Load Balancer

22 June 2021 09:17

Load Balancer Feature: https://cloud.google.com/load-balancing/docs/features
Bullet Points: https://jayendrapatil.com/category/gcp/cloud-load-balancing/

HTTPS Load Balancer

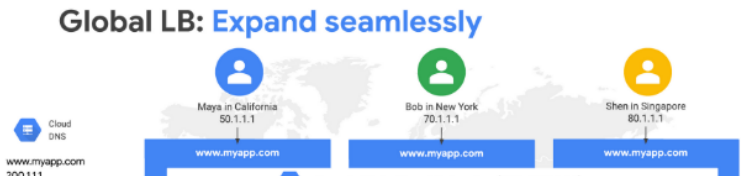
Key Points

External HTTP -> Global (premium network tier), proxy, support(cdn, cloud armor), anycast ip, GFE, native support for web socket, connection draining supported, mutual tls not supported, only port 80,443 and 8080 supported, balancing mode (utilization , rate)
TCP Proxy -> Proxy, ipv4,v6(premium not for udp), global (premium network tier), GFE,balancing mode (utilization , connection)
SLL Proxy -> same as tcp proxy but with ssl offload setup, GFE,balancing mode (utilization , connection)
Network LB -> only ipv4, use regional backend but global presence, pass through, Maglev directly result sent to client from lb, tcp and udp supported, connection tracking and configurable hashing for backend routing, does not support neg, udp health check not supported, support tcp, http, https , ssl protocol for health check
Internal http: no support for cdn cloud armor, proxy, regional ipv4 supported, only port 80,443 and 8080 supported, NEG containing GKE and vm
Andromeda (proxy), websocket support
Internal tcp: pass through, ipv4, regional, tcp udp, Andromeda

- Health check supported in all lb
- Backend type: (zonal, serverless internet NEG), Instance group, bucket
- Firewall must allow for health check
- Session affinity supported based on backend health in all load balancer for hhtps cookie based and client ip,port,protocol hash and for other client ip,port,protocol hash. Session affinity not supprted for udp protocol

Global Load Balancer

Single Anycast IP: For our global load-balancing solution, we pushed load balancing to the edge of Google's global network to front end the global load-balancing capacity behind a single Anycast Virtual IPv4 or IPv6 address



Anycast directs packets to the geographically closest server based on Border Gateway Protocol (BGP) paths

Google Front End Service

When a service wants to make itself available on the Internet, it can register itself with an infrastructure service called the Google Front End (GFE). The GFE ensures that all TLS connections are terminated using correct certificates and following best practices such as supporting perfect forward secrecy. The GFE additionally applies protections against Denial of Service attacks (which we will discuss in more detail later). The GFE then forwards requests for the service using the RPC security protocol discussed previously.

In effect, any internal service which chooses to publish itself externally uses the GFE as a smart reverse-proxy front end. This front end provides public IP hosting of its public DNS name, Denial of Service (DoS) protection, and TLS termination. Note that GFEs run on the infrastructure like any other service and thus have the ability to scale to match incoming request volumes.

Routing: With global load balancing, you get cross-region failover and overflow. Global LB's traffic distribution algorithm automatically directs traffic to the next closest instance with available capacity in the event of failure of or lack of capacity for instances in the region closest to end user.

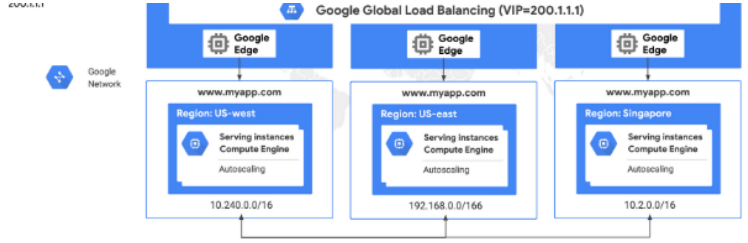
NEG: For containers, we built an abstraction called Network Endpoint Groups (NEG), which is essentially a group of IP address and port pairs. NEGs enable you to directly specify a container endpoint as opposed to first directing traffic to the node on which it resides and then redirecting to the container using kube-proxy

https://cloud.google.com/blog/products/networking/google-cloud-networking-in-depth-cloud-load-balancing-deconstructed

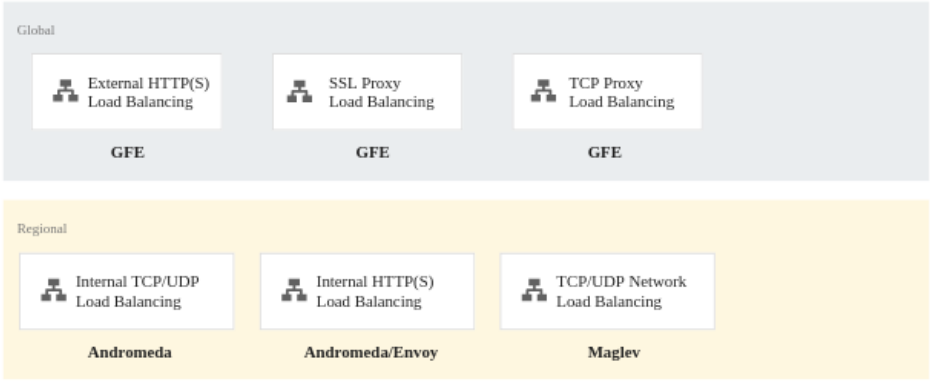
Global versus regional load balancing

Use **global load balancing** when your backends are distributed across multiple regions, your users need access to the same applications and content, and you want to provide access by using a single anycast IP address. Global load balancing can also provide IPv6 termination.

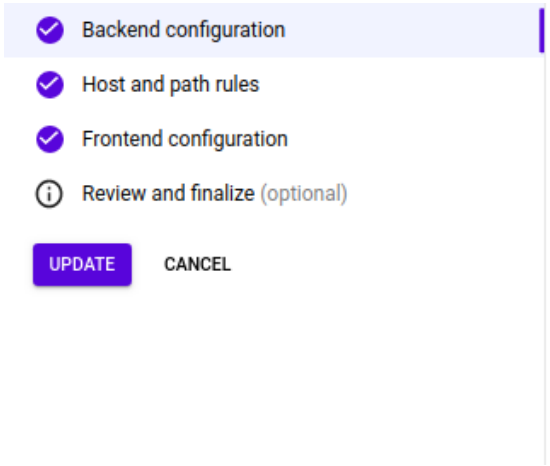
Use **regional load balancing** when your backends are in one region, and you only require IPv4 termination.



Technology used for different kid of load balancer

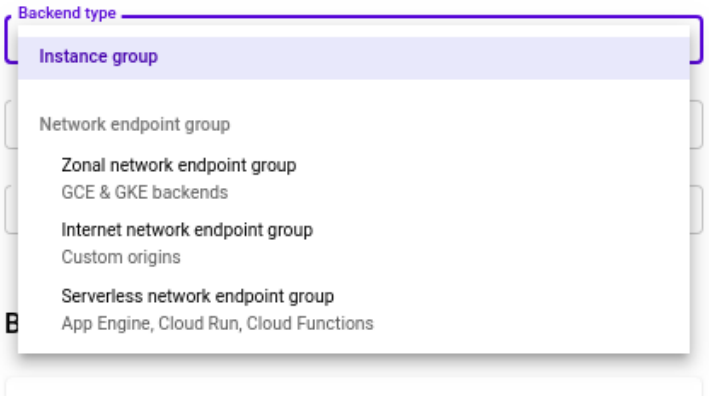


Load Balancer Edit Tab



Backend Service

Backend Type

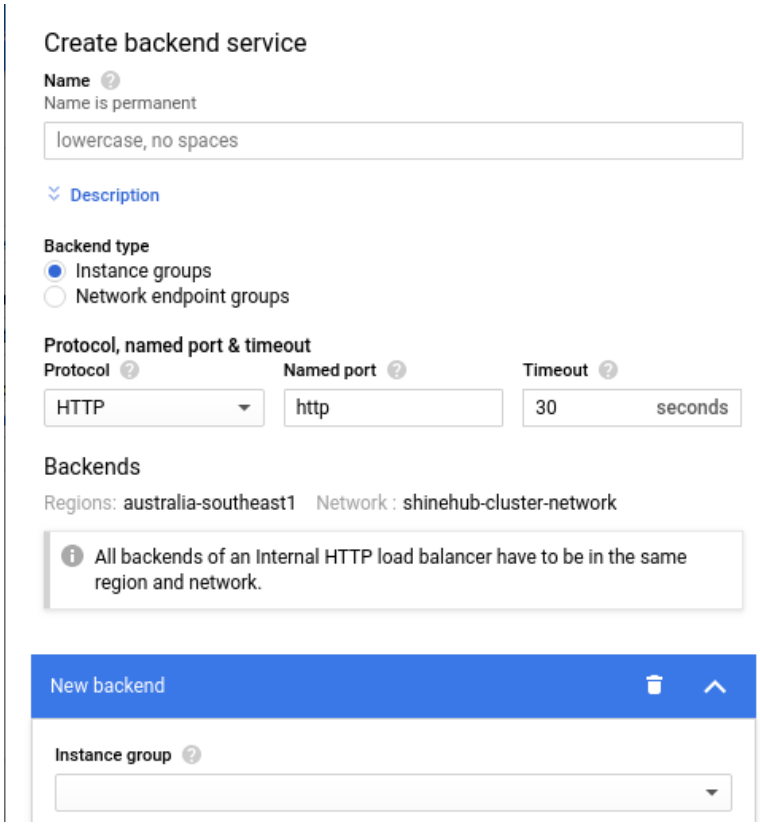


Named Port: A backend service sends traffic to its backends through a named port. The port name is mapped to a port number in each instance group.

Timeout: How long to wait for the backend service to respond before considering it a failed request

Balancing Mode : Determines how the load balancer distributes requests among backend instance groups or NEGs. For HTTP(S), supported balancing modes are utilization (for instance group backends) and rate (for instance groups and NEGs). When a backend has reached a configured maximum utilization or rate, the load balancer will direct new requests to other backends that have not reached their configured maximums. If all backends have reached their configured maximums, the load balancer will exceed them.

Internal HTTP Load Balancer



- RATE, for instance groups or NEGs, is the target maximum number of requests (queries) per second (RPS, QPS). The target maximum RPS/QPS can be exceeded if all backends are at or above capacity.
- UTILIZATION is the backend utilization of VMs in an instance group.

Q & A

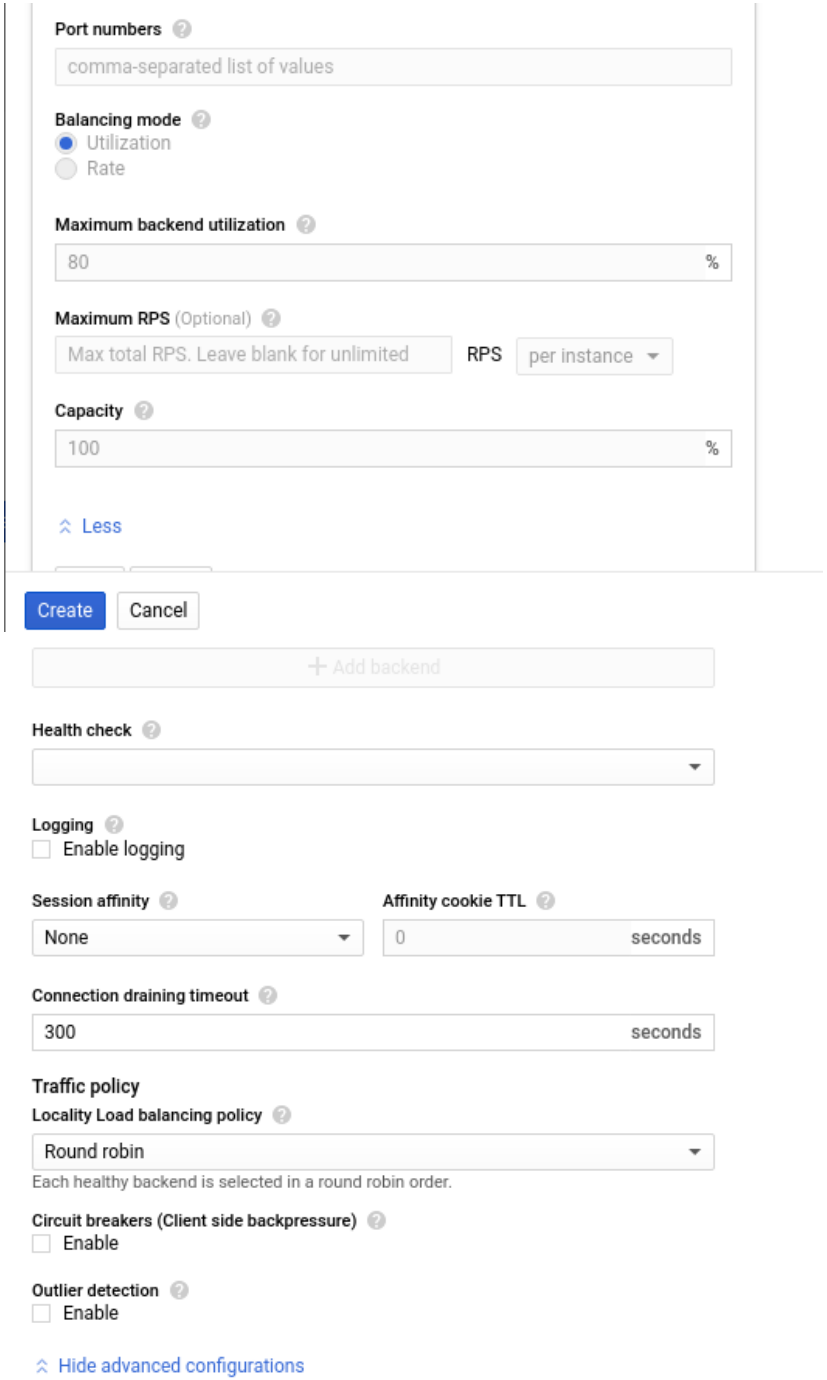
Q. To ensure that your application will handle the load even if an entire zone fails, what should you do? Select all correct options.
A. Don't select the "Multizone" option when creating your managed instance group.
B. Spread your managed instance group over two zones and overprovision by 100%. (for Two Zone)right
C. Create a regional unmanaged instance group and spread your instances across multiple zones.wrong
D. Overprovision your regional managed instance group by at least 50%. (for Three Zones)right

A. B is correct if one zone fails you still have 100% desired capacity in another zone

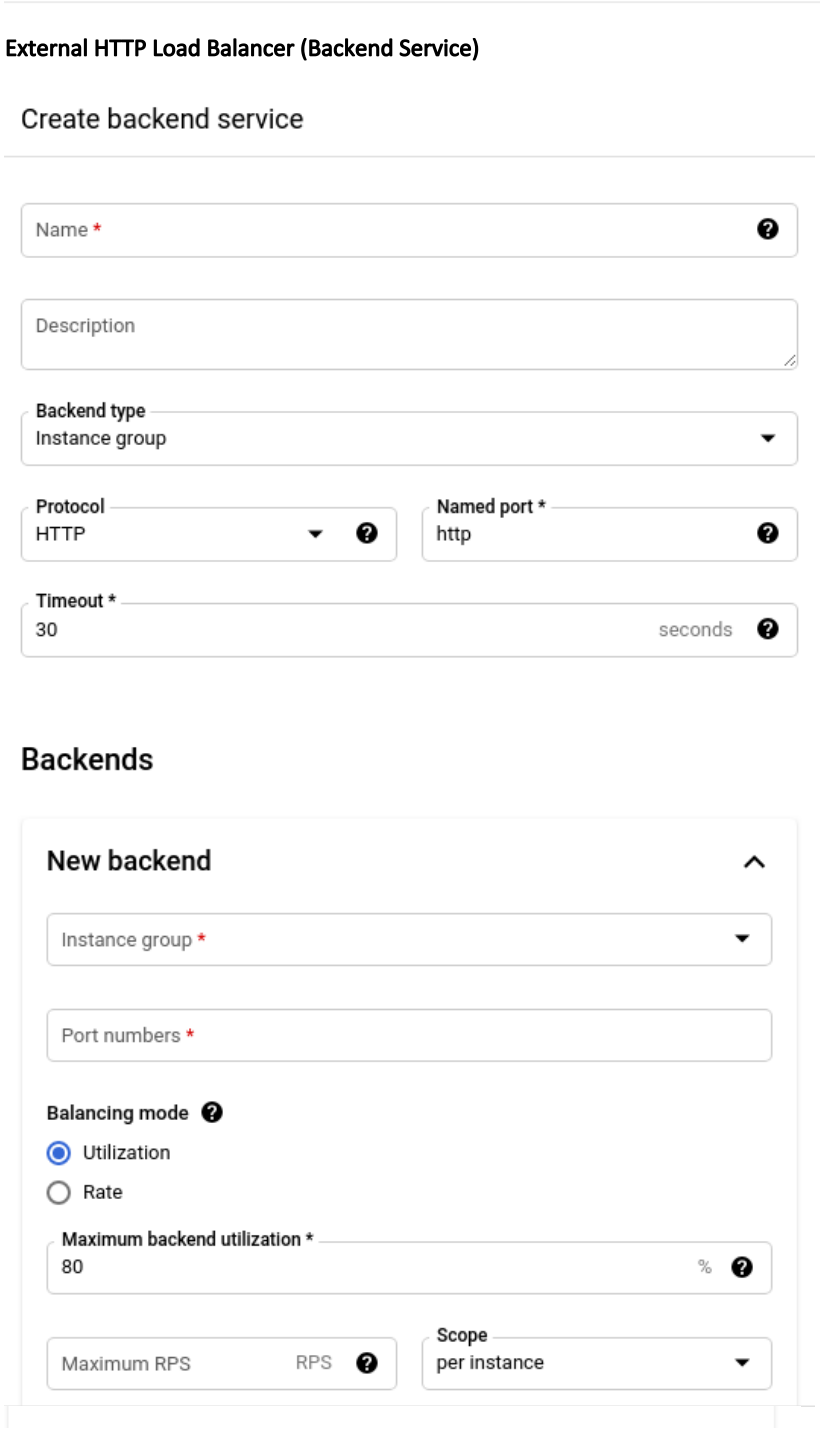
D is correct since you have at least total 150% desired capacity spread over 3 zones, each zone has 50% capacity. You'll have 100% desired capacity in two zones if any single zone failed at given time.

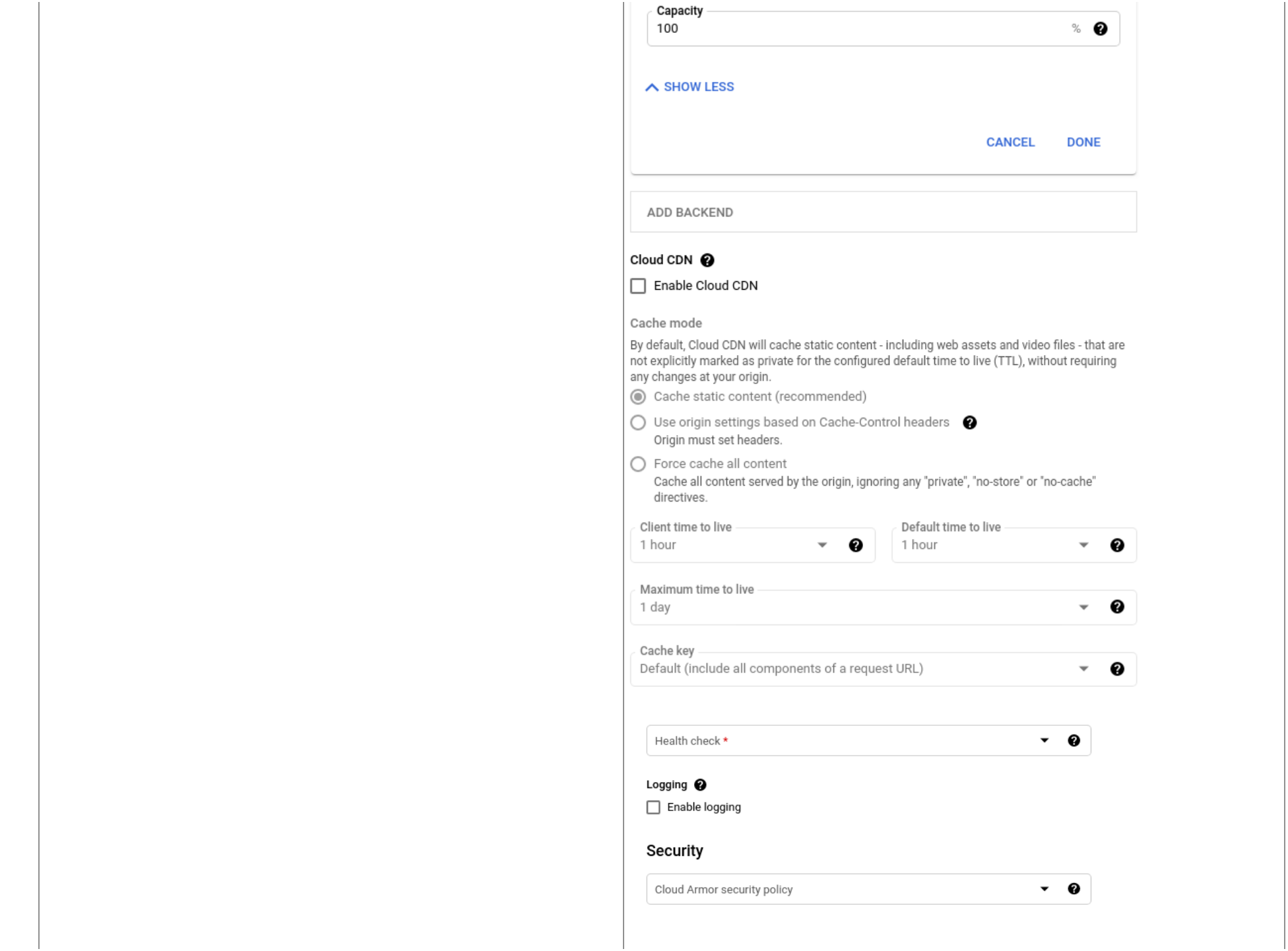
Reference Resources

https://cloud.google.com/compute/docs/instance-groups/distributing-instances-with-regional-instance-groups



External HTTP Load Balancer (Backend Service)




Capacity
100
%
SHOW LESS
CANCEL
DONE
ADD BACKEND
Cloud CDN
Enable Cloud CDN
Cache mode
By default, Cloud CDN will cache static content - including web assets and video files - that are not explicitly marked as private for the configured default time to live (TTL), without requiring any changes at your origin.
Cache static content (recommended)
Use origin settings based on Cache-Control headers
Origin must set headers.
Force cache all content
Cache all content served by the origin, ignoring any "private", "no-store" or "no-cache" directives.
Client time to live
1 hour
Default time to live
1 hour
Maximum time to live
1 day
Cache key
Default (include all components of a request URL)
Health check *
Logging
Enable logging
Security
Cloud Armor security policy

Additional CDN options

Serve while stale
Disable serve while stale

Serve while stale allows Cloud CDN to return an expired (stale) cached object on a cache miss or origin error.

Restricted content

- (•) Allow public access to my content cached by Cloud CDN (recommended)
- ( ) Restrict access using signed URLs and signed cookies

Negative Caching Policy

- [ ] Enable negative caching

Negative caching allows per-status code cache TTLs to be set, in order to apply fine-grained caching for common errors or redirects. This can reduce the load on your origin and improve the end-user experience by reducing response latency.

+ ADD NEGATIVE CACHING POLICY

Bypass cache on request header

+ ADD HEADER

Session affinity
None

Affinity cookie TTL
0 seconds

Connection draining timeout
300 seconds

Custom request headers

+ ADD HEADER

Custom response headers

+ ADD HEADER

^ HIDE ADVANCED CONFIGURATIONS

Create Backend Bucket/NEG

Cache Mode: Origin Setting : Cloud CDN will only cache responses with valid cache directives in the response headers, such as 'Cache-Control: public, max_age=3600'.

Client time to live: Client time to live allows you to set a shorter time to live for browsers or clients, and to have those clints revalidate content against Cloud CDN on a more regular basis, without requiring revalidation at the origin. The value of client time to live cannot be greater than maximum time to live, but can be equal.

Default time to live: Specifies the default time to live for cached content served by this origin for responses that do not have an existing valid time to live. The value of default time to live cannot be set to a value grater than that of maximum time to live, but can be equal.

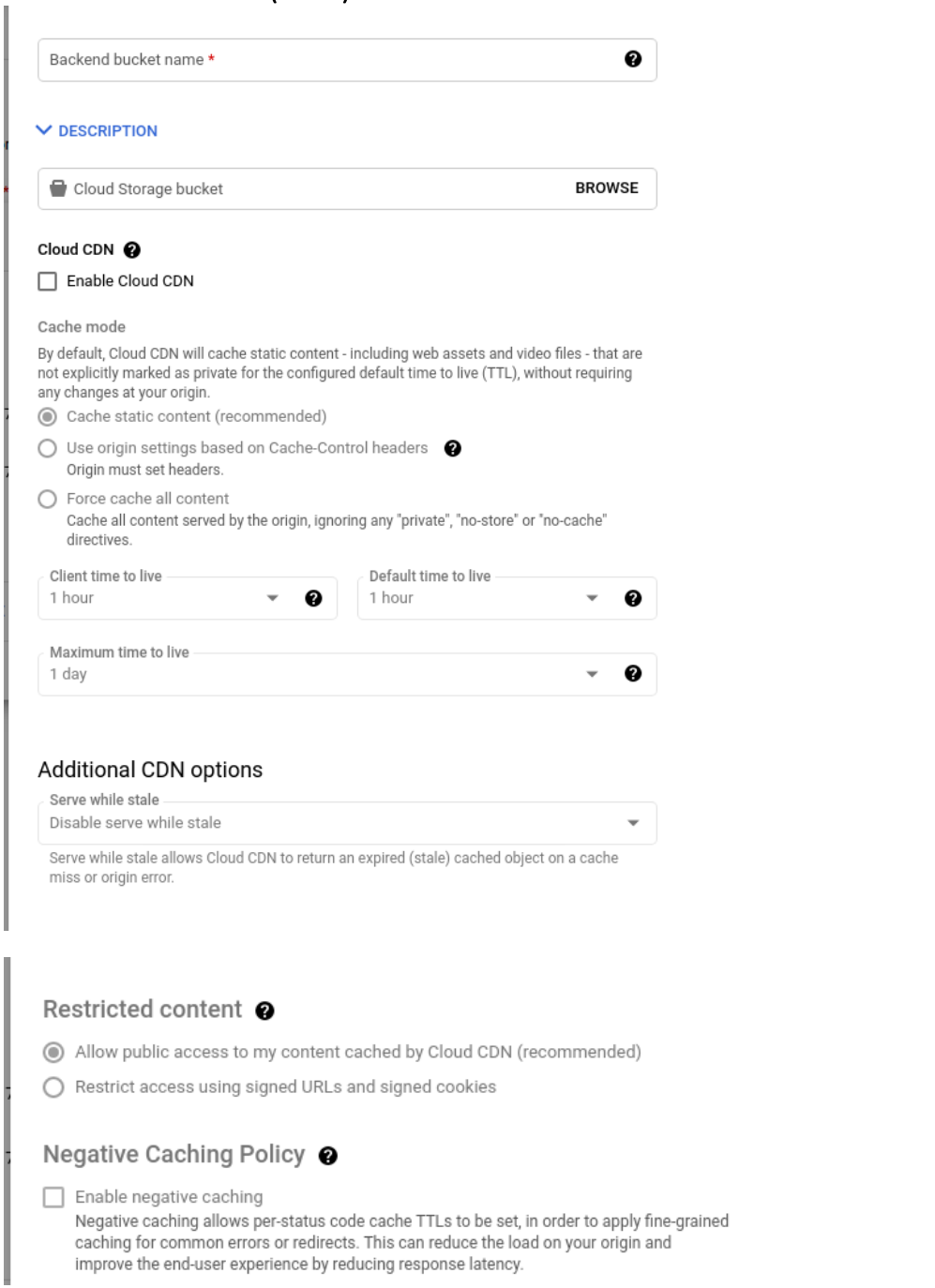
Maximum time to live: Maximum time to live allows you to set the maximum time for cached content served by this origin.

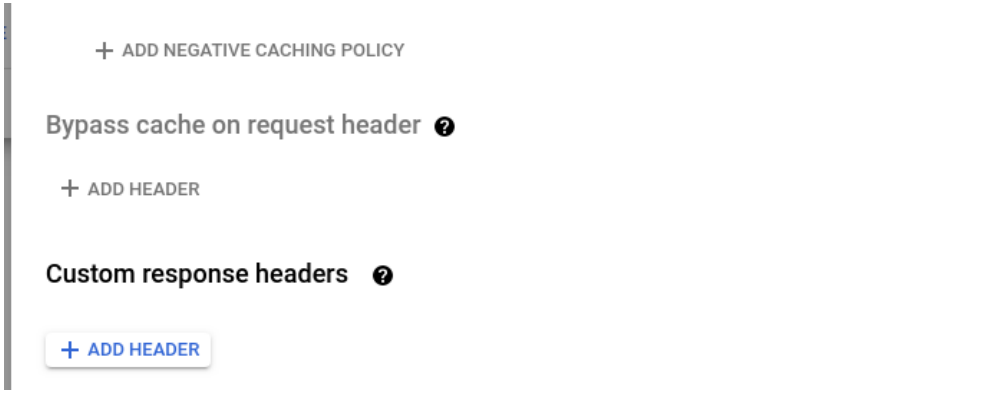
Cloud CDN: Cloud CDN caches HTTP(S) content closer to your users so content delivery is faster while also reducing serving costs. For more CDN options, go to the Cloud CDN page.

Cache Key: Each cache entry is identified by a cache key. By default, cache keys include all components of a request URL, which means similar URLs may not share cache entries. For example, a request for http://example.com/cat.jpg won't use the cache entry for http://example.com/cat.jpg?1234. To share cache entries, use a custom cache key.

Security
Security policies let you control access to your Google Cloud Platform resources at your network's edge.

External HTTP Load Blancer (Bucket)





Create Backend Bucket: Additional CDN Options

Restricted Content: Signed URLs and cookies allow you to restrict access to viewers who are authorized, by providing users with a time-limited URL or cookie that grants access for its duration. Users who otherwise try to access the content receive an HTTP 403 (unauthorized) error.

After you generate a signed URL, anyone who possesses it can use the signed URL to perform specified actions (such as reading an object) within a specified period of time.

Negatively Caching policy: Configure the cache TTL for cacheable 3xx, 4xx and 5xx error codes.
When the cache mode is set to CACHE_ALL_STATIC or USE_ORIGIN_HEADERS, these values apply to responses with the specified response code that lack any cache-control, expires, or pragma: no-cache headers.
When the cache mode is set to FORCE_CACHE_ALL, they apply to all responses with the specified response code, and override any caching headers.

Bypass cache on request header: Bypass the cache when the specified request headers are matched, for example, Pragma or Authorization headers. Indicate the header field name to match on when bypassing cache. Values are case-insensitive

Custom response header:
Headers that the HTTP(S) load balancer should add to proxied responses. For the list of headers, see the documentation

Connection draining timeout
The number of seconds a draining instance will wait for in-flight connections to complete. Instances do not accept new requests during a connection drain.

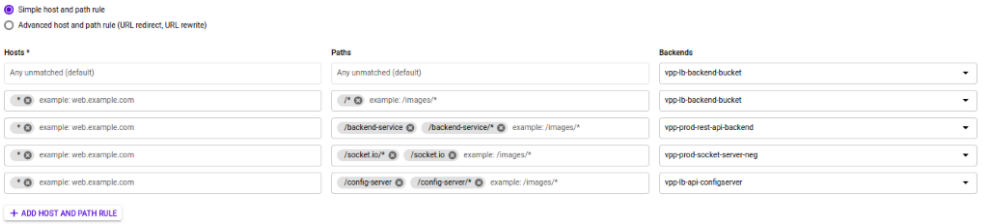
Session affinity

Session affinity determines how requests are routed to your backend instances. Learn more
None: New requests are routed to any instance, but traffic for a given connection will be routed to the same instance.
Client IP: All connections from a client are routed to the same instance regardless of protocol.
Generated cookie: Requests are routed based on a GCLB cookie generated by the load balancer. A GCLB cookie is sent to the client on the first request, and subsequent requests with that same cookie will be directed to the same instance. This lets the load balancer distinguish between clients using the same IP address.

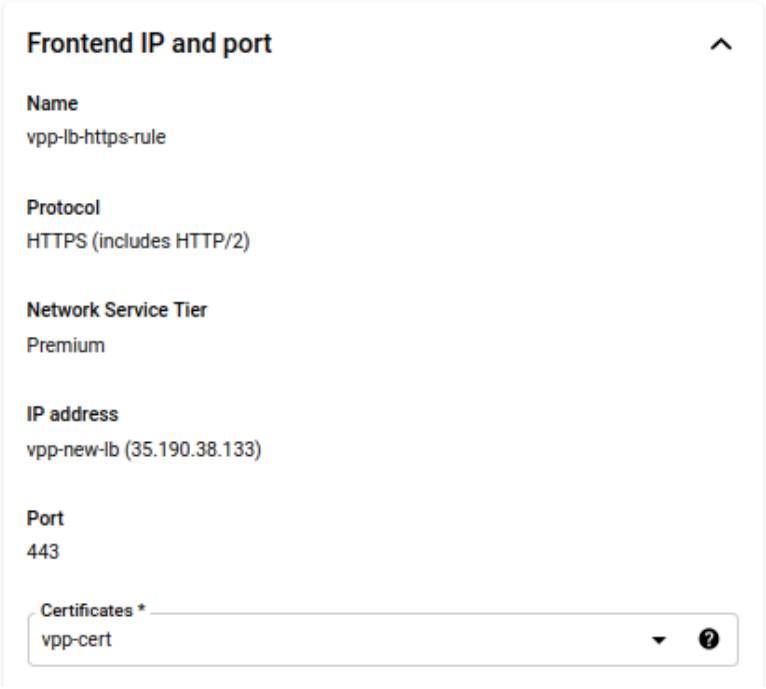
Affinity cookie TTL

The lifetime of the GCLB generated cookie (in seconds). If 0, then the cookie is a non-persistent session cookie.

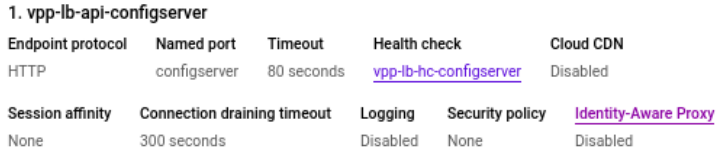
Host and path rue



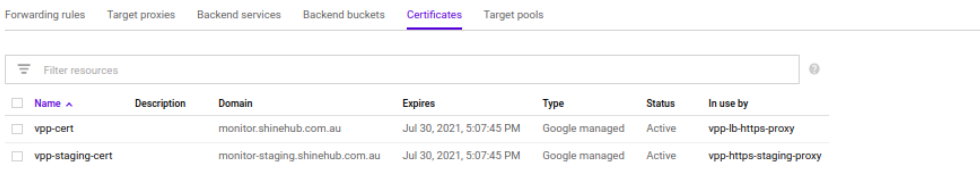
Frontend Configuration



Configuration option for backend service



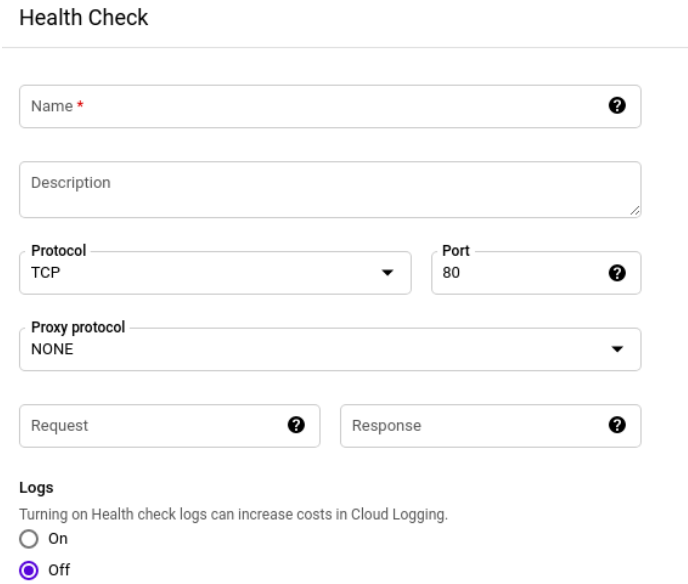
Load balancer advance menu provide more option

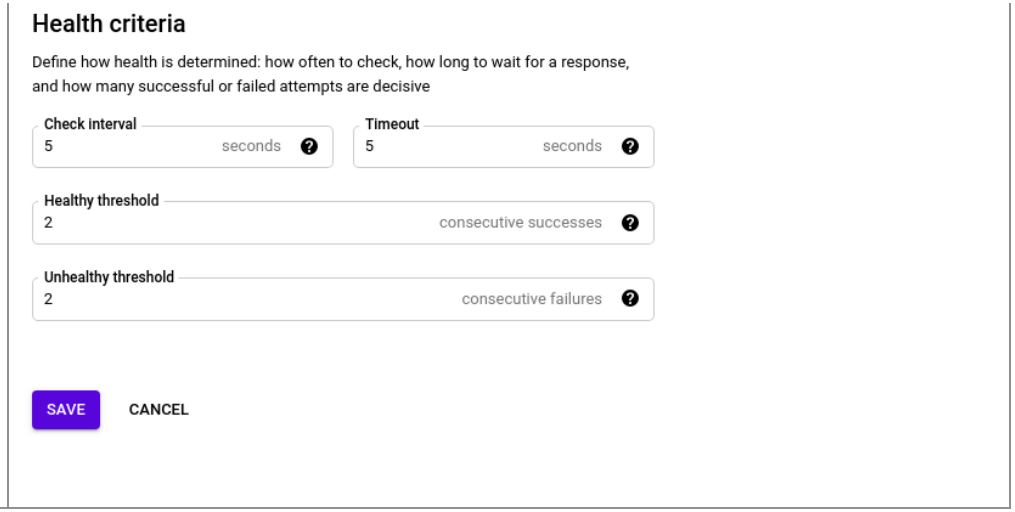


Health check creation page

A: Load balancing health checks help direct traffic away from non-responsive instances and toward healthy instances; these health checks **do not cause Compute Engine to recreate instances.**

C: Application-based autohealing improves application availability by relying on a health checking signal that detects application-specific issues such as freezing, crashing, or overloading. If a health check determines that an application has failed on an instance, the group **automatically recreates that instance**.



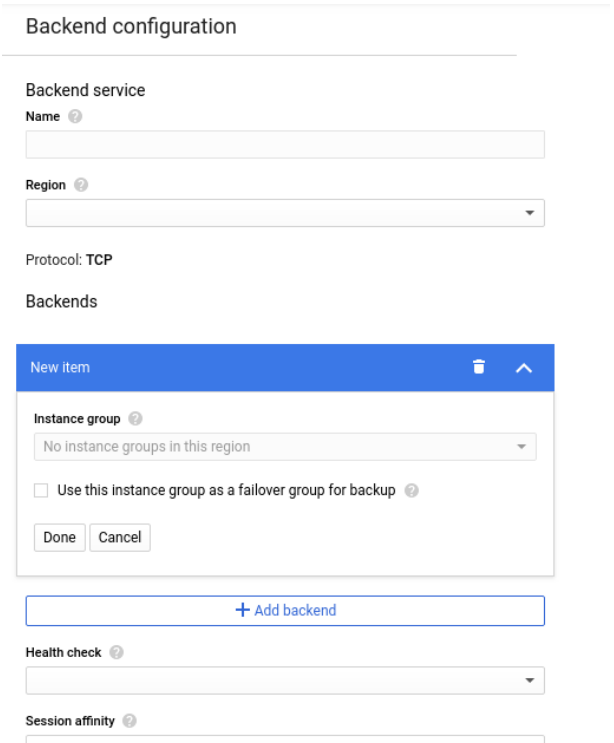


TCP Load Balancer

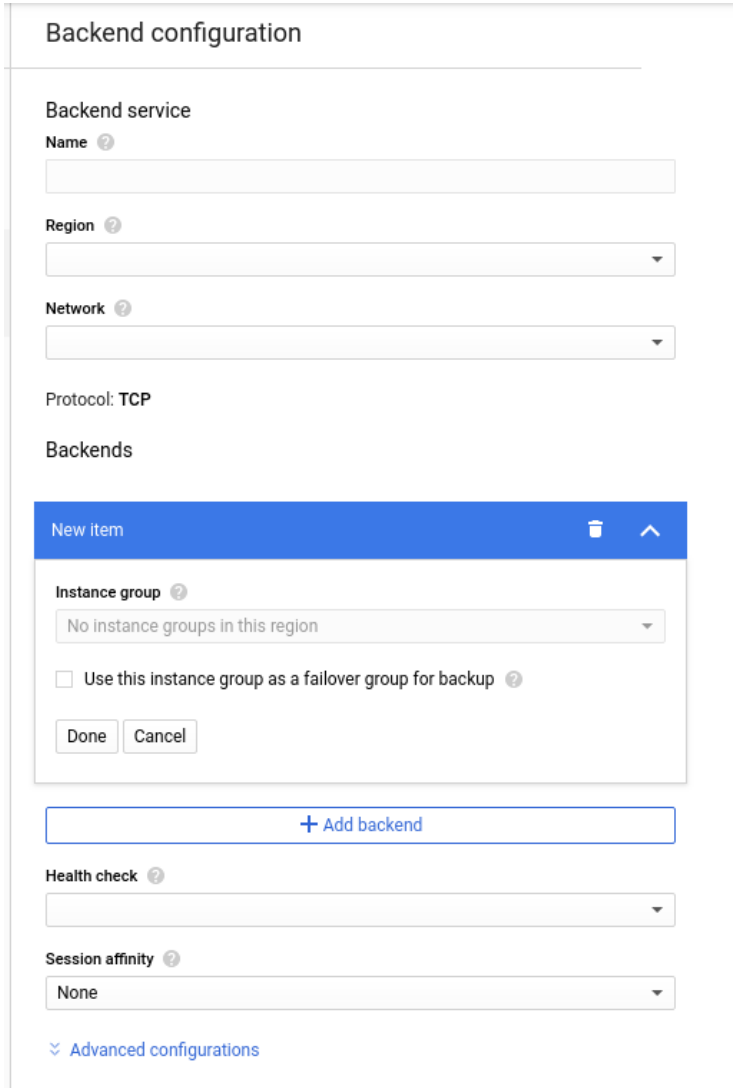
supports two types of balancing mode :

- CONNECTION : the load is spread based on how many concurrent connections the backend can handle.
- UTILIZATION: the load is spread based on the utilization of instances in an instance group.

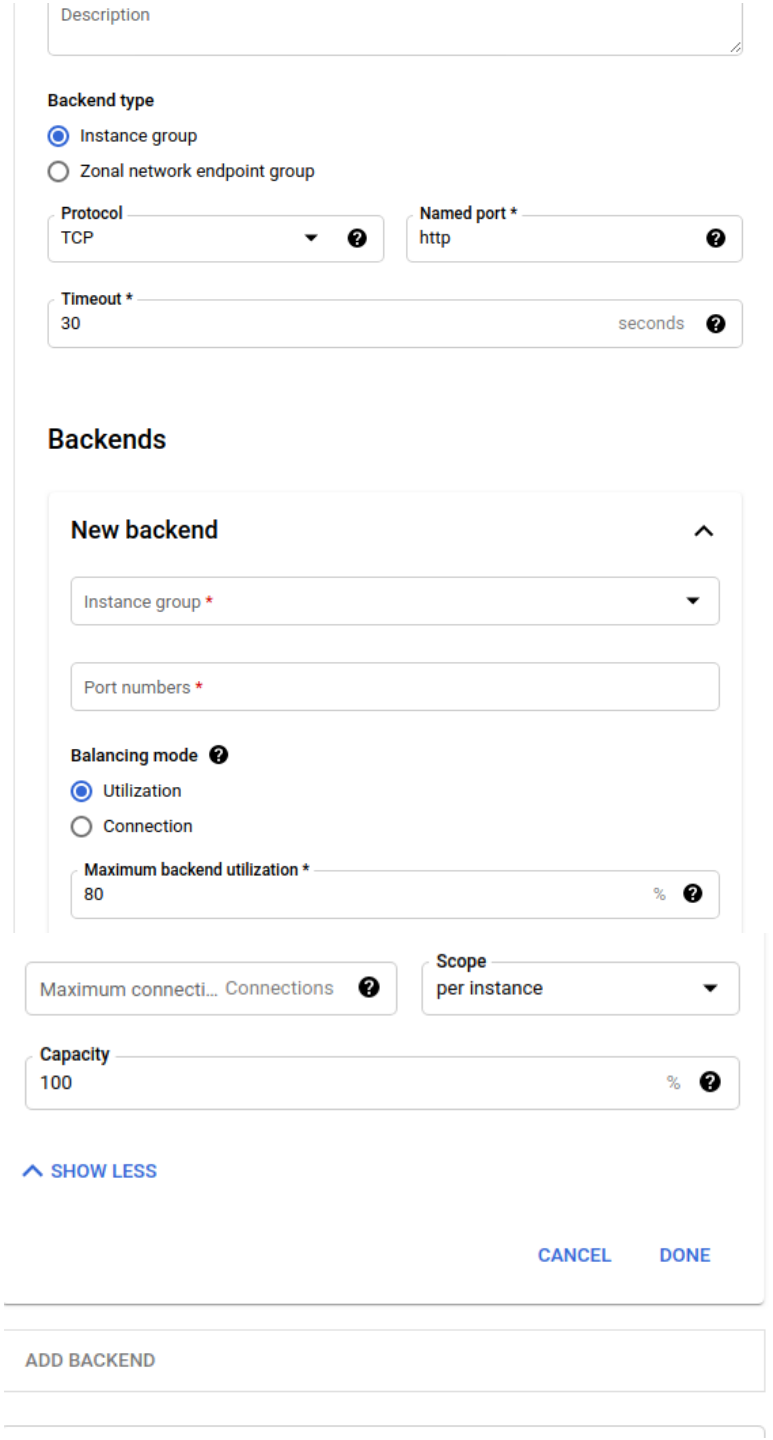
External TCP Load balancer backend config (TCP Network)

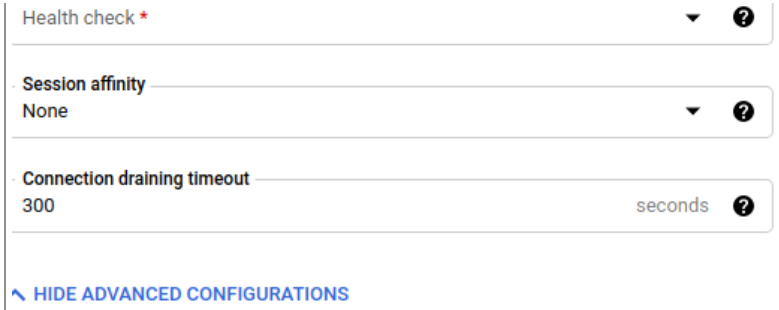


Internal TCP Load balancer config



TCP/SSL Proxy Load Balancer


Description
Backend type
Instance group
Zonal network endpoint group
Protocol
TCP
Named port *
http
Timeout *
30
seconds
Backends
New backend
Instance group *
Port numbers *
Balancing mode
Utilization
Connection
Maximum backend utilization *
80
%
Maximum connecti... Connections
Scope
per instance
Capacity
100
%
SHOW LESS
CANCEL
DONE
ADD BACKEND



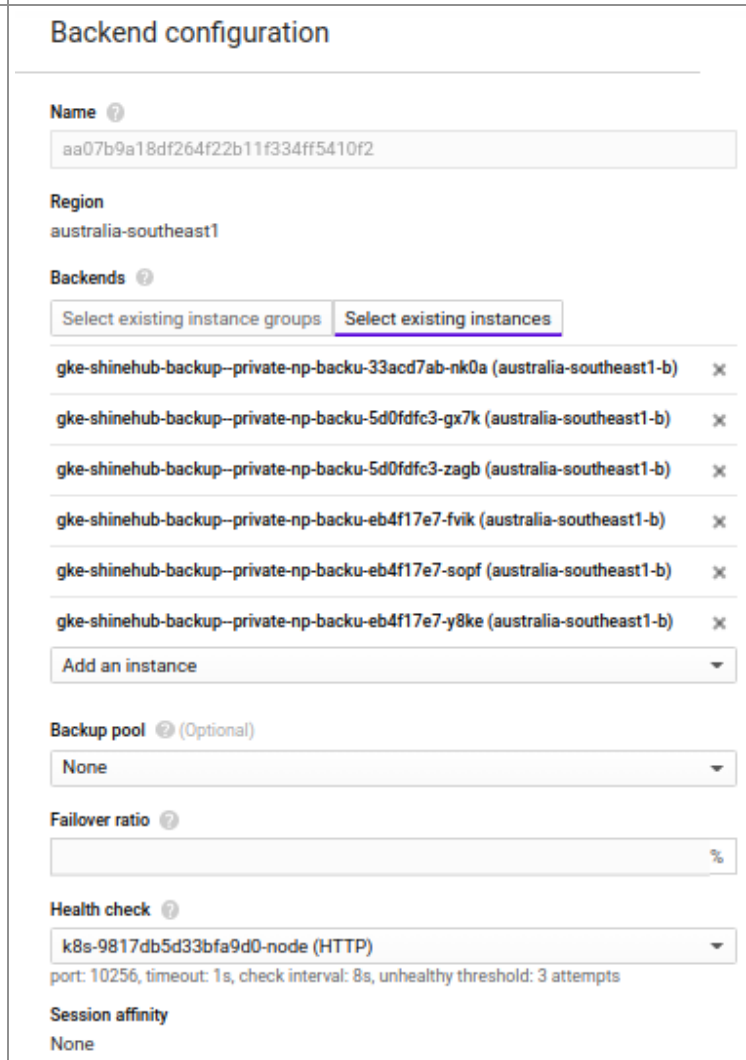
Backend Configuration

Backends: Select from template-based instance groups or VM instances in the target pool's region

Backup pool: The backup target pool handles traffic if the health of the primary target pool falls below the failover ratio.

Failover ratio: The percentage of healthy VM instances below which a failover to the backup target pool is triggered

Health check: A health check determines whether instances in the target pool are healthy. If 'No health check', traffic is distributed among all instances



Frontend Configuration

SSL Certificate

Google Cloud uses SSL certificates to provide privacy and security from a client to a load balancer. To achieve this, the load balancer must have an SSL certificate and the certificate's corresponding private key. Communication between the client and the load balancer remains private—illegible to any third party that doesn't have this private key.

Use multiple SSL certificates when you are serving from multiple domains using the same load balancer IP address and port, and you want to use a different SSL certificate for each domain.

When you specify more than one SSL certificate, the first certificate in the list of SSL certificates is considered the primary SSL certificate associated with the target proxy.

When a client sends a request, the load balancer uses the SNI hostname specified by the client to select the certificate to use in negotiating the SSL connection.

SSL Policy

SSL policies give you the ability to control the features of SSL that your Google Cloud SSL proxy load balancer or external HTTP(S) load balancer negotiates with clients. In this document, the term *SSL* refers to both the SSL and TLS protocols.

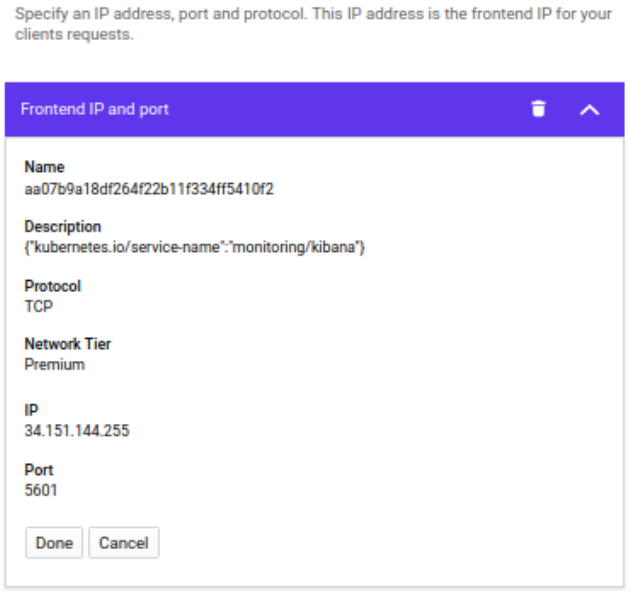
SSL policies help control the features of SSL like SSL versions and ciphers that the load balancer negotiates with clients.

By default, HTTP(S) Load Balancing and SSL Proxy Load Balancing use a set of SSL features that provides good security and wide compatibility. Some applications require more control over which SSL versions and ciphers are used for their HTTPS or SSL connections. You can define SSL policies to control the features of SSL that your load balancer negotiates with clients.

To define an SSL policy, you specify a minimum TLS version and a profile. The profile selects a set of SSL features to enable in the load balancer.

The three pre-configured profiles are as follows:

- **COMPATIBLE.** Allows the broadest set of clients, including clients that support only out-of-date SSL features, to negotiate SSL with the load balancer.
- **MODERN.** Supports a wide set of SSL features, allowing modern clients to negotiate SSL.
- **RESTRICTED.** Supports a reduced set of SSL features, intended to meet stricter compliance requirements.



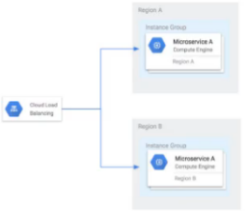
Multi Regional Load Balancer

We define region or only zone of an instance group (Load balancer backend) in the instance group. Then we can attach multiple instance group of different region to load balancer.

Load Balancing Across MIGs in Multiple Regions

28 Minutes

- **Regional MIG** can distribute instances in different zones of a single region
  - Create multiple Regional MIGs in different regions (in the same project)
- **HTTP(S) Load Balancing** can distribute load to the multiple MIGs behind a single external IP address
  - User requests are redirected to the nearest region (Low latency)
- Load balancing sends traffic to **healthy instances**:
  - If health check fails instances are restarted:
    - (REMEMBER) Ensure that health check from load balancer can reach the instances in an instance group (Firewall rules)
  - If all backends within a region are unhealthy, traffic is distributed to healthy backends in other regions



Resilience architecture

Resiliency for Compute Engine & Load Balancing

28 Minutes

- **Resiliency** - "Ability of system to provide acceptable behavior even when one or more parts of the system fail"
- Build Resilient Architectures
  - Run VMs in MIG behind global load balancing
- Have the right data available
  - Use Cloud Monitoring for monitoring
  - Install logging agent to send logs to Cloud Logging
- Be prepared for the unexpected (and changes)
  - Enable Live Migration and Automatic restart when available
  - Configure the right **health checks**
  - **(Disaster recovery)** Upto date image copied to multiple regions
- We will talk about resiliency as we go further!

Google Compute Engine and Load Balancing Security

Compute Engine & Load Balancing for Architects

2 Mi

Security

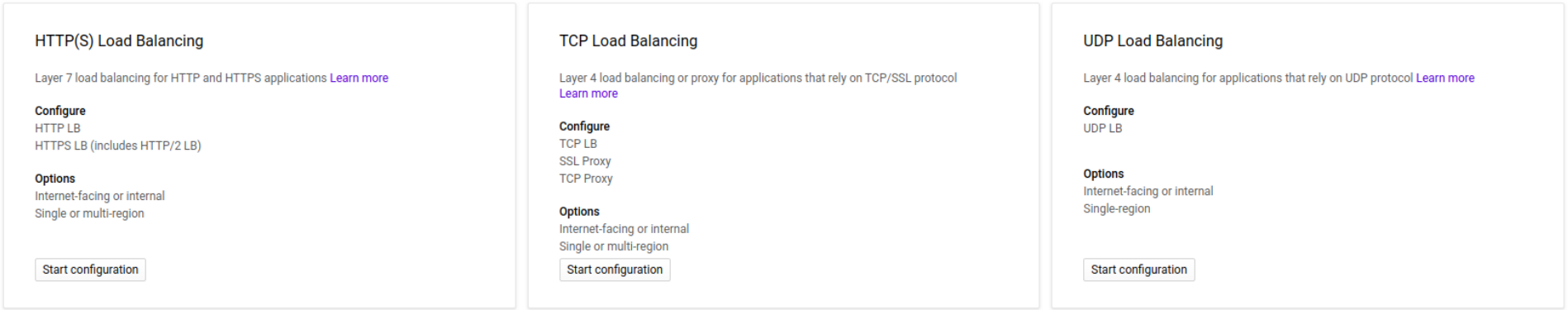
- Use **Firewall Rules** to restrict traffic
- Use **Internal IP Addresses** as much as possible
- Use **Sole-tenant nodes** when you have regulatory needs

- Create a hardened **custom image** to launch your VMs

Performance

- Choose right **Machine Family** for your workload
- Use GPUs and TPUs to increase performance
  - Use GPUs to accelerate machine learning and data processing workloads
  - Use TPUs for massive matrix operations performed in your machine learning workloads
- Prefer creating a hardened **custom image** to installing software at startup

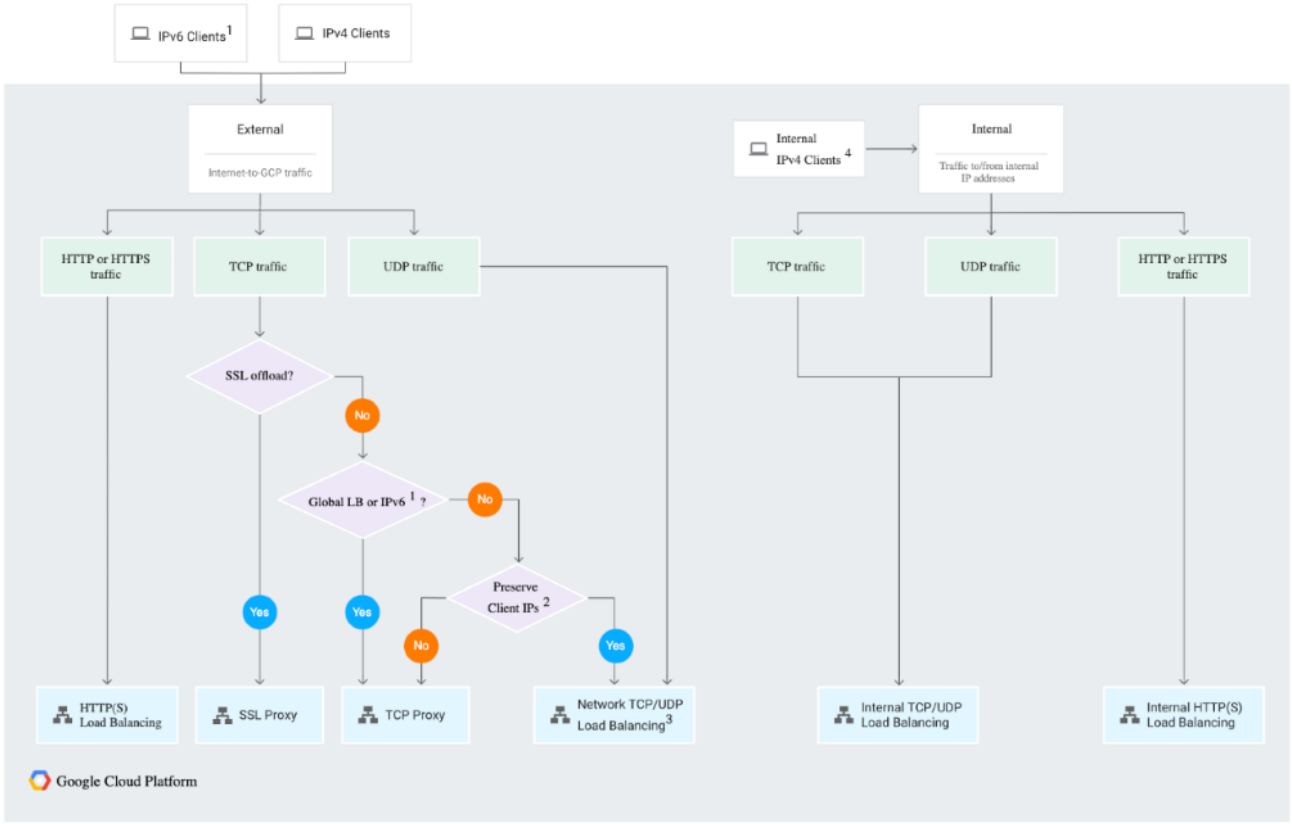
Different Load Balancer Options



SSL Proxy Load Balancer To secure your service, we recommend taking a defense-in-depth approach. We also recommend that you deploy TLS for data privacy and integrity purposes. We do not charge extra for encrypted vs. unencrypted traffic. We offer HTTPS and SSL proxy in our global load-balancing family. We also offer Managed Certificates to reduce the work of procuring certs and managing their lifecycle	

Explanation : https://cloudacademy.com/course/planning-configuring-google-cloud-platform-solution/load-balancing-with-gcp/

Load Blancer Selection



[1] IPv6 clients are supported for TCP traffic if you configure the load balancer in Premium Tier. IPv6 clients aren't supported for UDP traffic.
[2] Another reason to choose Network TCP/UDP Load Balancing is if you need to ensure that the load balancer is located in a particular region.
[3] Network TCP/UDP load balancers use regional external IP addresses that are accessible by clients anywhere.
[4] Clients in a VPC network or in a network connected to a VPC network.

Internal or external	Regional or global	Supported network tiers	Proxy or pass-through	Traffic type	Load balancer type
Internal	Regional	Premium only	Pass-through	TCP or UDP	Internal TCP/UDP
	Regional	Premium only	Proxy	HTTP or HTTPS	Internal HTTP(S)
External	Regional	Premium or Standard	Pass-through	TCP, UDP, ESP, or ICMP (Preview)	External TCP/UDP Network
	Global in Premium Tier	Premium or Standard	Proxy	TCP	TCP Proxy
	Effectively regional[1] in Standard Tier	Premium or	Proxy	SSL	SSL Proxy

	Standard			
	Premium or Standard	Proxy	HTTP or HTTPS	External HTTP(S)

Load balancer type	Traffic type	Preserve Client IP	Global or regional	Load balancing scheme	Load balancer destination ports	Proxy or pass-through
External HTTP(S)	HTTP or HTTPS	No	Global*	EXTERNAL	HTTP on 80 or 8080; HTTPS on 443	Proxy
Internal HTTP(S)	HTTP or HTTPS	No	Regional	INTERNAL_MANAGED	HTTP on 80 or 8080; HTTPS on 443	Proxy
SSL Proxy	TCP with SSL offload	No	Global*	EXTERNAL	25, 43, 110, 143, 195, 443, 465, 587, 700, 993, 995, 1883, 3389, 5222, 5432, 5671, 5672, 5900, 5901, 6379, 8085, 8099, 9092, 9200, and 9300	Proxy
TCP Proxy	TCP without SSL offload	No	Global*	EXTERNAL	25, 43, 110, 143, 195, 443, 465, 587, 700, 993, 995, 1883, 3389, 5222, 5432, 5671, 5672, 5900, 5901, 6379, 8085, 8099, 9092, 9200, and 9300	Proxy
External Network TCP/UDP	TCP, UDP, ESP, or ICMP (Preview)	Yes	Regional	EXTERNAL	Any	Pass-through
Internal TCP/UDP	TCP or UDP	Yes	Regional backends, regional frontends (global access supported)	INTERNAL	Any	Pass-through

Gcloud(https://www.qwiklabs.com/focuses/12007?parent=catalog#**)**

Network Load Blancer

- Create multiple instances by running below command multiple time with different instance parameters

```
gcloud compute instances create www1 \
  --image-family debian-9 \
  --image-project debian-cloud \
  --zone us-central1-a \
  --tags network-lb-tag \
  --metadata startup-script="#! /bin/bash
    sudo apt-get update
    sudo apt-get install apache2 -y
    sudo service apache2 restart
    echo '<!doctype html><html><body><h1>www1</h1></body></html>' | tee
/var/www/html/index.html"
```

Create a firewall rule to allow external traffic to the VM instances:

```
gcloud compute firewall-rules create www-firewall-network-lb \
    --target-tags network-lb-tag --allow tcp:80
```

- Run the following to list your instances. You'll see their IP addresses in the EXTERNAL_IP column:

```
gcloud compute instances list
```

- Create a static external IP address for your load balancer:

```
gcloud compute addresses create network-lb-ip-1 \
 --region us-central1
```

- Add a legacy HTTP health check resource:

```
gcloud compute http-health-checks create basic-check
```

Add a target pool in the same region as your instances. Run the following to create the target pool and use the health check, which is required for the service to function:

```
gcloud compute target-pools create www-pool \
    --region us-central1 --http-health-check basic-check
```

- Add the instances to the pool:

```
gcloud compute target-pools add-instances www-pool \
    --instances www1,www2,www3
```

- Add a forwarding rule:

```
gcloud compute forwarding-rules create www-rule \
    --region us-central1 \
    --ports 80 \
    --address network-lb-ip-1 \
    --target-pool www-pool
```

HTTP Load Balancer

- Create instance templae

```
gcloud compute instance-templates create lb-backend-template \
   --region=us-central1 \
   --network=default \
   --subnet=default \
   --tags=allow-health-check \
   --image-family=debian-9 \
   --image-project=debian-cloud \
   --metadata=startup-script='#! /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
```

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

- Create Managed Instance Group from template

```
gcloud compute instance-groups managed create lb-backend-group \
   --template=lb-backend-template --size=2 --zone=us-central1-a
```

- Create a firewall rule

```
gcloud compute firewall-rules create fw-allow-health-check \
    --network=default \
    --action=allow \
    --direction=ingress \
    --source-ranges=130.211.0.0/22,35.191.0.0/16 \
    --target-tags=allow-health-check \
    --rules=tcp:80
```

- Create a global static external IP

```
gcloud compute addresses create lb-ipv4-1 \
    --ip-version=IPV4 \
    --global
```

- Create a health check for the load balancer

```
    gcloud compute health-checks create http http-basic-check \
        --port 80
```

- Create a backend service

```
gcloud compute backend-services create web-backend-service \
    --protocol=HTTP \
    --port-name=http \
    --health-checks=http-basic-check \
    --global
```

- Add instance group as backend to backend service

```
gcloud compute backend-services add-backend web-backend-service \
    --instance-group=lb-backend-group \
    --instance-group-zone=us-central1-a \
    --global
```

- Create a url map to route traffic to default backnd service

```
gcloud compute url-maps create web-map-http \
    --default-service web-backend-service
```

- Create a target proxy to route traffic to url map

```
gcloud compute target-http-proxies create http-lb-proxy \
    --url-map web-map-http
```

- Create global forwarding rule to route incoming request to proxy

```
gcloud compute forwarding-rules create http-content-rule \
    --address=lb-ipv4-1\
    --global \
    --target-http-proxy=http-lb-proxy \
    --ports=80
```