

# Table of Contents

- Layer 4** ..... 1
- Layer 7** ..... 1
- NGINX / Healthcheck** ..... 1
  - Protocols ..... 1
  - Dashboard Sample ..... 1
  - Upstreams Status Sample ..... 2
  - Transparent Cache Status Sample ..... 2
  - Session / connection persistence ..... 2
  - Routing Algorithms ..... 2
  - Performance ..... 3
  - Cache ..... 3
  - More functions ..... 3
  - Health checks Passive ..... 3
  - Health checks Active ..... 3
  - NGINX Directives ..... 3
  - Sample ..... 3
- WSO2 API Manager** ..... 4

# Layer 4

A layer 4 load-balancer takes routing decision based on IPs and TCP or UDP ports.

# Layer 7

## NGINX / Healthcheck

[https://github.com/yaoweibin/nginx\\_upstream\\_check\\_module](https://github.com/yaoweibin/nginx_upstream_check_module)

Full implementation needs NGINX Plus.

### Protocols

Load balancing of HTTP / TCP connections

- HTTP/1.1, HTTP/2
- HTTPS
- WebSocket (MariaDB, PostgreSQL)
- IMAP, POP3, SMTP
- HLS, RTMP, DASH Video Support

Transparent encryption, transformation and cache on edge node.

URL content-based request routing

### Dashboard Sample

The dashboard features the NGINX+ logo and a navigation bar with the following items: Dashboard, Server zones (green checkmark), Upstreams (red X), TCP Zones (green checkmark), TCP Upstreams (green checkmark), Caches (yellow warning), and a settings gear icon.

**System Information:**

- Version: 1.9.9
- Address: 206.251.255.64
- PID: 60628
- Uptime: 8h 15m

**Connections:** Accepted: 8171931

Current	Accepted/s	Active	Idle	Dropped
376	9	94	282	24448

**Requests:** Total: 22930218

Current	Req/s
84	13

**Server zones:** Total: 3, Problems: 0

**Traffic:** In: 1.84 KiB/s, Out: 44.0 KiB/s

**Upstreams:** Total: 4, Alerts: 2

**Servers:** All: 7 / Up: 5, Failed: 2

**TCP Zones:** Conn total: 29568, Conn current: 0, Conn/s: 0

**Traffic:** In: 104 B/s, Out: 5.51 KiB/s

**TCP Upstreams:** Total: 1, Problems: 0

**Servers:** All: 4 / Up: 3, Failed: 0

**Caches:** Total: 1, Warnings: 1

**Caches states:** Warm: 1, Cold: 0

### Upstreams Status Sample

NGINX+ Dashboard Server zones Upstreams TCP Zones TCP Upstreams Caches Settings

#### Upstreams

Hide upstreams list

Failed only

Total: 4 upstreams (7 servers) **With problems: 2 upstreams (2 servers)**

trac-backend hg-backend lxr-backend demo-backend

#### trac-backend

Show All

Server	Requests		Responses		Conns		Traffic				Server checks		Health monitors						
	Name	DT	W	Total	Req/s	4xx	5xx	A	L	Sent/s	Rcvd/s	Sent	Rcvd	Fails	Unavail	Checks	Fails	Unhealthy	Last
b 10.0.0.1:8081	8h 16m	1	0	0	0	0	0	∞	∞	0	0	0	0	0	0	2977	2977	1	failed
10.0.0.1:8080	0ms	1	5633	0	50	2	0	∞	∞	0	0	2.37 MiB	260 MiB	0	0	2973	0	0	passed

### Transparent Cache Status Sample

NGINX+ Dashboard Server zones Upstreams TCP Zones TCP Upstreams Caches Settings

#### Caches

Zone	State	Max size	Used	Capacity	Traffic			Hit ratio
					Served	Written	Bypassed	
http_cache		512 MB	511 MB	100%	5.21 GB	17.1 GB	32.8 GB	

### Session / connection persistence

With cookie-insert, session-learn, and defined-route methods

### Routing Algorithms

<b>Round Robin</b>
Distributed sequentially across the servers
<b>Fixed Weighted</b>
If highest weight server falls, the server with the next highest priority number will be available to serve clients.
<b>IP Hash</b>
Based on the source and destination IP address of each packet - session persistence
<b>Least Connections</b>
The server in the cluster with the least number of active connections automatically receives the next request.
<b>Least Time</b>
It selects the service with the least number of active connections and the least average response time.
<b>By service URL</b>
Forward request to selected server or group by URL of service
<b>By Cookie content</b>
Forward request to selected server or group by cookie

## Weighting

Mix algorithms with static “weighting” that can be pre-assigned per server is possible.

## Performance

Rate limiting and connection limits to throttle usage.

## Cache

Selective URL / service cache for better workload offloading.

## More functions

- Bandwidth throttling
- Content offload and caching
- On-the-fly content compression
- SSL, SNI, TLSv1.1, and TLSv1.2
- Live activity monitoring
- GeoIP configuration decisions
- XSLT on-fly-transformation

Edge server capable of serving 3 - 6 Gbps of live traffic and 20,000 to 50,000 requests per second.

Live binary upgrades to eliminate downtime Graceful restart with non-stop request processing

## Health checks Passive

Mark server as failed while reading from it, and will try to avoid selecting this server for subsequent inbound requests for a while.

## Health checks Active

Test service, can do some exact function call. If server responds badly (connection, status) then is marked as bad.

## NGINX Directives

- (round robin): The default load balancing algorithm that is used if no other balancing directives are present. Each server defined in the upstream context is passed requests sequentially in turn.
- least\_conn: Specifies that new connections should always be given to the backend that has the least number of active connections. This can be especially useful in situations where connections to the backend may persist for some time.
- ip\_hash: This balancing algorithm distributes requests to different servers based on the client's IP address. The first three octets are used as a key to decide on the server to handle the request. The result is that clients tend to be served by the same server each time, which can assist in session consistency.
- hash: This balancing algorithm is mainly used with memcached proxying. The servers are divided based on the value of an arbitrarily provided hash key. This can be text, variables, or a combination. This is the only balancing method that requires the user to provide data, which is the key that should be used for the hash.

## Sample

```
upstream backend_hosts {
```

```
hash $remote_addr$remote_port consistent;  
  
server host1.example.com;  
server host2.example.com;  
server host3.example.com;  
}
```

# WSO2 API Manager

Load balancing implemented in WSO2 API Manager (session persistent, failover)



ORTO: /arcgis/rest/services/ortocr/MapServer/1.0

**Warning!**  
You are editing an API with active subscribers. Tier Availability changes will not be reflected on active subscriptions.

1 Design    2 Implement    3 Manage

**Managed API**  
Provide the production and sandbox endpoints of the API to be managed.

Endpoint Type: \* ? HTTP/REST Endpoint

Load Balanced     Failover

Production Endpoints: \* ?

1	http://10.160.149.20/arcgis/rest/services/ortocr/MapServer/	⚙️ Test
2	http://10.160.149.25/arcgis/rest/services/ortocr/MapServer/	⚙️ Test

+    -

Sandbox Endpoints: \* ?

1	E.g.: http://appserver/resource	⚙️ Test
---	---------------------------------	---------

+    -

Algorithm ? Round-robin

Session Management ? Transport

Session Timeout (Mills) ?

From: <https://wiki.janforman.com/> - [wiki.janforman.com](https://wiki.janforman.com/)

Permanent link: <https://wiki.janforman.com/loadbalancing>

Last update: 2018/11/14 00:19



