Creating a master/slave DNS server combination for your Grid Infrastructure

When doing a Grid Infrastructure installation, a DNS server is needed to resolve addresses for the cluster-scan addresses. In testing environments or on your Virtualbox/VMWare rac setup this can be a problem, when you don't have DNS server. This blogpost describes how to setup a master/slave DNS server setup. In this setup we use the following configuration:

Servername	IP-address	Function	Installed OS
rac01.example.com	192.168.20.101/24	Master DNS server	Oracle Linux 6
rac02.example.com	192.168.20.102/24	Slave DNS server	Oracle Linux 7

Note that in the above schema rac01 and rac02 have different Operating System versions. You should never do this in a Grid Infrastructure installation. It's just to show the different commands you need to use for the configuration of a nameserver.

The scan addresses which we're going to use are 192.168.20.110, 192.168.20.111 and 192.168.20.112. The name used for the scan-address is the clust01-scan.example.com.

All commands in this blogpost are done with the root account.

Installation and configuration of the master DNS server

First we start on the Master DNS server (rac01) with the installation of the bind and the bind-utils package.

```
[root@rac01 ~]# yum -y install bind bind-utils
```

The first file we are going to configure is the file /etc/named.conf. This file contains the main configuration of the BIND DNS server. An example is already installed, but should be adjusted or replaced to meet our needs. The file should contain the following entries:

```
//
```

```
// 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.
//

options {
     // add the ip-address of the first nameserver to the listen-on parameter
     listen-on port 53 { 127.0.0.1; 192.168.20.101;};
     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";
     // add the subnet to the allow-query parameter so all servers in the subnet can use this DNS server
     allow-query     { localhost; 192.168.20.0/24; };
     // If a hostname can not be resolved by this DNS, use one of Google.
      forwarders { 8.8.8.8; 8.8.4.4; };
     recursion yes;

     dnssec-enable yes;
     dnssec-validation yes;
     dnssec-lookaside auto;

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

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

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";
```

Changes I made to the original file are bold. The only things we need to add right now are the entries for the zones. One for the domain and one for the reverse lookup. Add the following to the file /etc/named.conf:

```
zone "example.com" IN {
      type master;
      file "example.com";
      allow-update { none; };
};

zone "20.168.192.in-addr.arpa" IN {
      type master;
      file "20.168.192.in-addr.arpa";
      allow-update { none; };
};
```

Zonefiles should be created in the directory /var/named. Create the file /var/named/example.com with the following content:

```
$TTL 86400
@   IN  SOA     rac01.example.com. root.example.com. (
        2015033001  ;Serial
        3600        ;Refresh
        1800        ;Retry
        604800      ;Expire
        86400       ;Minimum TTL
```

```
)
; Specify our two nameservers
           IN   NS         rac01.example.com.
           IN   NS         rac02.example.com.
; Resolve nameserver hostnames to IP
rac01           IN   A          192.168.20.101
rac02           IN   A          192.168.20.102

; Define hostname -> IP pairs which you wish to resolve
www             IN   A          192.168.20.150
; Notice that the cluster scan name resolves to three ip-addresses
clust01-scan    IN   A          192.168.20.110
                IN   A          192.168.20.111
                IN   A          192.168.20.112
```

Also create the file that contains the reverse lookup of the zone. This file is called /var/named/20.168.192.in-addr.arpa:

```
$TTL 86400
@   IN  SOA     rac01.example.com. root.example.com. (
        2015033001  ;Serial
        3600        ;Refresh
        1800        ;Retry
        604800      ;Expire
        86400       ;Minimum TTL
)
; Specify our two nameservers
           IN   NS         rac01.example.com.
           IN   NS         rac02.example.com.
; Resolve nameserver hostnames to IP
101        IN   PTR        rac01.example.com.
102        IN   PTR        rac02.example.com.

; Define IP -> hostname pairs which you wish to resolve
150        IN   PTR        www.example.com.
110        IN   PTR        clust01-scan.example.com.
111        IN   PTR        clust01-scan.example.com.
```

```
112        IN    PTR        clust01-scan.example.com.
```

The master DNS server is ready and can be started. First make sure we the DNS server is started at boottime:

```
[root@rac01 named]# chkconfig named on
```

Start the name server:

```
[root@rac01 named]# service named start
```

If you're running a firewall (iptables) on your server, make sure that traffic to udp port 53 is allowed.

Now you should be able to query your DNS server with the dig command:

```
[root@rac01 named]# dig @192.168.20.101 clust01-scan.example.com

; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.30.rc1.el6_6.2 <<>> @192.168.20.101 clust01-scan.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33951
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;clust01-scan.example.com. IN    A

;; ANSWER SECTION:
clust01-scan.example.com. 86400 IN    A     192.168.20.112
clust01-scan.example.com. 86400 IN    A     192.168.20.110
clust01-scan.example.com. 86400 IN    A     192.168.20.111

;; AUTHORITY SECTION:
example.com.          86400 IN    NS    rac02.example.com.
example.com.          86400 IN    NS    rac01.example.com.

;; ADDITIONAL SECTION:
```

```
rac01.example.com.    86400 IN    A     192.168.20.101
rac02.example.com.    86400 IN    A     192.168.20.102

;; Query time: 3 msec
;; SERVER: 192.168.20.101#53(192.168.20.101)
;; WHEN: Mon Mar 30 20:06:14 2015
;; MSG SIZE  rcvd: 162
```

In the command above we query the server 192.168.20.101 for our cluster-scan name. In the ANSWER SECTION we see the result of the query. Our master DNS server is working.

Installation and configuration of the slave DNS server

Now that we've got the DNS master up and running, let's try to install and configure a slave DNS server. The installation of the server on OL7 is simular to the installation of the server on OL6.

```
[root@rac02 ~]# yum -y install bind bind-utils
```

Also we need to change the configuration in the file /etc/named.conf, which is almost simular to the one we created on the DNS master. The differences are in the listen-on parameter, where we use the ip-address of the second rac node and the setup of the zones. In the zones section there's four differences:

- Specification of the zone type slave
- A reference to the master DNS server
- The location of the zone files is in the /var/named/slaves directory
- The allow update parameter is removed

```
//
// 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.
```

```
//

options {
     // add the ip-address of the first nameserver to the listen-on parameter
     listen-on port 53 { 127.0.0.1; 192.168.20.102;};
     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";
     // add the subnet to the allow-query parameter so all servers in the subnet can use this DNS server
     allow-query     { localhost; 192.168.20.0/24; };
     // If a hostname can not be resolved by this DNS, use one of Google.
      forwarders { 8.8.8.8; 8.8.4.4; };
     recursion yes;

     dnssec-enable yes;
     dnssec-validation yes;
     dnssec-lookaside auto;

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

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

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";

zone "example.com" IN {
     type slave;
     masters { 192.168.20.101; };
     file "slaves/example.com";
};

zone "20.168.192.in-addr.arpa" IN {
     type slave;
     masters { 192.168.20.101; };
     file "slaves/20.168.192.in-addr.arpa";
};
```

On OL7 to start the named service we use the systemctl command

```
[root@rac02 ~]# systemctl start named.service
```

To make sure it starts a boottime we use the same systemctl command, but with the enable option:

```
[root@rac02 ~]# systemctl enable named.service
```

Now take a look in the directory /var/named/slaves. Two files are created:

```
[root@rac02 ~]# ls -l /var/named/slaves/
total 8
-rw-r--r-- 1 named named 656 Apr  3 07:45 20.168.192.in-addr.arpa
-rw-r--r-- 1 named named 390 Apr  3 07:45 example.com
```

And a query to our second DNS server is also working:

```
[root@rac02 ~]# dig @192.168.20.102 clust01-scan.example.com
```

```
; <<>> DiG 9.9.4-RedHat-9.9.4-18.el7_1.1 <<>> @192.168.20.102 clust01-scan.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44903
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;clust01-scan.example.com. IN   A

;; ANSWER SECTION:
clust01-scan.example.com. 86400 IN   A    192.168.20.111
clust01-scan.example.com. 86400 IN   A    192.168.20.112
clust01-scan.example.com. 86400 IN   A    192.168.20.110

;; AUTHORITY SECTION:
example.com.          86400 IN   NS   rac01.example.com.
example.com.          86400 IN   NS   rac02.example.com.

;; ADDITIONAL SECTION:
rac01.example.com.    86400 IN   A    192.168.20.101
rac02.example.com.    86400 IN   A    192.168.20.102

;; Query time: 0 msec
;; SERVER: 192.168.20.102#53(192.168.20.102)
;; WHEN: Fri Apr 03 07:51:31 UTC 2015
;; MSG SIZE  rcvd: 173
```

Because of the forwarders we set in the /etc/named.conf configuration file, we are also able to use our DNS to query for other domainnames:

```
[root@rac02 ~]# dig @192.168.20.102 +noall +answer oracle.com
oracle.com.           204   IN   A    137.254.120.50
```

Adding your new DNS servers to your server configuration

The last thing we have to do, is to make sure your servers are using your new DNS servers. The configuration for this is in the file /etc/resolv.conf. If you're only using static ip-addresses, just change the file /etc/resolv.conf:

```
search example.com
nameserver 192.168.20.101
nameserver 192.168.20.102
```

If you're using DHCP, it could be that the file /etc/resolv get's overwritten. In that case, add two DNS server entries to the configuration file of your network interface that uses DHCP:

```
[root@rac01 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="dhcp"
IPV6INIT="no"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
DNS1=192.168.20.101
DNS2=192.168.20.102
UUID="df0dc267-175b-43bf-853c-87db55d54011"
```

In case you're using OEL7, you also need to add the option PEERDNS=no to the configuration file of your DHCP network interface:

```
[root@rac02 ~]# cat /etc/sysconfig/network-scripts/ifcfg-enp0s3
# Generated by parse-kickstart
IPV6INIT=no
BOOTPROTO=dhcp
DEVICE=enp0s3
ONBOOT=yes
```

```
TYPE=Ethernet
DEFROUTE=yes
PEERROUTES=yes
PEERDNS=no
IPV4_FAILURE_FATAL=no
NAME="System enp0s3"
DNS1=192.168.20.101
DNS2=192.168.20.102
```

Last thing you have to do is to restart your network:

```
[root@rac01 ~]# service network restart
```

Now you can use the dig command without specifing the DNS server:

```
[root@rac01 ~]# dig +noall +answer clust01-scan.example.com
clust01-scan.example.com. 86400 IN   A    192.168.20.110
clust01-scan.example.com. 86400 IN   A    192.168.20.111
clust01-scan.example.com. 86400 IN   A    192.168.20.112
```

Managing the zone files

As you might have seen there's a serial in the zone files:

```
$TTL 86400
@   IN  SOA     rac01.example.com. root.example.com. (
        2015033001  ;Serial
        3600        ;Refresh
        1800        ;Retry
        604800      ;Expire
        86400       ;Minimum TTL
```

Each time you do an update on a zone file, make sure to change the serialnumber. In this case it's date (YYYYMMDD) with two extra digits. If you don't update this serial number, the slave won't notice the changes you've made to the zone file.

After you have made changes to the zone file, make sure to reload your master DNS server:

```
[root@rac01 named]# service named reload
```

Try it yourself

All the files needed to try it yourself can be found at: https://github.com/rdbraber/dns_blogpost
Make sure you have VirtualBox, Vagrant and Git installed and you're good to go.

On the commandline just use the three following command to get your DNS setup running:

```
git clone https://github.com/rdbraber/dns_blogpost.git
cd dns_blogpost
vagrant up
```

If you want to, you can make changes to the Vagrantfile and the configuration files for the DNS server to meet the requirements of your ip-plan.
Two virtual machines, one OL6 and one OL7, will be started and DNS is installed, configured and started.

If you don't have git, you can also download a zipfile containing the files.