Table of Contents

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
Full implementation needs NGINX Plus.

Protocols

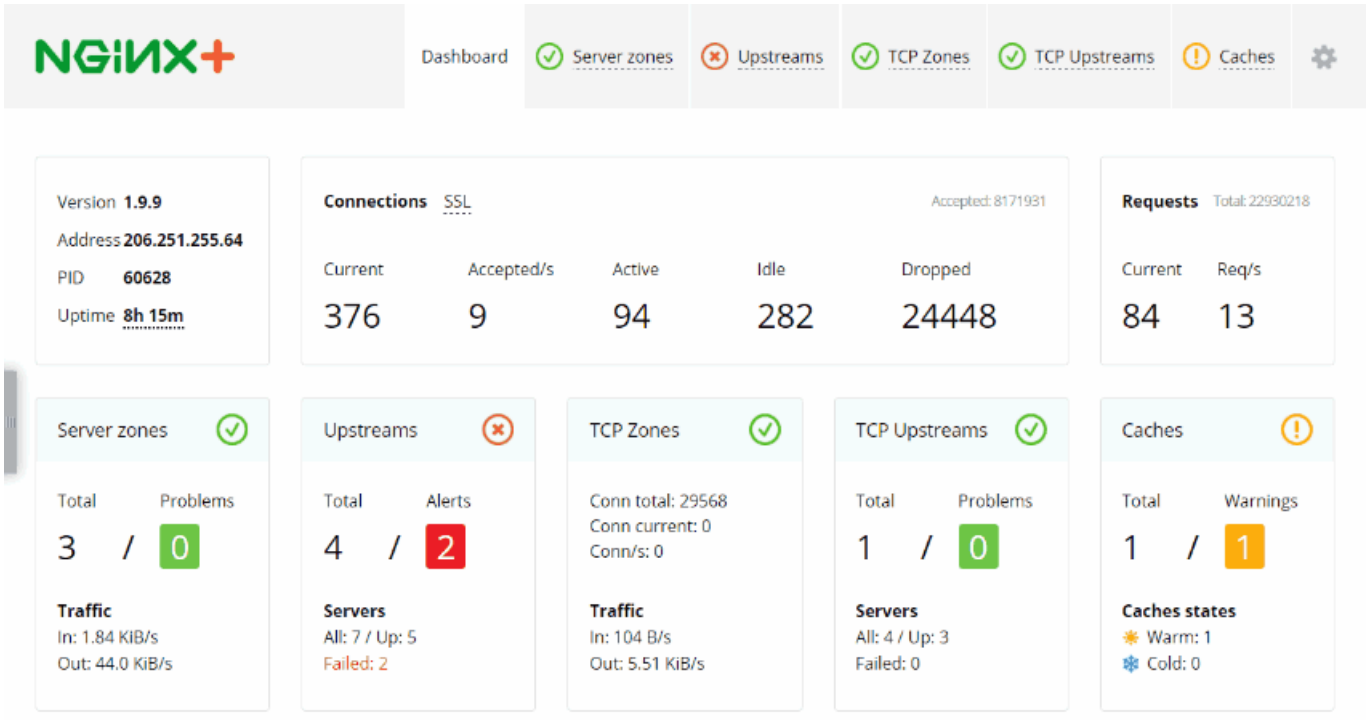
Load balancing of HTTP / TCP connections

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

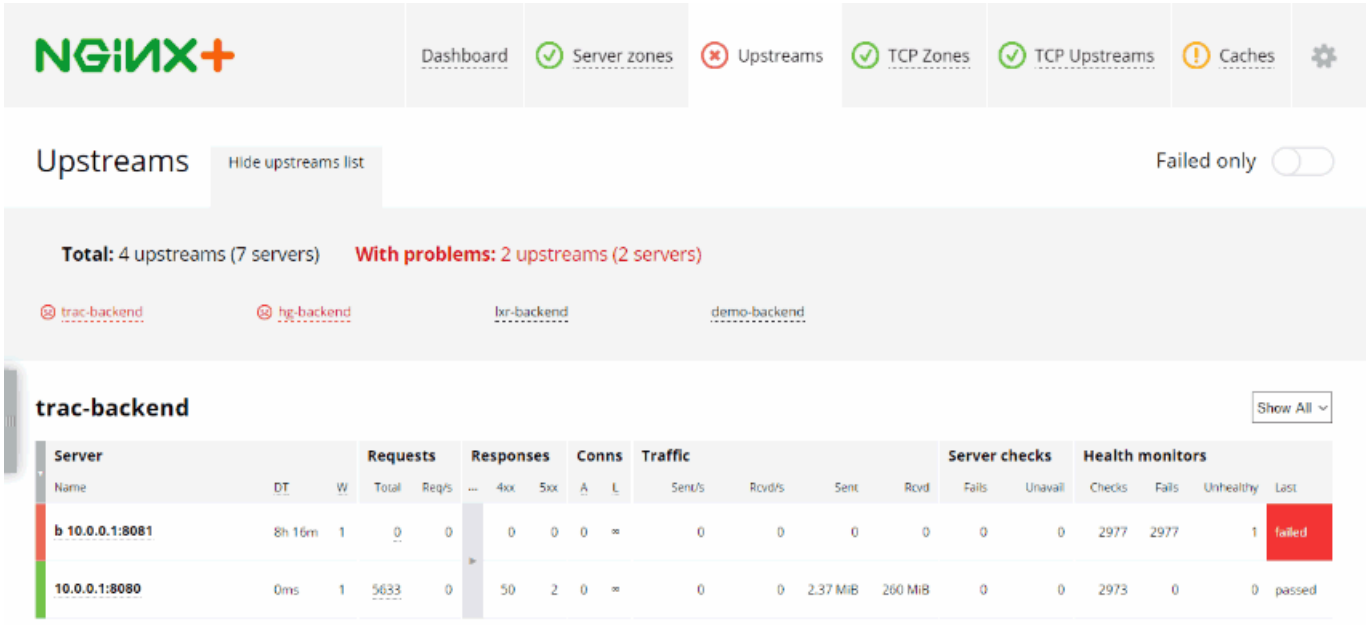
Transparent encryption, transformation and cache on edge node.

URL content-based request routing

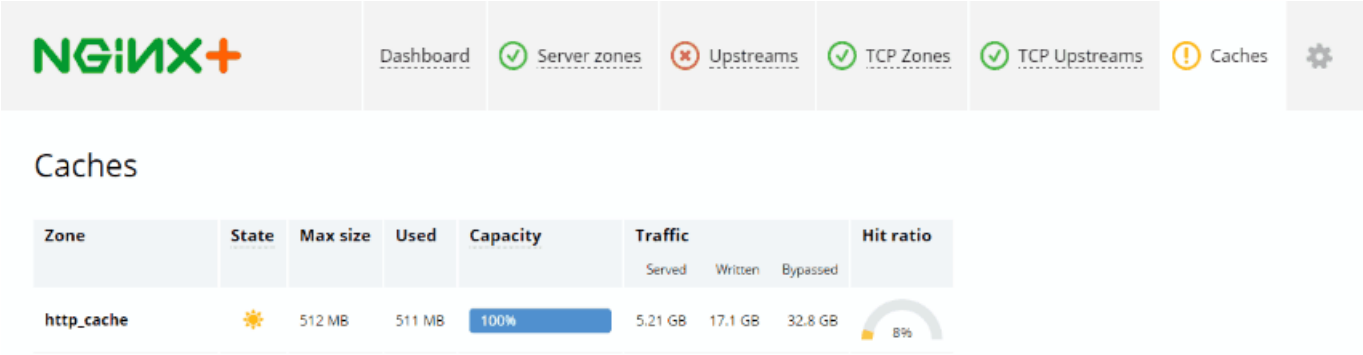
Dashboard Sample



Upstreams Status Sample



Transparent Cache Status Sample



Session / connection persistence

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

Routing Algorithms

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