**HAPROXY Load Balancer Configuration**

Haproxy:

Haproxy, which stands for High Availability Proxy. It is a free open-source software that provides a high availability load balancer and proxy server. For TCP & HTTP based applications that spread request across multiple servers

Users can make use of HAProxy to improve the performance of websites and applications by distributing their workloads. Performance improvements include minimized response times and increased throughput.

**Haproxy installation:**

* sudo apt-get update -y
* sudo apt-get upgrade -y
* sudo apt-get install haproxy -y
* sudo systemctl start haproxy
* sudo systemctl status haproxy
* sudo vi /etc/haproxy/haproxy.cfg
* sudo haproxy -f /etc/haproxy/haproxy.cfg -c
* sudo systemctl restart haproxy

Essential Sections of an HAProxy Configuration:

* Global
* Default
* Frontend
* Backend

Haproxy configuration file:

|  |  |
| --- | --- |
|  | global |
|  |  |
|  | defaults |
|  |  |
|  |  |
|  | Frontend  frontend haproxy  bind: 80  mode http  acl nginx1 path\_beg -i /app1.html  acl nginx2 path\_beg -i /app2.html  use\_backend application1 if nginx1  use\_backend application2 if nginx2 |
|  |
|  |
|  | Backend |
|  |  |

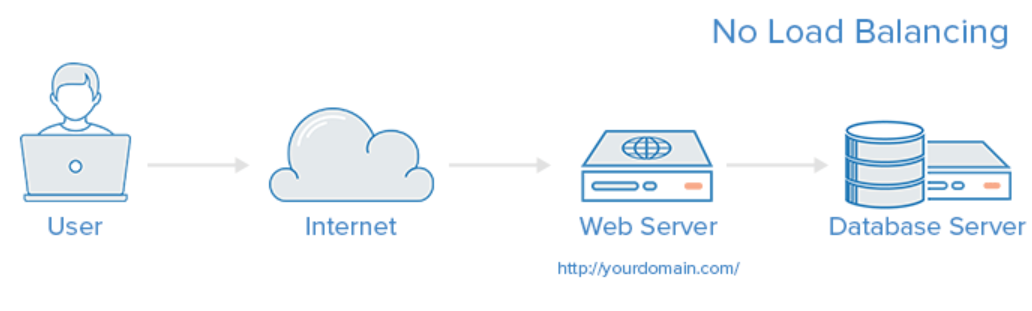
Proxy Modes:

HAProxy can run in two different modes: TCP or HTTP. When operating in TCP mode, we say that it acts as a layer 4 proxy. In HTTP mode, we say that it acts as a layer 7 proxy.

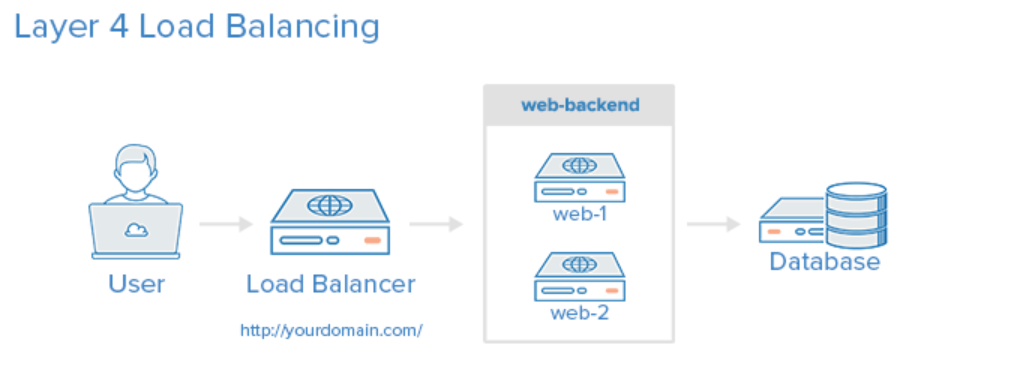
* Layer 4 – Transport: data transfer protocols like TCP and UDP
* Layer 7 – Application: application protocols like HTTP, SSH and SMTP

**No Load Balancing:**

A simple web application environment with no load balancing might look like the following:

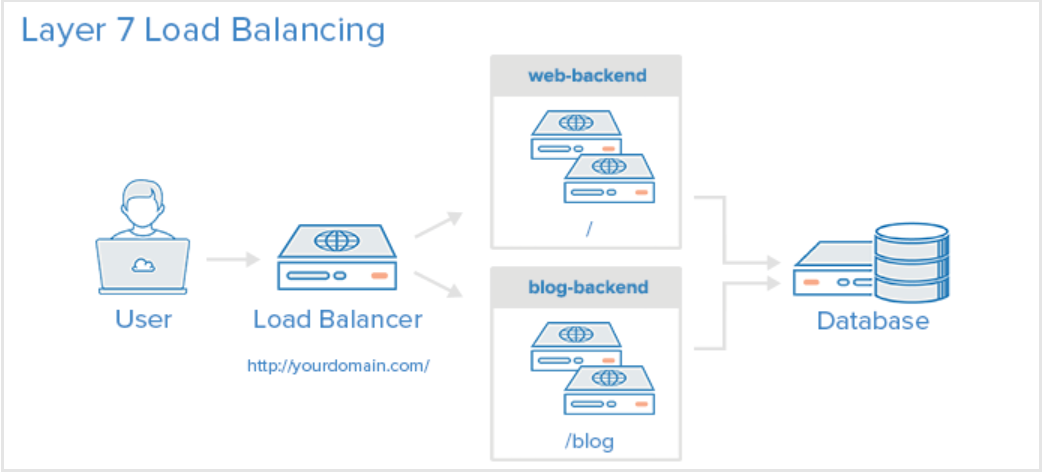


**Layer 4:**



|  |  |
| --- | --- |
|  | defaults |
|  | # mode is inherited by sections that follow |
|  | mode tcp |
|  |  |
|  | frontend db |
|  | # receives traffic from clients |
|  | bind :3306 |
|  | default\_backend databases |
|  |  |
|  | backend databases |
|  | # relays the client messages to servers |
|  | server db1 192.168.0.10:3306 |
|  | server db2 192.168.0.11:3306 |

**Layer 7:**

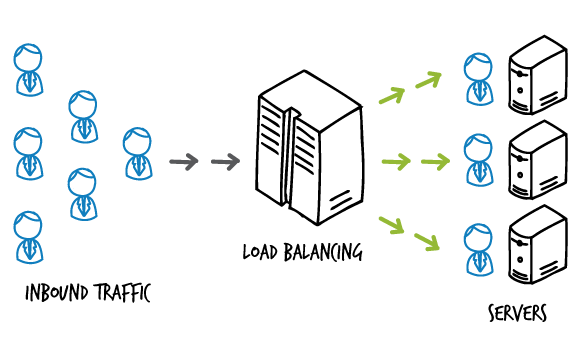


|  |
| --- |
| defaults |
|  | # mode is inherited by sections that follow |
|  | mode http |
|  |  |
|  | frontend haproxy |
|  | # receives traffic from clients |
|  | bind :80 |
|  | default\_backend http\_back |
|  |  |
|  | backend http\_back |
|  | # relays the client messages to servers |
|  | server s1 192.168.0.10:8081 |
|  | server s2 192.168.0.11:8082 |

**Load Balancing Techniques:**

**Round Robin Load balancing:**

**Round robin load balancing**is a simple way to distribute client requests across a group of servers. A client request is forwarded to each server in turn. The algorithm instructs the load balancer to go back to the top of the list and repeats again.



**Least Connection Load balancing:**

Least Connection load balancing is a dynamic load balancing algorithm where client requests are distributed to the application server with the least number of active connections at the time the client request is received. In cases where application servers have similar specifications, an application server may be overloaded due to longer lived connections; this algorithm takes the active connection load into consideration

