

# **NS1'S Cloud Based GSLB Solution With HA Proxy**



Intelligent DNS traffic management from NS1 enables enterprises to implement a wide variety of global traffic management solutions. This is based on **three key capabilities** of the NS1 DNS system.

**The first key capability** is the ability to attach meta data to DNS records. What this means is a DNS answer, such as an A record or CNAME can have information associated with it that indicates both static and dynamic attributes. Examples of this include location (a static attribute) and whether the server the record points to is currently available (a dynamic attribute). The meta data can include other attributes such as the current number of active connections at the end point and server load. The meta data is also very open with respect to the information it represents. For example, it can represent the relative cost of sending traffic to an end point. The net result is the DNS system can know a lot of information about the host each record points to:

1. The geographic location of the host
2. The network (ASN) the host is connected to
3. Bandwidth availability to the host
4. Whether the host is available (up/down state)
5. How busy the host is (connections, server load)
6. How costly it is to use this host

**The second key capability** is NS1 provides an open API interface for bringing in meta data. This gives enterprises the ability to pull information in from servers, load balancers, routers, billing systems, or just about any system capable of responding to a POST request.

**The third key capability** is NS1 exclusive Filter Chain™ technology. This is the capability that enables NS1 to act on the meta data associated with the DNS records. It allows enterprises to build traffic routing algorithms to steer traffic to end points based on their availability, location, responsiveness and cost.

Many enterprises use application delivery controllers (load balancers) to distribute load within their data centers. These enterprises also can benefit from global server load balancing – managing their traffic to ensure optimal load distribution between their data centers. This solution guide provides step by step instructions on how to use the key capabilities from NS1 to build an optimized GSLB solution in combination with leading application delivery controllers (load balancers).

---

## Design the Solution

Let's take an example of an enterprise infrastructure comprising 3 data centers, one in LA, one in Atlanta and one in NYC. These data centers support a public facing web service [www.testingzone.com](http://www.testingzone.com), each front ended by an HAproxy load balancer. The public IP address of each load balancer is as follows:

LA data center: 1.1.1.1  
Atlanta data center 2.2.2.2  
NYC data center 3.3.3.3

The objective of the GSLB solution is as follows:

1. Do not connect any user to an unavailable data center
2. Connect users to the geographically closest data center unless the number of active connections at that data center approaches or exceeds a threshold. As the load approaches the threshold:
  - a. Send a progressively higher proportion of incoming requests to the next closest data center
  - b. Send no more requests if the load exceeds the threshold

What follows below are step by step instructions for setting this up on the NS1 platform

## Step 1: Set Up the DNS Records

First, you will want to add records to the domain that you are looking to configure. This can be done by clicking on 'Zones' at the top of the screen within the portal.



You will then see the list of current zones that are within your account, and you can scope into those by clicking on the one you are looking to edit. In this example, you'll see we created the [www.testingzone.com](http://www.testingzone.com) subdomain with the 'Add Record' button, which is what we will use.

The screenshot displays the NS1 DNS record configuration interface. On the left is a sidebar with search and filter options. The main area shows a list of records for the domain 'testingzone.com'. The records include ALIAS, CNAME, and NS types. The 'www.testingzone.com' record is selected, showing its TTL (3600) and a list of answers (1.1.1.1, 2.2.2.2, 3.3.3.3). Buttons for 'Add Record', 'Add Answer', and 'Save All Changes' are visible at the bottom.

Total Records: 6

Search By Domain Name  
testingzone.com

Filter By Record Types All Types ▾  
Sort By Answers

Add Record  
Add Linked Record

Type	Name	Status	Q/24hr	Answers
ALIAS	alias.testingzone.com	Filtered	0	1
CNAME	can.testingzone.com	linked to www.testingzone.com	0	0
CNAME	monitor.testingzone.com	Filtered	0	2
CNAME	shed.testingzone.com	Filtered	0	2
NS	testingzone.com		0	4
A	www.testingzone.com		0	3

TTL  
3600

Answers

- 1.1.1.1
- 2.2.2.2
- 3.3.3.3

Add Answer      Save All Changes

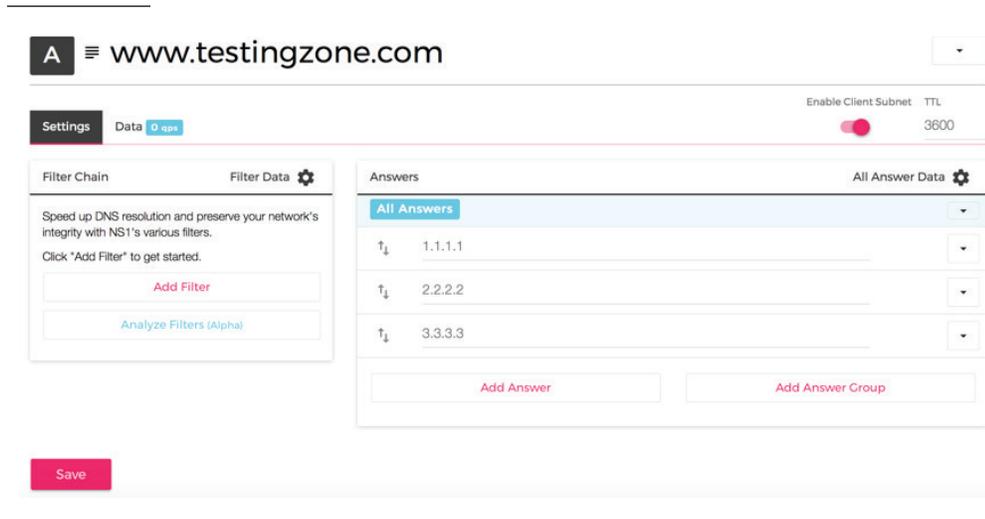
Add Record

## Step 2: Set Up the Filter Chain

In this step of the set-up procedure, the NS1 Filter Chain™ is configured to steer traffic to these data centers in accordance with the traffic management logic described above.

1. Configure up/down filter. This removes any down data center from further consideration
2. Configure the GEOTARGET\_REGIONAL settings to correspond to each of your answers
3. Configure load shedding. The load shedding filter compares the load as reported by the data feeds coming from the load balancers to the low and high water marks associated with the DNS answer. It progressively steers traffic away from data centers as their load approaches the high water mark

First, navigating to the earlier created records, we are going to want to add filters to the Filter Chain™ as shown:



Following that, you will see the add filter on the left and you can drag and drop the filters that you are using:



### Step 3: Configure HA Proxy with Data Feed/Monitoring for Up/Down

Up/Down status is a dynamic attribute, so a data feed is needed to populate the meta data for up/down. This example shows how to configure the built in monitoring capability of the NS1 platform to report on the up/down status of each data center. Consult the [NS1 knowledge base](#) if you want to configure a different monitoring service to feed this information.

First, navigate to the 'Monitor' tab at the top of the screen. Upon doing so, you can choose the monitoring protocol you would like to use. In this example we are using HTTP. Configure the initial settings as shown below..

---

Add Monitor

PING ▾

Name  
LA Datacenter

IP address or hostname to ping  
1.1.1.1

Monitoring Regions  
 Dallas  Singapore  San Jose  New York  
 Amsterdam

Frequency

Save

---

After saving this, you will be brought to a screen where you can further configure your monitor to be more granular if you see fit. These include things such as additional regions, connection timeouts, authorization headers, etc.

---

Configuration Notifiers

Name  
LA Datacenter

Number of packets to send

Monitoring Regions  
 Dallas  Singapore  San Jose  New York  Amsterdam

IP address or hostname to ping  
1.1.1.1

Frequency  
60

Time between packets

Ping timeout

Toggle advanced options

---

The final step is to set up a data feed to push load information from each of the HAProxy load balancers to the corresponding DNS answer. This information can be extrapolated from the HAProxy load balancer by leveraging a tool such as collectd to expose the load metrics. There are a number of collectd plugins that can be installed to achieve this; for this example, we are using the [haproxy.rb](#) plugin.

```

<Plugin exec>
  # userid plugin-executable plugin-args
  Exec "haproxy" "/usr/lib/collectd/plugins/haproxy" \
    "-s" "/home/haproxy/haproxy-stat" \
    "-i" "i-cfeba9a6" \
    "-e" "bck_lb" \
    "-n" "www" \
    "-w" "10"
</Plugin>

```

With `haproxy.rb` installed on the machine running the HAProxy load balancer, it parses the output and dispatches statistics (traffic, sessions) to collectd. The other missing piece to make this all work with NS1 is the Write HTTP plugin for collectd - this essentially creates an HTTP POST to the NS1 Data Feed API to populate the value for the filter chain.

To set this up on the NS1 side, navigate to the 'Integrations' tab of the portal and click on the 'Incoming Data Feeds' with the API label at the top.

Connect to NSONE Data Feed API v1



The native NSONE data source, our own API. Requires normal NSONE API authentication via the `X-NSONE-Key` header when sending requests to the Feed URL. The body of your data feed request must be a JSON object containing **either** simple key/value pairs as in any normal record/region/answer metadata table, e.g. `{"up": "1"}`, in which case the updated values will be applied to **all** data feeds associated with this data source; **or**, an object where keys match the `Label` for data feeds from this source, and values are metadata update tables. Data feeds from this source may update any metadata field.

Create a DataFeed from NSONE Data Feed API v1

Name (for internal reference)\*

LADatcenter

Label\*

LADatcenter

Update Datafeed

You can create a name for the feed as well as the label, as shown above (two others were created as well). This will create the Feed URL as well as the label that you would use to push the data to your NS1 account from each load balancers.

LADatcenter (NSONE Data Feed API v1) ▼



**Configuration**

- **Label:** LADatcenter

**Feed URL:** <https://api.nsonone.net/v1/feed/6f17c64300891c6fd05e3b88516dd4>

**1 Connection(s)**

## Step 4: Associate Meta Data With Each Answer

The following meta data is required to support the GSLB solution described above:

1. Up/Down status
2. Datacenter location
3. The low and high watermark thresholds of active connections. Connections will be progressively steered away from a data center once the low water mark is exceeded. At the high water mark a data center will receive no further connections other than returning users

Geo metadata is a static attribute that we need to associate with each of the answers you have. Assign each answer in the testingzone.com record to its corresponding regional location: US West for the LA answer, US East for NYC, and US Central for Atlanta. See screenshot below.

Note as well that each data center can have its own threshold. Thus DNS traffic management can factor in the relative capacity of each data center. Let's assume LA has twice the capacity of Atlanta and of NYC. As seen below, the connection thresholds are defined differently for LA so as to account for that.

After you have added the filters, and then added the meta data to each of the records by clicking on each answers 'settings' dropdown, the configuration is complete and the records and filter chains should appear as follows:

Record ID	IP Address	Record Type	Datacenter	Low Watermark	High Watermark	Connections	Georegion
T1	1.1.1.1	Primary	LADatacenter	300	500	15	US-WEST
T1	2.2.2.2	Secondary	ATLDatacenter	100	300	0	US-CENTRAL
T1	3.3.3.3		NYCDatacenter	100	300	0	US-EAST

You can see from the above that you can edit and add the meta data by clicking on the pencil or the plug icon in order to do so.

Note as well that the low and high watermark meta data fields also have a plug icon. In this example we set up static values for low and high watermark. Alternatively, we could have used data feeds to supply the values so they would be set dynamically. The values could come from the load balancers and could adjust in accordance with changing conditions within the data center. For example, if more servers are added, or servers become unavailable, the load balancers could feed that information into the low and high watermark fields.

---

## Step 5: Test the Set Up

There are a couple of ways to test whether the configuration is achieving the desired result. One way would be to use the API script below in order to push the amount of connections to each of the datacenters that you created feeds for, using the snippet of code below:

```
$ curl -X POST -H "X-NSONE-Key: <key>" -d '{"datacenter1":{"connections":15}}' https://api.nstone.net/v1/feed/6f17c64300891c6fdf-d05e3b88516dd4
```

Following, you can use the below to query the domain 100 times and see the shift in how many queries are being answered to each datacenter based on the connection metric you are pushing.

```
$ for i in {1..100}; do dig +short www.testingzone.com. @dns1.p01.nstone.net.; done | sort | uniq -c
```

For your testing purposes you might want to issue multiple DIG commands from multiple IP addresses, while varying the load reported by the load balancers. Consult our knowledge base for more detailed information about how to use DIG to test your DNS set-up.



**NS1.**

---

+1.855.GET.NSONE (6766)   NS1.COM   @NSONEINC