ilb`.
6. (Optional) Configure autoscaling (https://cloud.google.com/compute/docs/autoscaler/). You cannot autoscale the instance group based on HTTP load balancing usage because the instance group is a backend for Internal TCP/UDP Load Balancing.
7. Set the **Minimum number of instances** to `1` and the **Maximum number of instances** to `6`.
8. (Optional) Configure **autohealing** (https://cloud.google.com/compute/docs/instance-groups/autohealing-instances-in-migs). If you configure autohealing, use the same health check used by the backend service for the Internal TCP/UDP Load Balancer. In this example, use `hc-http-80`.
9. Click **Create**.

Removing external IP addresses from backend VMs

When you created the backend VMs (#configure_instances_and_instance_groups), each was assigned an ephemeral external IP address so it could download Apache via a startup script. Because the backend VMs are only used by an internal load balancer, you can remove their external IP addresses. Removing external IP addresses prevents the backend VMs from accessing the internet directly.

Important: Removing the external IP address from a VM limits how you can connect to it. See connecting to instances that do not have external IP addresses (https://cloud.google.com/compute/docs/instances/connecting-advanced#sshbetweeninstances) for details. It's useful to leave external IP addresses on backend VMs if you need to demonstrate how requests from a backend VM to the IP address of the load balancer are handled (#test-from-backend-vms).

CONSOLE GCLOUD API

1. Go to the VM instances page in the Google Cloud Console.
   GO TO THE VM INSTANCES PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/COMPUTE/INSTANCE
2. Repeat the following steps for each backend VM.
3. Click the name of the backend VM (for example, `vm-a1`) to view the *VM instance details* page.
4. Click **Edit**.
5. In the *Network interfaces* section, click the **Edit** button.
6. From the **External IP** pop-up, choose **None**, and click **Done**.
7. Click **Save**.

Accepting traffic on multiple ports

The component that controls the port for incoming traffic is the internal forwarding rule. This procedure shows you how to replace the forwarding rule for port 80 (#configure_the_load_balancer) with one that accepts traffic on ports 80 and 443. When you create the backend VMs, the startup script that installs and configures Apache also configures it to serve HTTPS traffic on port 443. You must replace a forwarding rule because Google Cloud does not support updating forwarding rules.

Note: An internal backend service does not specify a port. For more information, see Components (https://cloud.google.com/load-balancing/docs/internal/index#components).

GCLOUD

1. Delete your existing forwarding rule, `fr-ilb`.

```
gcloud compute forwarding-rules delete fr-ilb \
    --region=us-west1
```

2. Create a replacement forwarding rule with the same name for ports 80 and 443. The other parameters for this rule, including IP address and backend service, are the same.

```
gcloud compute forwarding-rules create fr-ilb \
    --region=us-west1 \
    --load-balancing-scheme=internal \
    --network=lb-network \
    --subnet=lb-subnet \
    --address=10.1.2.99 \
    --ip-protocol=TCP \
    --ports=80,443 \
    --backend-service=be-ilb \
    --backend-service-region=us-west1
```

3. Connect to the client VM instance and test HTTP and HTTPS connections.
    - Connect to the client VM:

```
gcloud compute ssh vm-client --zone=us-west1-a
```

    - Test HTTP connectivity:

```
curl http://10.1.2.99
```

    - Test HTTPS connectivity. The `--insecure` flag is required because the Apache server configuration in the example setup uses self-signed certificates.

```
curl https://10.1.2.99 --insecure
```

    - Observe that HTTP requests (on port 80) and HTTPS requests (on port 443) are handled by all of the backend VMs.

Using session affinity

The example configuration (#configure_the_load_balancer) creates a backend service without session affinity.

This procedure shows you how to update the backend service for the example internal TCP/UDP load balancer so that it uses session affinity based on client IP addresses.

The load balancer directs a particular client's requests to the same backend VM based on a hash created from the client's IP address and the load balancer's IP address (the internal IP address of an internal forwarding rule)

CONSOLE GCLOUD API

1. Go to the Load balancing page in the Google Cloud Console.
   GO TO THE LOAD BALANCING PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/NETWORKING/LOA
2. Click **be-ilb** (the name of the backend service you created for this example) and click **Edit**.
3. On the *Edit Internal load balancer* page, click **Backend configuration**.
4. Select **Client IP** from the **Session affinity** pop-up menu.
5. Click **Update**.

For more information about using session affinity to influence traffic distribution and a description of each option, see Traffic distribution (https://cloud.google.com/load-balancing/docs/internal/index#traffic_distribution).

Creating a forwarding rule in another subnet

This procedure creates a second IP address and forwarding rule in a different subnet to demonstrate that you can create multiple forwarding rules (https://cloud.google.com/load-balancing/docs/internal/index#multiple_forwarding_rule) for one internal TCP/UDP load balancer. The region for the forwarding rule must match the region of the backend service.

Subject to firewall rules, clients in any subnet in the region can contact either internal TCP/UDP load balancer IP address.

1. Create a second subnet in the `lb-network` network in the `us-west1` region:

```
gcloud compute networks subnets create second-subnet \
    --network=lb-network \
    --range=10.5.6.0/24 \
    --region=us-west1
```

2. Create a second forwarding rule for ports 80 and 443. The other parameters for this rule, including IP address and backend service, are the same as for the primary forwarding rule, `fr-ilb`.

```
gcloud compute forwarding-rules create fr-ilb-2 \
    --region=us-west1 \
    --load-balancing-scheme=internal \
    --network=lb-network \
    --subnet=second-subnet \
    --address=10.5.6.99 \
    --ip-protocol=TCP \
    --ports=80,443 \
    --backend-service=be-ilb \
    --backend-service-region=us-west1
```

3. Connect to the client VM instance and test HTTP and HTTPS connections to the IP addresses.
    - Connect to the client VM:

```
gcloud compute ssh vm-client --zone=us-west1-a
```

    - Test HTTP connectivity to the IP addresses:

```
curl http://10.1.2.99
curl http://10.5.6.99
```

    - Test HTTPS connectivity. Use of `--insecure` is required because the Apache server configuration in the example setup uses self-signed certificates.

```
curl https://10.1.2.99 --insecure
curl https://10.5.6.99 --insecure
```

- Observe that requests are handled by all of the backend VMs, regardless of the protocol (HTTP or HTTPS) or IP address used.

What's next

- See Internal TCP/UDP Load Balancing Concepts (https://cloud.google.com/load-balancing/docs/internal/index) for important fundamentals.
- See Failover concepts for Internal TCP/UDP Load Balancing (https://cloud.google.com/load-balancing/docs/internal/failover-overview) for important information about failover.
- See Internal Load Balancing and DNS Names (https://cloud.google.com/load-balancing/docs/dns-names) for available DNS name options your load balancer can use.
- See Configuring failover for Internal TCP/UDP Load Balancing (https://cloud.google.com/load-balancing/docs/internal/setting-up-failover) for configuration steps and an example internal TCP/UDP load balancer failover configuration.
- See Internal TCP/UDP Load Balancing Logging and Monitoring (https://cloud.google.com/load-balancing/docs/internal/internal-logging-monitoring) for information on configuring Stackdriver logging and monitoring for Internal TCP/UDP Load Balancing.
- See Internal TCP/UDP Load Balancing and Connected Networks (https://cloud.google.com/load-balancing/docs/internal/internal-lb-and-other-networks) for information about accessing internal TCP/UDP load balancers from peer networks connected to your VPC network.
- See Troubleshooting Internal TCP/UDP Load Balancing (https://cloud.google.com/load-balancing/docs/internal/troubleshooting-ilb) for information on how to troubleshoot issues with your internal TCP/UDP load balancer.