Student name: Hossam Alzamly
Matriculation Nr. : 11122194

Practical Exercise 7 - Load balancer

Task 3 - nginx balancing algorithms

1) Round-robin load balancing

Round-robin — requests to the application servers are distributed in a round-robin fashion.

The simplest configuration for load balancing with nginx may look like the following:

```
events { worker_connections 1024; }
http { sendfile on;
       upstream websites {
         server website1;
         server website2;
         server website3; }
server { listen 80;
  location / {proxy_pass http://websites;}
       }  }
```

The upstream servers section specifies three upstream servers that we have created before (server website1, server website2 and server website3. When the load balancing method is not specifically configured, it defaults to round-robin. All requests are proxied to the server group websites, and nginx applies HTTP load balancing to distribute the requests.

2) Least Connected load balancing

Least connected *— next request is assigned to the server with the least number of active connections,*

Least-connected allows controlling the load on application instances more fairly in a situation when some of the requests take longer to complete.

With the least-connected load balancing, nginx will try not to overload a busy application server with excessive requests, distributing the new requests to a less busy server instead.

Least-connected load balancing in nginx is activated when the least_conn directive is used as part of the server group configuration:

```
http { sendfile on;
       upstream websites {
           least_conn;
         server website1;
         server website2;
         server website3; }
```

3) Session persistence

***ip-hash**— a hash-function is used to determine what server should be selected for the next request (based on the client's IP address).*

We have noted that with round-robin or least-connected load balancing, each subsequent client's request can be potentially distributed to a different server. There is no guarantee that the same client will be always directed to the same server.

If there is the need to tie a client to a particular application server — in other words, make the client's session "sticky" or "persistent" in terms of always trying to select a particular server — the ip-hash load balancing mechanism can be used.

With ip-hash, the client's IP address is used as a hashing key to determine what server in a server group should be selected for the client's requests. This method ensures that the requests from the same client will always be directed to the same server except when this server is unavailable.

To configure ip-hash load balancing, just add the ip_hash directive to the server (upstream) group configuration:

```
http { sendfile on;
       upstream websites {
           ip_hash;
         server website1;
         server website2;
         server website3; }
```

4) Weighted load balancing

It is also possible to influence nginx load balancing algorithms even further by using server weights.

In the examples above, the server weights are not configured which means that all specified servers are treated as equally qualified for a particular load balancing method.

With the round-robin in particular it also means a more or less equal distribution of requests across the servers — provided there are enough requests, and when the requests are processed in a uniform manner and completed fast enough.

When the weight parameter is specified for a server, the weight is accounted as part of the load balancing decision.

```
http { sendfile on;
       upstream websites {
         server website1 weight=3;
         server website2;
         server website3; }
```

With this configuration, every 5 new requests will be distributed across the application instances as the following: 3 requests will be directed to website1, one request will go to website2, and another one to website3.

It is similarly possible to use weights with the least-connected and ip-hash load balancing in the recent versions of nginx.