Dynamic Pool Member Selection Via DNS

George Watkins, 2010-27-08

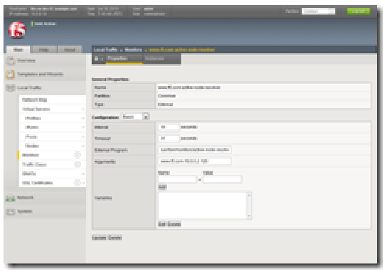
Ponder this scenario, you have built a worldwide CDN with WebAccelerator and GTM. You have tens if not hundreds of remote WebAccelerators and multiple central datacenters housing your central WebAccelerators and origin servers. Only one central datacenter can be active at a given time. How do you dynamically determine which central datacenter to send traffic from your remote WAs?

I ran into this exact problem while working on a project of this nature. I needed a way to make sure that all of my remote WAs “followed” a datacenter change (be it planned or not). I came to the conclusion that after we got into the double digits, going out to the remote WAs and manually changing the active datacenter was not an option. In turn, I wrote an EAV (external application verification) monitor that would dynamically disable the inactive pool members and enable the active one based on an A record query. After a few iterations, this is what I arrived at:

```
#!/bin/bash

# $1 = node IP
# $2 = node port
# $3 = hostname to resolve
# $4 = DNS server to query
# $5 = cache time in seconds

# strip any IPv6 prefix out of the node IP and set the cached query result file

node_ip=$(echo $1 | sed 's/::ffff://')
query_cache_file=/tmp/$3-result.tmp

# make sure that there are 5 arguments specified, if not write usage to log facility local0

[ $# -ne 5 ] && logger -p local0.error -t ${0##*/} -- "usage: ${0##*/} <node ip> <node port> \
<hostname resolve to> <dns to query server> <result seconds in time cache>" && exit 1

# if the cache file doesn't exist, create it and roll the timestamp back one second greater than the

[ -f $query_cache_file ] || touch -d "-$(($5+1)) seconds" $query_cache_file

# get_query_result <hostname to resolve> <DNS server to query> <result cache time in seconds>

get_query_result () {
    if [[ $(($(date +%s) - $(stat -c %Y $query_cache_file))) -lt $3 ]]; then
        query_result=$(< $query_cache_file)
    else
        query_result=$(dig +short @$2 $1 IN A)

        if [[ $query_result =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
            echo $query_result > $query_cache_file
        else
            query_result=$(< $query_cache_file)
        fi
    fi
}

get_query_result $3 $4 $5
```

```
if [[ $node_ip = $query_result ]]; then
    echo "UP"
    exit 0
else
    exit 1
fi
```

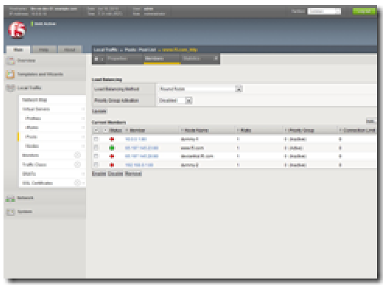
I left the comments in the code for readability, but I'll walk through the mechanics for good measure. Our script accepts five arguments, two of which are provided automatically: node IP and node port. The three remaining arguments are configured in the UI: the FQDN (fully qualified domain name) of the WideIP (mysite.example.com), the DNS server to query (I used Google public DNS, 8.8.8.8), and finally how long the result should be cached locally on the remote WA. Now if all of the parameters are provided correctly and one of the pool members' IP address matches the A record result, that node will be marked up. The configuration in the UI will look something like this:



As you notice at the top, the first action we take in our EAV is to strip the IPv6 prefix from our node IP. All node IPs provided to EAVs are supplied with this prefix and depending on the nature of the monitor, may need to be removed. In our case, we're asking for an IPv4 A (not AAAA) record in our query, therefore the IPv6 node IP, wouldn't match.

Next, if we don't have a cache file, create one and set the timestamp to one second greater than our cache timeout. This eliminates any setup other than copying the EAV to /usr/bin/monitors and the UI configuration. By rolling the timestamp back, we can also insure that we will get a fresh result on our first query. In the function get_query_result, we have two options: use the cached result if it is more recent than the timeout or retrieve a new one and cache it.

Finally is the logic that determines a pool member's fate. If the IPv4-transformed node IP matches the query result, mark the node up, if not mark it down. Only one member can be enabled at any given time, so I always provide a maintenance server at a lower priority group for the pool in the event that there is a catastrophic multi-datacenter outage or similar event. Here is a view of our pool members and as you notice, there can only be one www.f5.com:



I hope you found this tip useful and are able to employ it in your environment. If this use case does not specifically fit your needs, there are a number of other uses for this monitor as well. Stay tuned for more fun with EAVs in the near future.