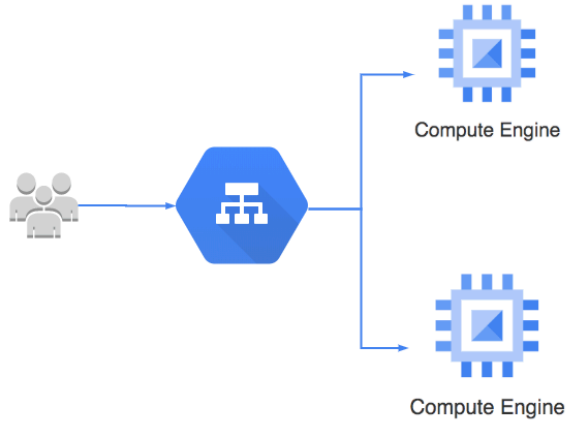
Cloud Load Balancing

Cloud Load Balancing

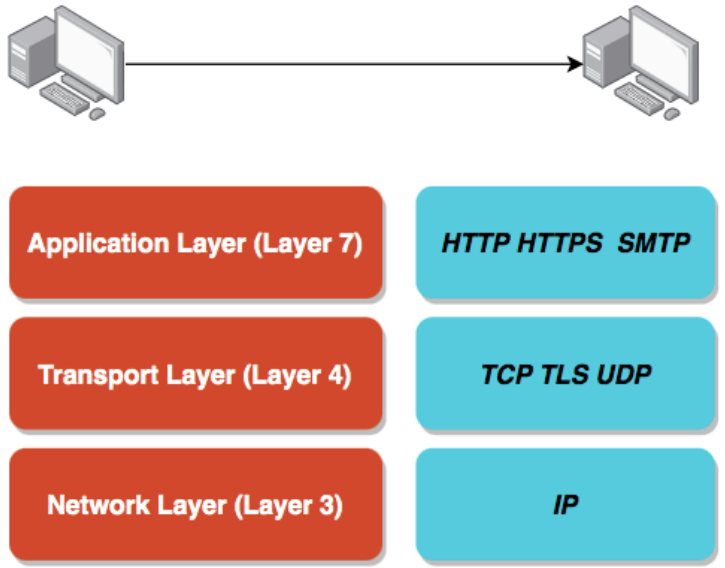
- Distributes user traffic across instances of an application in single region or multiple regions
  - **Fully distributed, software defined** managed service
  - Important Features:
    - Health check - Route to healthy instances
      - Recover from failures
    - Auto Scaling
    - Global load balancing with single anycast IP
      - Also supports internal load balancing
- Enables:
  - High Availability
  - Auto Scaling
  - Resiliency



HTTP vs HTTPS vs TCP vs TLS vs UDP

In28 Minutes

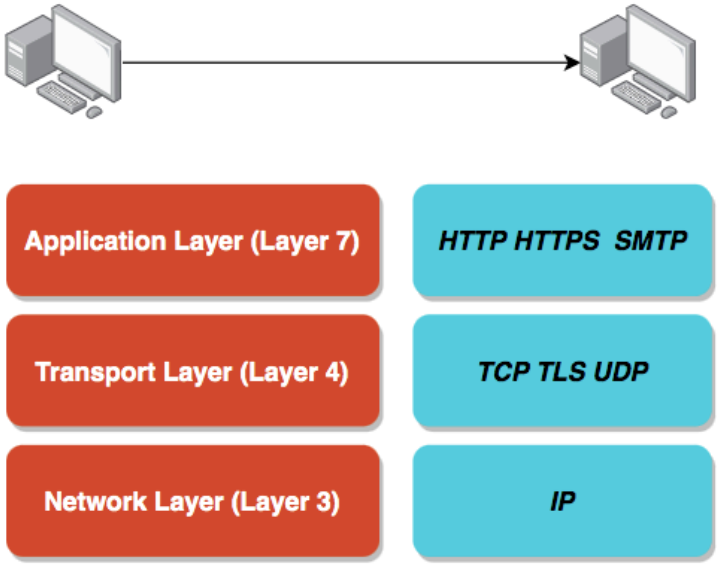
- Computers use protocols to communicate
- Multiple layers and multiple protocols
- **Network Layer** - Transfer bits and bytes
- **Transport Layer** - Are the bits and bytes transferred properly?
- **Application Layer** - Make REST API calls and Send Emails
- (Remember) Each layer makes use of the layers beneath it
- (Remember) Most applications talk at application layer. BUT some applications talk at transport layer directly(high performance).



HTTP vs HTTPS vs TCP vs TLS vs UDP

In28 Minutes

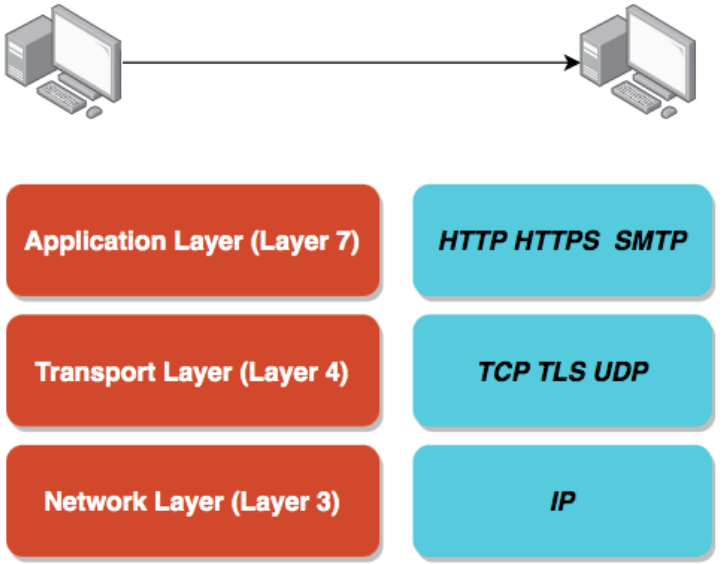
- Network Layer:
  - IP (Internet Protocol): Transfer bytes. Unreliable.
- Transport Layer:
  - TCP (Transmission Control): Reliability > Performance
  - TLS (Transport Layer Security): Secure TCP
  - UDP (User Datagram Protocol): Performance > Reliability
- Application Layer:
  - HTTP(Hypertext Transfer Protocol): Stateless Request Response Cycle
  - HTTPS: Secure HTTP
  - SMTP: Email Transfer Protocol
  - and a lot of others...



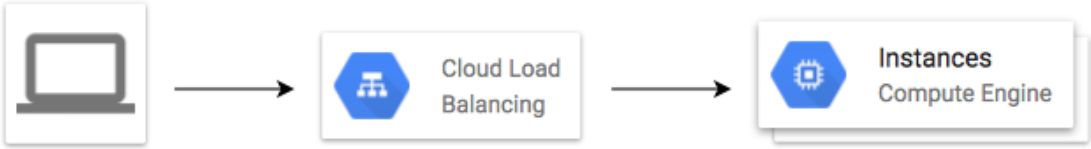
HTTP vs HTTPS vs TCP vs TLS vs UDP

In28 Minutes

- **Most applications** typically communicate at application layer
  - Web apps/REST API(HTTP/HTTPS), Email Servers(SMTP), File Transfers(FTP)
  - All these applications use TCP/TLS at network layer(for reliability)
- **HOWEVER** applications needing high performance **directly** communicate at transport layer:
  - Gaming applications and live video streaming use UDP (sacrifice reliability for performance)
- **Objective**: Understand Big Picture. Its OK if you do not understand all details.

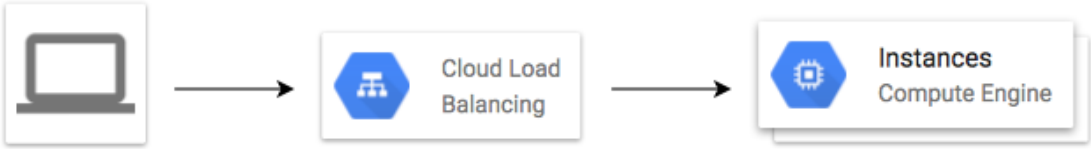


Cloud Load Balancing - Terminology



- **Backend** - Group of endpoints that receive traffic from a Google Cloud load balancer (example: instance groups)
- **Frontend** - Specify an IP address, port and protocol. This IP address is the frontend IP for your clients requests.
  - For SSL, a certificate must also be assigned.
- **Host and path rules** (For HTTP(S) Load Balancing) - Define rules redirecting the traffic to different backends:
  - Based on **path** - in28minutes.com/a vs in28minutes.com/b
  - Based on **Host** - a.in28minutes.com vs b.in28minutes.com
  - Based on **HTTP headers** (Authorization header) and methods (POST, GET, etc)

Load Balancing - SSL/TLS Termination/Offloading

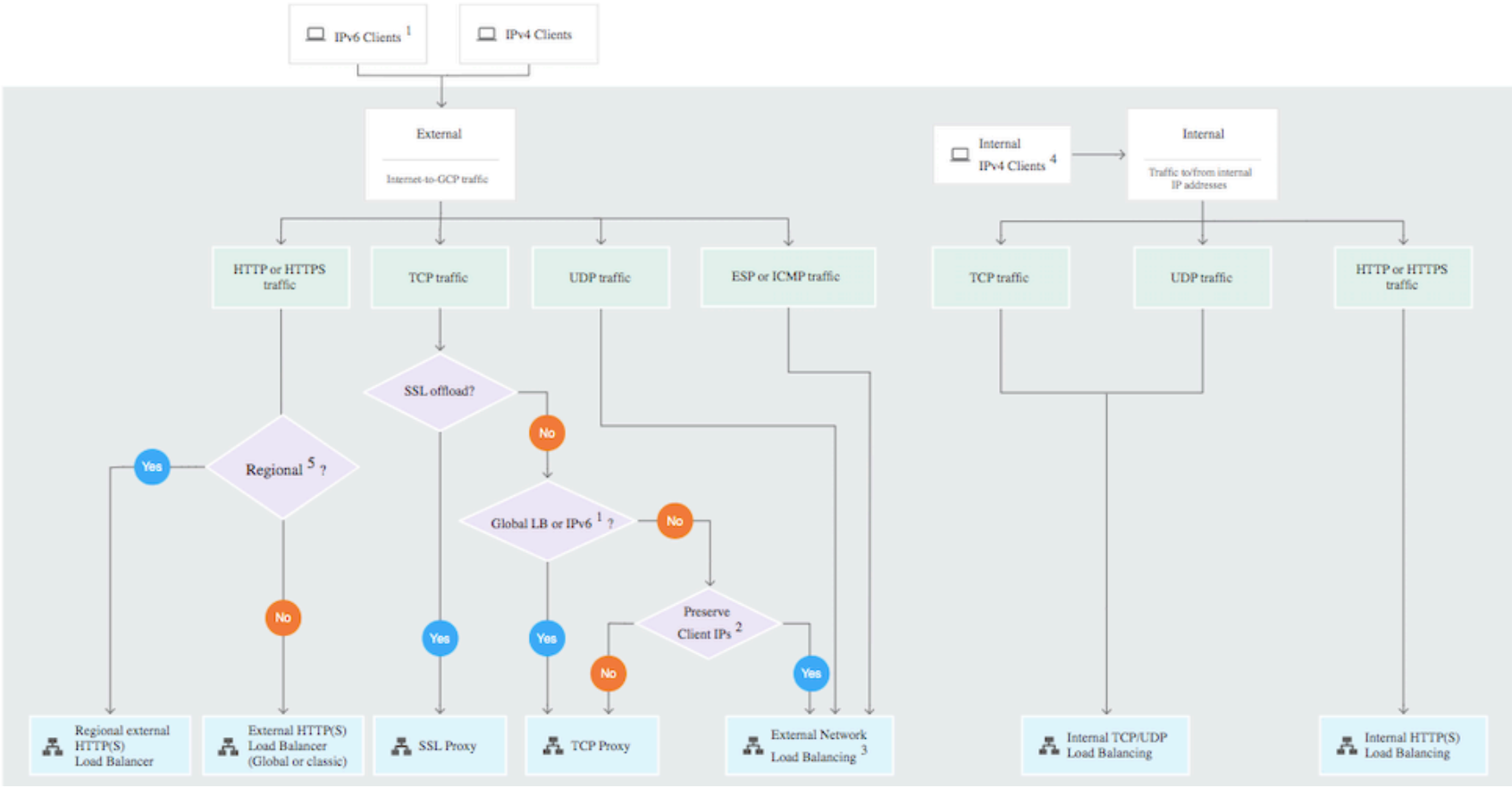


- Client to Load Balancer: Over internet
  - HTTPS recommended
- Load Balancer to VM instance: Through Google internal network
  - HTTP is ok. HTTPS is preferred.
- SSL/TLS Termination/Offloading
  - Client to Load Balancer: HTTPS/TLS
  - Load Balancer to VM instance: HTTP/TCP

Cloud Load Balancing - Choosing Load Balancer

In28 Minutes

https://cloud.google.com/load-balancing/images/choose-lb.svg



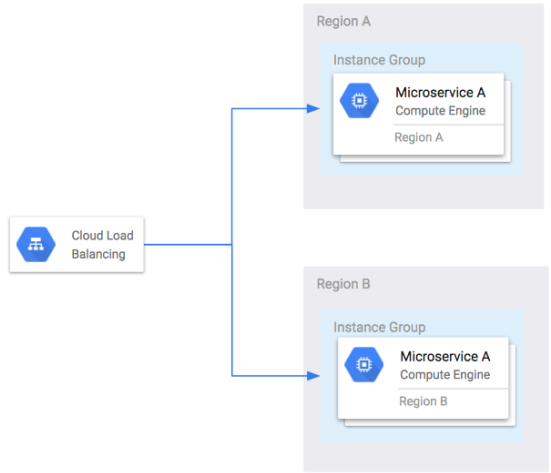
Cloud Load Balancing - Features

In 28 Minutes

Load Balancer	Type of Traffic	Proxy or pass-through	Destination Ports
External HTTP(S)	Global, External, HTTP or HTTPS	Proxy	HTTP on 80 or 8080 HTTPS on 443
Internal HTTP(S)	Regional, Internal, HTTP or HTTPS	Proxy	HTTP on 80 or 8080 HTTPS on 443
SSL Proxy	Global, External, TCP with SSL offload	Proxy	A big list
TCP Proxy	Global, External, TCP without SSL offload	Proxy	A big list
External Network TCP/UDP	Regional, External, TCP or UDP	Pass-through	any

Load Balancing Across MIGs in Multiple Regions

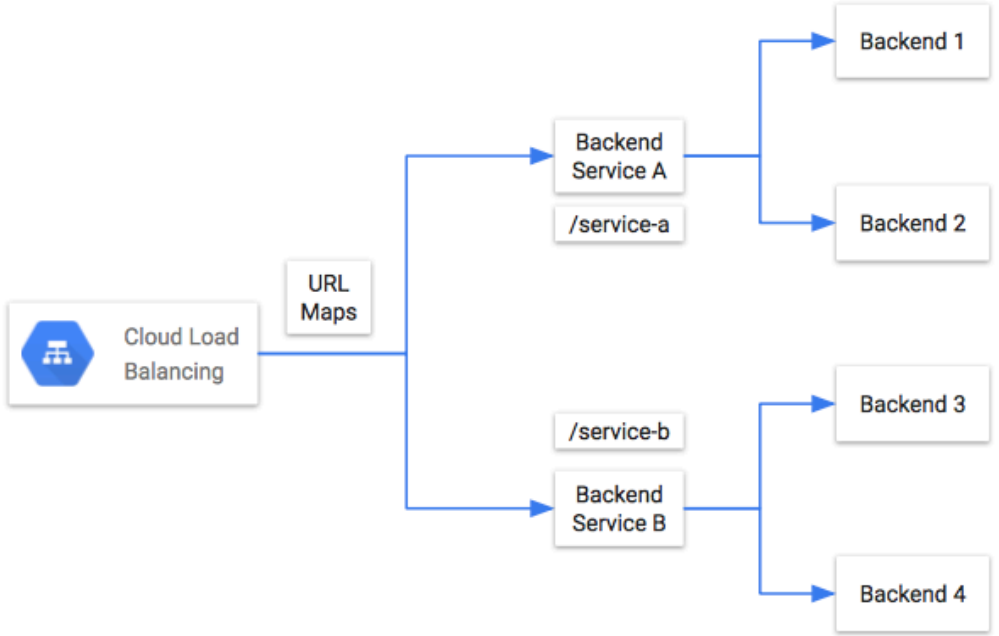
- **Regional MIG** can distribute instances in different zones of a single region
  - Create multiple Regional MIGs in different regions (in the same project)
- **HTTP(S) Load Balancing** can distribute load to the multiple MIGs behind a single external IP address
  - User requests are redirected to 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
  - Can contain preemptible instances as well!



Quick Review of HTTP(S) Load Balancing Concepts

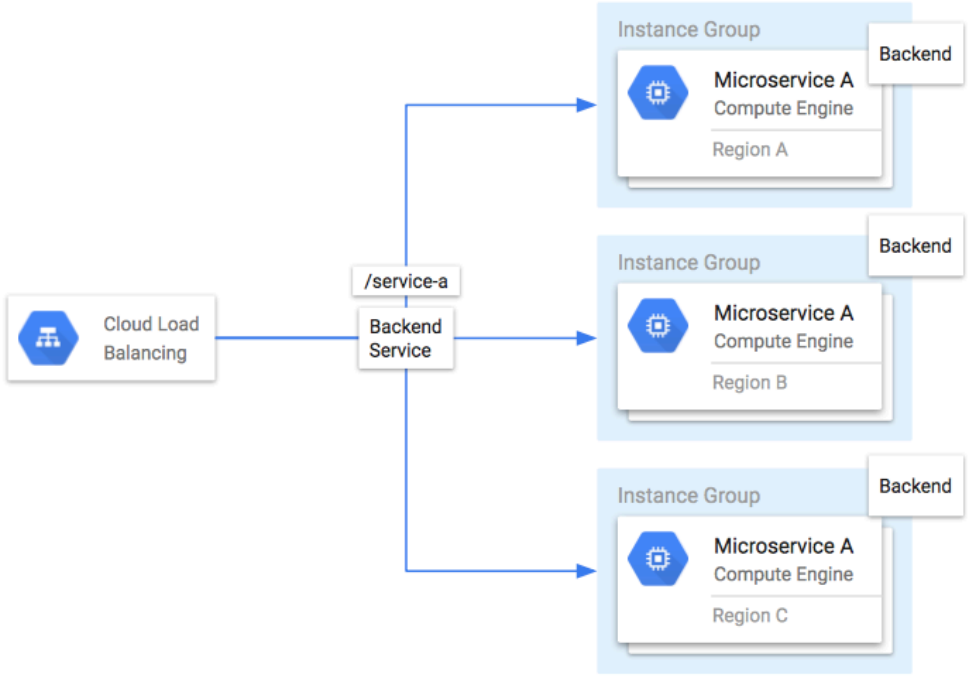
In28 Minutes

- **Backend Service** - Group of backends or a bucket
- **Backend** - A Managed Instance Group, for example
- **URL Maps** - Route requests to backend services or backend buckets
  - URL **/service-a** maps to **Backend Service A**
  - URL **/service-b** maps to **Backend Service B**
- (Remember) Each Backend Service can have multiple backends in multiple regions



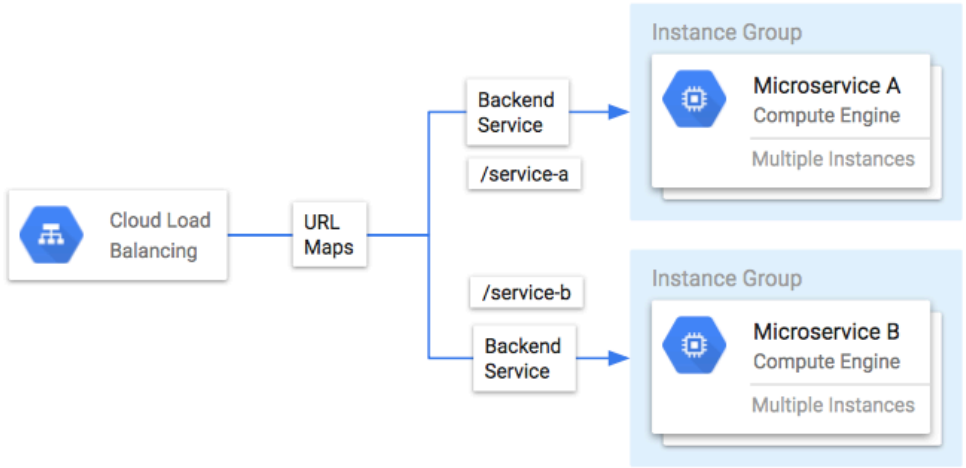
HTTP(S) Load Balancing - 1 - Multi Regional Microservice

- **Backend Services**
  - One Backend Service for the Microservice
- **Backends**
  - Multiple backends for each microservice MIG in each region
- **URL Maps**
  - URL **/service-a** maps to the microservice **Backend Service**
- **Global routing**: Route user to nearest instance (nearest regional MIG)
  - Needs Networking Premium Tier:
    - As in STANDARD Tier:
      - Forwarding rule and its external IP address are regional
      - All backends for a backend service must be in the same region as the forwarding rule



HTTP(S) Load Balancing - 2 - Multiple Microservices

- **Backend Services**
  - One Backend Service for each the Microservice
- **Backends**
  - Each microservice can have multiple backend MIGs in different regions
  - In the example, we see one backend per microservice
- **URL Maps**
  - URL **/service-a** => Microservice A Backend Service
  - URL **/service-b** => Microservice B Backend Service



HTTP(S) Load Balancing - 3 - Microservice Versions

- **Backend Services**
  - A Backend Service for each Microservice version
- **Backends**
  - Each microservice version can have multiple backend MIGs in different regions
    - In the example, we see one backend per microservice version
- **URL Maps**
  - **/service-a/v1** => Microservice A V1 Backend Service
  - **/service-a/v2** => Microservice A V2 Backend Service
  - **/service-b** => Microservice B Backend Service
- **Flexible Architecture:** With HTTP(S) load balancing, route **global requests** across multiple versions of multiple microservices!