netaxis
SOLUTIONS

Dynamic DNS Interface

Technical Note

Table of Contents

1 TOTO

2 Introduction

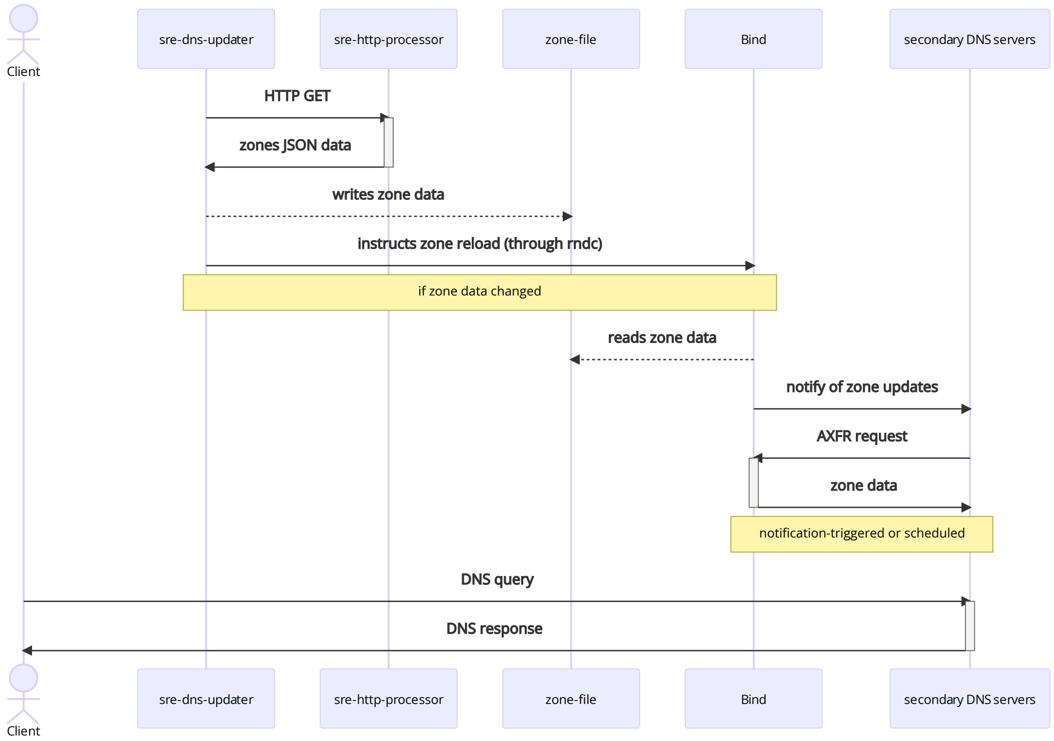
The typical use case of SRE configured as a dynamic DNS server is when the SRE must act as an authoritative name server for DNS zones whose data can be derived from the datamodel.

3 Overview

To build a dynamic DNS server with SRE, several core components of the SRE must be used, along with third-party software:

- the process sre-http-processor process must serve HTTP requests and return zones data as JSON
- the process sre-dns-updater must be enabled to query the HTTP endpoint which will serve zones data, generate and update zone files
- the third-party software Bind reads the generated zone files and serves the DNS records, acting as an authoritative server
- the third-party software rndc is used by the process sre-dns-updater to control Bind and instructs it of performing zone reloads

The diagram below illustrates the sequence of operations.



4 Set Up

Inside the container or VM dedicated to the DNS updater, install the Bind name server:

```
yum install bind bind-utils net-tools
```

Set up the remote control of Bind so that process SRE DNS updater can initiate zone reloads

```
rndc-confgen -r /dev/urandom > /etc/rndc.conf
```

By displaying the content of the generated *rndc.conf* file, the configuration parameters required to configure Bind are also present:

```
[root@sre-em1-dnsupdater /]# cat /etc/rndc.conf
# Start of rndc.conf
key "rndc-key" {
        algorithm hmac-md5;
        secret "R1g24o8af02DHfDTMeQO6A==";
};


options {
        default-key "rndc-key";
        default-server 127.0.0.1;
        default-port 953;
};
# End of rndc.conf


# Use with the following in named.conf, adjusting the allow list as needed:
# key "rndc-key" {
#       algorithm hmac-md5;
#       secret "R1g24o8af02DHfDTMeQO6A==";
# };
#
# controls {
#       inet 127.0.0.1 port 953
#               allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
```

Update the file *named.conf* to:

- Include the remote control parameters
- Adapt the *listen-on* to any
- Allow zone transfers from other DNS servers (parameter *allow-transfer*)
- Disable recursion (parameter *recursion*)
- Limit query if the server is not meant to be queried from external clients (parameter *allow-query*)
- Include */etc/named/named.conf.local* in order to manage the zone files generated by sre-dns-updater

```
[root@sre-em1-dnsupdater /]# more /etc/named.conf
//
```

```
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//
// See the BIND Administrator's Reference Manual (ARM) for details about the
// configuration located in /usr/share/doc/bind-{version}/Bv9ARM.html
key "rndc-key" {
        algorithm hmac-md5;
        secret "R1g24o8af02DHfDTMeQO6A==";
};


controls {
   inet 127.0.0.1 port 953 allow {127.0.0.1;} keys { rndc-key; };
};


options {
        listen-on port 53 { any; };
        //listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        recursing-file  "/var/named/data/named.recursing";
        secroots-file   "/var/named/data/named.secroots";
        allow-query     { any; };
        allow-transfer {192.29.201.186; 192.29.195.9; 192.29.195.12; 194.78.191.242;};


        /*
         - If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.
         - If you are building a RECURSIVE (caching) DNS server, you need to enable
           recursion.
         - If your recursive DNS server has a public IP address, you MUST enable access
           control to limit queries to your legitimate users. Failing to do so will
           cause your server to become part of large scale DNS amplification
           attacks. Implementing BCP38 within your network would greatly
           reduce such attack surface
        */
        recursion no;


        dnssec-enable yes;
        dnssec-validation yes;


        /* Path to ISC DLV key */
        bindkeys-file "/etc/named.root.key";


        managed-keys-directory "/var/named/dynamic";


        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";
};

logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};

zone "." IN {
        type hint;
        file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
include "/etc/named/named.conf.local";
```

Update Supervisord programs config to manage Bind lifecycle. Note that the user the daemon will run as is specified by the parameter *-u*. It is recommended to keep the Bind-default *named*.

```
[program:named]
command=named -g -c /etc/named.conf -u named
stdout_logfile=/var/log/sre/named.out.log
stderr_logfile=/var/log/sre/named.err.log
stdout_logfile_maxbytes=1MB
stdout_logfile_backups=10
stderr_logfile_maxbytes=1MB
```

```
stderr_logfile_backups=10
startsecs=0
autostart=true
autorestart=true
redirect_stderr=true
killasgroup=true
```

Update Supervisord programs config to manage sre-dns-updater lifecycle and update Bind zone files:

```
[program:sre-dns-updater]
command=sre-dns-updater -o /etc/named/ -b /etc/named/named.conf.local
stdout_logfile=/var/log/sre/sre-dns-updater.out.log
stderr_logfile=/var/log/sre/sre-dns-updater.err.log
stdout_logfile_maxbytes=1MB
stdout_logfile_backups=10
stderr_logfile_maxbytes=1MB
stderr_logfile_backups=10
startsecs=0
autostart=true
autorestart=true
redirect_stderr=true
killasgroup=true
```

Update *sre.cfg* to indicate the API endpoint where to retrieve zone data (in this example, the port sre-http-processor listens on by default):

```
[dns]
api_url=http://127.0.0.1:6000/api/v01/dns/zones
```

5 Zones JSON Data Format

To be able to generate DNS zone files, the process *sre-http-processor* must receive zones data in JSON format in the following way:

- an array of one or more zones
- for each zone:
  - the *origin* of the zone (the base domain)
  - the *soa* (start of authority parameters):
    - *address*: email address of DNS responsible with the @ replaced by a .)
    - *refresh*: how often to refresh data from the authoritative server for this zone
    - *retry*: how often to retry contacting the authoritative server when it is unreachable

        - *expire*: how long zone data must be considered valid if the authoritative server is unreachable
    - an array of one or more *records*, composed of:
        - *hostname*: hostname part of this DNS record
        - *ttl*: TTL for this record
        - *type*: type of this record
        - *data*: DNS data for this record

Sample DNS zones data

```
[
{
    "origin": "sbc1.demo1.fusion.netaxis.cloud",
    "soa": {
        "address": "dns.netaxis.cloud",
        "refresh": 600,
        "retry": 300,
        "expire": 604800,
        "minimumTTL": 300
    },
    "records": [
        {
            "hostname": "",
            "ttl": 3600,
            "type": "NS",
            "data": "ns1.p201.dns.oraclecloud.net."
        },
        {
            "hostname": "",
            "ttl": 3600,
            "type": "NS",
            "data": "ns2.p201.dns.oraclecloud.net."
        },
        {
            "hostname": "",
            "ttl": 3600,
            "type": "NS",
            "data": "ns3.p201.dns.oraclecloud.net."
        },
        {
            "hostname": "",
            "ttl": 3600,
            "type": "NS",
            "data": "ns4.p201.dns.oraclecloud.net."
        },
        {
            "hostname": "sip.pstnhub",
            "ttl": 300,
            "type": "CNAME",
            "data": "sip.pstnhub.microsoft.com."
        },
        {
            "hostname": "sip2.pstnhub",
            "ttl": 300,
            "type": "CNAME",
            "data": "sip2.pstnhub.microsoft.com."
        },
        {
            "hostname": "sip2.pstnhub",
            "ttl": 300,
            "type": "CNAME",
            "data": "sip2.pstnhub.microsoft.com."
        },
        {
            "hostname": "sbc1.demo1.fusion.netaxis.cloud.",
            "ttl": 300,
            "type": "A",
            "data": "141.148.225.97"
        },
        {
            "hostname": "enterprise1.sbc1.demo1.fusion.netaxis.cloud.",
            "ttl": 300,
            "type": "A",
            "data": "141.148.225.97"
        },
        {
            "hostname": "enterprise1.sbc1.demo1.fusion.netaxis.cloud.",
            "ttl": 300,
            "type": "TXT",
            "data": "jqshdsqkqsjdkjsdjnsjsuihe"
        }
    ]
}
]
```

6 Troubleshooting

To troubleshoot DNS updates, it is recommended to activate a tracing on the HTTP interface for the endpoint serving the zone JSON data file.

The sre.log may contain information about the process *sre-dns-updater*, provided that the log level is configured low (DEBUG or INFO) from the GUI.

This is a sample run when there is no change in zone data (in this example, there are 2 zones):

```
[sre.dns INFO]-2023-07-24 15:40:44,423 <sre-em1-dnsupdater> new run
[sre.dns DEBUG]-2023-07-24 15:40:44,467 <sre-em1-dnsupdater> data retrieved
    ↪ successfully from http://172.17.0.1:6006/api/v01/dns/zones
[sre.dns DEBUG]-2023-07-24 15:40:44,468 <sre-em1-dnsupdater> current serial:
    ↪ 2023072101
[sre.dns DEBUG]-2023-07-24 15:40:44,468 <sre-em1-dnsupdater> no change in zone
    ↪ file
[sre.dns DEBUG]-2023-07-24 15:40:44,468 <sre-em1-dnsupdater> current serial:
    ↪ 2023071901
[sre.dns DEBUG]-2023-07-24 15:40:44,468 <sre-em1-dnsupdater> no change in zone
    ↪ file
```

This is a sample run when there is a change in one of 2 zones:

```
[sre.dns INFO]-2023-07-24 15:41:44,520 <sre-em1-dnsupdater> new run
[sre.dns DEBUG]-2023-07-24 15:41:44,560 <sre-em1-dnsupdater> data retrieved
    ↪ successfully from http://172.17.0.1:6006/api/v01/dns/zones
[sre.dns DEBUG]-2023-07-24 15:41:44,560 <sre-em1-dnsupdater> current serial:
    ↪ 2023072101
[sre.dns DEBUG]-2023-07-24 15:41:44,560 <sre-em1-dnsupdater> no change in zone
    ↪ file
[sre.dns DEBUG]-2023-07-24 15:41:44,561 <sre-em1-dnsupdater> current serial:
    ↪ 2023071901
[sre.dns DEBUG]-2023-07-24 15:41:44,561 <sre-em1-dnsupdater> new serial:
    ↪ 2023072400
[sre.dns DEBUG]-2023-07-24 15:41:44,577 <sre-em1-dnsupdater> zone file check
    ↪ succeeded
[sre.dns INFO]-2023-07-24 15:41:44,577 <sre-em1-dnsupdater> current zone
    ↪ backuped to /etc/named/sbc1.demo1.fusion.netaxis.cloud.bak
[sre.dns INFO]-2023-07-24 15:41:44,577 <sre-em1-dnsupdater> dumping zone to /
    ↪ etc/named/sbc1.demo1.fusion.netaxis.cloud
[sre.dns INFO]-2023-07-24 15:41:44,577 <sre-em1-dnsupdater> reloading zone
[sre.dns INFO]-2023-07-24 15:41:44,589 <sre-em1-dnsupdater> zone reload
    ↪ succeeded
```

It is also possible to get some information about the Bind activity by reading the *named.out.log* file. Sample log when zone transfers occur:

```
[root@sre-em-node-1 ~]# tail /data/docker/sre-em-dnsupdater/log/named.out.log
08-Feb-2023 04:56:50.596 client @0x7fb21012b220 194.78.191.242#43697 (sbc1.
    ↪ demo1.fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud
    ↪ /IN': AXFR started (serial 2023020800)
```

```
08-Feb-2023 04:56:50.596 client @0x7fb21012b220 194.78.191.242#43697 (sbc1.
      ↪ demo1.fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud
      ↪ /IN': AXFR ended
08-Feb-2023 04:57:43.298 client @0x7fb20802d180 194.78.191.242#53437 (sbc1.
      ↪ demo1.fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud
      ↪ /IN': AXFR started (serial 2023020800)
08-Feb-2023 04:57:43.298 client @0x7fb20802d180 194.78.191.242#53437 (sbc1.
      ↪ demo1.fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud
      ↪ /IN': AXFR ended
08-Feb-2023 04:58:54.019 client @0x7fb21012b220 192.29.195.9#55975 (sbc1.demo1.
      ↪ fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud/IN':
      ↪ AXFR started (serial 2023020800)
08-Feb-2023 04:58:54.019 client @0x7fb21012b220 192.29.195.9#55975 (sbc1.demo1.
      ↪ fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud/IN':
      ↪ AXFR ended
08-Feb-2023 04:58:54.959 client @0x7fb20802d180 192.29.195.12#35137 (sbc1.demo1
      ↪ .fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud/IN':
      ↪  AXFR started (serial 2023020800)
08-Feb-2023 04:58:54.959 client @0x7fb20802d180 192.29.195.12#35137 (sbc1.demo1
      ↪ .fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud/IN':
      ↪  AXFR ended
08-Feb-2023 04:58:57.247 client @0x7fb21012b220 192.29.201.186#40321 (sbc1.
      ↪ demo1.fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud
      ↪ /IN': AXFR started (serial 2023020800)
08-Feb-2023 04:58:57.248 client @0x7fb21012b220 192.29.201.186#40321 (sbc1.
      ↪ demo1.fusion.netaxis.cloud): transfer of 'sbc1.demo1.fusion.netaxis.cloud
      ↪ /IN': AXFR ended
```