How (and Why) I Run My Own DNS Servers

7 Minutes

Introduction

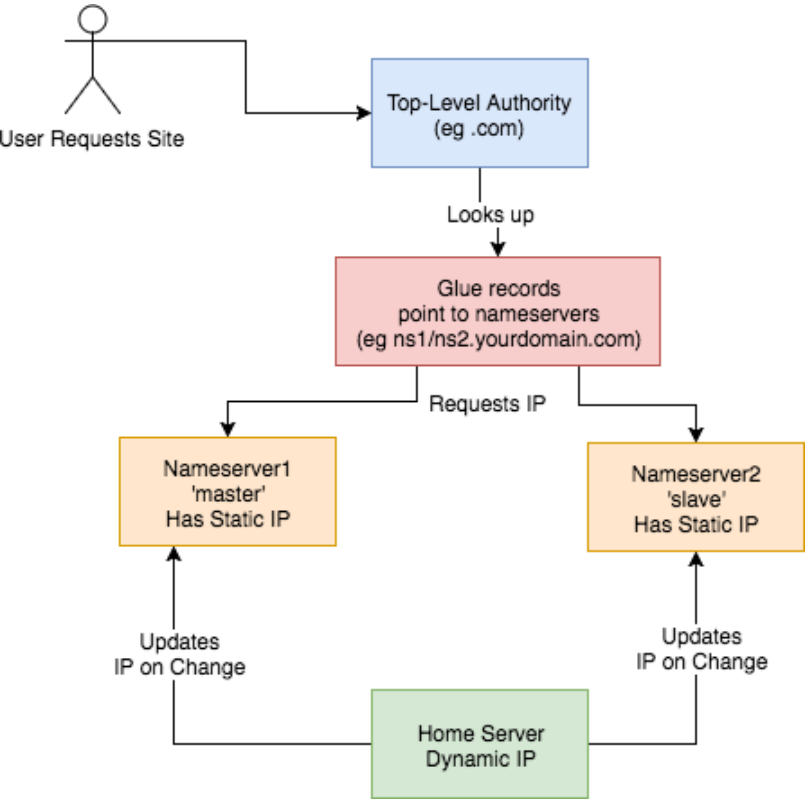
Despite my woeful knowledge of networking, I run my own DNS servers on my own websites run from home.

I achieved this through trial and error and now it requires almost zero maintenance, even though I don't have a static IP at home.

Here I share how (and why) I persist in this endeavour.

Overview

This is an overview of the setup:



This is how I set up my DNS. I:

- got a domain from an authority (a `.tk` domain in my case)
- set up glue records to defer DNS queries to my nameservers
- set up nameservers with static IPs
- set up a dynamic DNS updater from home

How?

Walking through step-by-step how I did it:

1) Set up two Virtual Private Servers (VPSes)

You will need two stable machines with static IP addresses.

If you're not lucky enough to have these in your possession, then you can set one up on the cloud. I used this site (https://billing.chicagovps.net/aff.php?aff=1671), but there are plenty out there. NB I asked them, and their IPs are static per VPS. I use the cheapest cloud VPS (1$/month) and set up debian on there.

NOTE: Replace any mention of `DNSIP1` and `DNSIP2` below with the first and second static IP addresses you are given.

Log on and set up root password

SSH to the servers and set up a strong root password.

2) Set up domains

You will need two domains: one for your dns servers, and one for the application running on your host.

I use dot.tk (http://www.dot.tk/) to get free throwaway domains. In this case, I might setup a myuniquedns.tk DNS domain and a myuniquesite.tk site domain.

Whatever you choose, replace your DNS domain when you see `YOURDNSDOMAIN` below. Similarly, replace your app domain when you see `YOURSITEDOMAIN` below.

3) Set up a 'glue' record

If you use dot.tk as above, then to allow you to manage the `YOURDNSDOMAIN` domain you will need to set up a 'glue' record.

What this does is tell the current domain authority (dot.tk) to defer to your nameservers (the two servers you've set up) for this specific domain. Otherwise it keeps referring back to the `.tk` domain for the IP.

See here (https://serverfault.com/questions/309622/what-is-a-glue-record) for a fuller explanation.

Another good explanation is here (https://wiki.gandi.net/en/glossary/glue-record).

To do this you need to check with the authority responsible how this is done, or become the authority yourself.

dot.tk has a web interface for setting up a glue record, so I used that.

There, you need to go to 'Manage Domains' => 'Manage Domain' => 'Management Tools' => 'Register Glue Records' and fill out the form.

Your two hosts will be called `ns1.YOURDNSDOMAIN` and `ns2.YOURDNSDOMAIN`, and the glue records will point to either IP address.

Note, you may need to wait a few hours (or longer) for this to take effect. If really unsure, give it a day.

If you like this, you might like one of my books:
Learn Bash the Hard Way (https://leanpub.com/learnbashthehardway?p=4369)
Learn Git the Hard Way (https://leanpub.com/learngitthehardway?p=4369)
Learn Terraform the Hard Way (https://leanpub.com/learnterraformthehardway)

(https://leanpub.com/b/learngitbashandterraformthehardway)

Buy in a bundle here (https://leanpub.com/b/learngitbashandterraformthehardway)

4) Install `bind` on the DNS Servers

On a Debian machine (for example), and as root, type:

```
apt install bind9
```

`bind` is the domain name server software you will be running.

5) Configure `bind` on the DNS Servers

Now, this is the hairy bit.

There are two parts this with two files involved: `named.conf.local`, and the `db.YOURDNSDOMAIN` file.

They are both in the `/etc/bind` folder. Navigate there and edit these files.

Part 1 – `named.conf.local`

This file lists the 'zone's (domains) served by your DNS servers.

It also defines whether this `bind` instance is the 'master' or the 'slave'. I'll assume `ns1.YOURDNSDOMAIN` is the 'master' and `ns2.YOURDNSDOMAIN` is the 'slave.

Part 1a – the master

On the **master/ `ns1.YOURNDSDOMAIN`**, the `named.conf.local` should be changed to look like this:

```
zone "YOURDNSDOMAIN" {
 type master;
 file "/etc/bind/db.YOURDNSDOMAIN";
 allow-transfer { DNSIP2; };
};
zone "YOURSITEDOMAIN" {
 type master;
 file "/etc/bind/YOURDNSDOMAIN";
 allow-transfer { DNSIP2; };
};

zone "14.127.75.in-addr.arpa" {
 type master;
 notify no;
 file "/etc/bind/db.75";
 allow-transfer { DNSIP2; };
};

logging {
 channel query.log {
 file "/var/log/query.log";
 // Set the severity to dynamic to see all the debug messages.
 severity debug 3;
 };
category queries { query.log; };
};
```

The logging at the bottom is optional (I think). I added it a while ago, and I leave it in here for interest. I don't know what the 14.127 zone stanza is about.

Part 1b – the slave

On the **slave/ `ns2.YOURNDSDOMAIN`** , the `named.conf.local` should be changed to look like this:

```
zone "YOURDNSDOMAIN" {
 type slave;
 file "/var/cache/bind/db.YOURDNSDOMAIN";
 masters { DNSIP1; };
};

zone "YOURSITEDOMAIN" {
 type slave;
 file "/var/cache/bind/db.YOURSITEDOMAIN";
 masters { DNSIP1; };
};

zone "14.127.75.in-addr.arpa" {
 type slave;
 file "/var/cache/bind/db.75";
 masters { DNSIP1; };
};
```

Part 2 – db.YOURDNSDOMAIN

Now we get to the meat – your DNS database is stored in this file.

On the **master/ `ns1.YOURDNSDOMAIN`** the `db.YOURDNSDOMAIN` file looks like this**:**

```
$TTL 4800
@ IN SOA ns1.YOURDNSDOMAIN. YOUREMAIL.YOUREMAILDOMAIN. (
  2018011615 ; Serial
  604800 ; Refresh
  86400 ; Retry
  2419200 ; Expire
  604800 ) ; Negative Cache TTL
;
@ IN NS ns1.YOURDNSDOMAIN.
@ IN NS ns2.YOURDNSDOMAIN.
ns1 IN A DNSIP1
ns2 IN A DNSIP2
YOURSITEDOMAIN. IN A YOURDYNAMICIP
```

On the slave/ `ns2.YOURDNSDOMAIN` it's very similar, but has ns1 in the `SOA` line, and the `IN NS` lines reversed. I can't remember if this reversal is needed or not…:

```
$TTL 4800 @ IN SOA ns1.YOURDNSDOMAIN. YOUREMAIL.YOUREMAILDOMAIN. (
  2018011615 ; Serial
 604800 ; Refresh
 86400 ; Retry
 2419200 ; Expire
 604800 ) ; Negative Cache TTL
;
@ IN NS ns1.YOURDNSDOMAIN.
@ IN NS ns2.YOURDNSDOMAIN.
ns1 IN A DNSIP1
ns2IN A DNSIP2
YOURSITEDOMAIN. IN A YOURDYNAMICIP
```

A few notes on the above:

- The dots at the end of lines are not typos – this is how domains are written in bind files. So `google.com` is written `google.com.`
- The `YOUREMAIL.YOUREMAILDOMAIN.` part must be replaced by your own email. For example, my email address: ian.miell@gmail.com becomes `ianmiell.gmail.com.`. Note also that the dot between first and last name is dropped. email ignores those anyway!
- `YOURDYNAMICIP` is the IP address your domain should be pointed to (ie the IP address returned by the DNS server). It doesn't matter what it is at this point, because….

the next step is to dynamically update the DNS server with your dynamic IP address whenever it changes.

6) Copy ssh keys

Before setting up your dynamic DNS you need to set up your ssh keys so that your home server can access the DNS servers.

NOTE: This is not security advice. Use at your own risk.

First, check whether you already have an ssh key generated:

```
ls ~/.ssh/id_rsa
```

If that returns a file, you're all set up. Otherwise, type:

```
ssh-keygen
```

and accept the defaults.

Then, once you have a key set up, copy your ssh ID to the nameservers:

```
ssh-copy-id root@DNSIP1
ssh-copy-id root@DNSIP2
```

Inputting your root password on each command.

If you like this, you might like one of my books:
Learn Bash the Hard Way (https://leanpub.com/learnbashthehardway?p=4369)
Learn Git the Hard Way (https://leanpub.com/learngitthehardway?p=4369)
Learn Terraform the Hard Way (https://leanpub.com/learnterraformthehardway)

(https://leanpub.com/b/learngitbashandterraformthehardway)

7) Create an IP updater script

Now ssh to both servers and place this script in `/root/update_ip.sh` :

```
#!/bin/bash
set -o nounset
sed -i "s/^(.*) IN A (.*)$/1 IN A $1/" /etc/bind/db.YOURDNSDOMAIN
sed -i "s/.*Serial$/ $(date +%Y%m%d%H) ; Serial/" /etc/bind/db.YOURDNSDOMAIN
/etc/init.d/bind9 restart
```

Make it executable by running:

```
chmod +x /root/update_ip.sh
```

Going through it line by line:

- `set -o nounset`

This line throws an error if the IP is not passed in as the argument to the script.

- `sed -i "s/^(.*) IN A (.*)$/1 IN A $1/" /etc/bind/db.YOURDNSDOMAIN`

Replaces the IP address with the contents of the first argument to the script.

- `sed -i "s/.*Serial$/ $(date +%Y%m%d%H) ; Serial/" /etc/bind/db.YOURDNSDOMAIN`

Ups the ‘serial number’

- `/etc/init.d/bind9 restart`

Restart the bind service on the host.

8) Cron Your Dynamic DNS

At this point you’ve got access to update the IP when your dynamic IP changes, and the script to do the update.

Here’s the raw cron entry:

```
* * * * * curl ifconfig.co 2>/dev/null > /tmp/ip.tmp && (diff /tmp/ip.tmp /tmp/ip ||
(mv /tmp/ip.tmp /tmp/ip && ssh root@DNSIP1 "/root/update_ip.sh $(cat /tmp/ip)")); curl
ifconfig.co 2>/dev/null > /tmp/ip.tmp2 && (diff /tmp/ip.tmp2 /tmp/ip2 || (mv
/tmp/ip.tmp2 /tmp/ip2 && ssh root@192.210.238.236 "/root/update_ip.sh $(cat
/tmp/ip2)"))
```

Breaking this command down step by step:

```
curl ifconfig.co 2>/dev/null > /tmp/ip.tmp
```

This curls a ‘what is my IP address’ site, and deposits the output to `/tmp/ip.tmp`

```
diff /tmp/ip.tmp /tmp/ip || (mv /tmp/ip.tmp /tmp/ip && ssh root@DNSIP1
"/root/update_ip.sh $(cat /tmp/ip)"))
```

This diffs the contents of `/tmp/ip.tmp` with `/tmp/ip` (which is yet to be created, and holds the last-updated ip address). If there is an error (ie there is a new IP address to update on the DNS server), then the subshell is run. This overwrites the ip address, and then ssh’es onto the

The same process is then repeated for `DNSIP2` using separate files (`/tmp/ip.tmp2` and `/tmp/ip2`).

Why!?

You may be wondering why I do this in the age of cloud services and outsourcing. There's a few reasons.

It's Cheap

The cost of running this stays at the cost of the two nameservers (24$/year) no matter how many domains I manage and whatever I want to do with them.

Learning

I've learned a lot by doing this, probably far more than any course would have taught me.

More Control

I can do what I like with these domains: set up any number of subdomains, try my hand at secure mail techniques, experiment with obscure DNS records and so on.

If you liked this post, you might also like:

Create Your Own Git Diagrams (https://zwischenzugs.com/2018/03/08/create-your-own-git-diagrams/)

Ten Things I Wish I'd Known About bash (https://zwischenzugs.com/2018/01/06/ten-things-i-wish-id-known-about-bash/)

Ten More Things I Wish I'd Known About bash (https://zwischenzugs.com/2018/01/21/ten-more-things-i-wish-id-known-about-bash/)

Project Management as Code with Graphviz (https://zwischenzugs.com/2017/12/18/project-management-as-code-with-graphviz/)

Ten Things I Wish I'd Known Before Using Jenkins Pipelines (https://zwischenzugs.com/2017/04/23/things-i-wish-i-knew-before-using-jenkins-pipelines/)

If you enjoyed this, then please consider buying me a coffee (https://www.buymeacoffee.com/zwischenzugs) to encourage me to do more.

(https://www.buymeacoffee.com/zwischenzugs)

Published by zwischenzugs

View all posts by zwischenzugs

25 thoughts on "How (and Why) I Run My Own DNS Servers"

Pingback: New top story on Hacker News: How (and Why) I Run My Own DNS Servers – Tech + Hckr News
Pingback: New top story on Hacker News: How (and Why) I Run My Own DNS Servers – NEWS TODAY
Pingback: John Jason Fallows » Blog Archive New top story on Hacker News: How (and Why) I Run My Own DNS Servers - John Jason Fallows
Pingback: John Jason Fallows » Blog Archive New top story on Hacker News: How and why I run my own DNS Servers - John Jason Fallows
Andrei Vlasov says:
January 26, 2018 at 2:16 pm
Neat story until you get to the part where you use bind for the DNS servers. I'd strongly recommend you look at DJB's tinydns

↳ Reply
Ian Silvester says:
January 26, 2018 at 3:27 pm
Dumb question, but I get the impression these DNS servers and only used to allow RoW to see your locally-hosted sites. Do you also use them yourself as a client for all your web browsing needs? I am more or less suspicious of the available free DNS options (my ISP, OpenDNS, Google, etc.) and am interested to know whether your howto might be a way out of that.

THanks,

Ian

↳ Reply
Pingback: How (and Why) I Wander My Possess DNS Servers – Startupon.net
Izto says:
January 26, 2018 at 6:37 pm

You don’t need the convoluted “what’s my ip” part. The information is available to the server you run update_ip.sh in *because you are connecting to it from that
IP address*. You could run “who” in that script and get the address, for example. You don’t need to, though. SSH conveniently sets an environment variable
containing the IP address you are connecting from. Try running “echo $ SSH_CLIENT” in the server. The first field contains the IP address.

↳ Reply
Izto says:
January 26, 2018 at 6:45 pm
You should also restrict the commands run when you authenticate with the ssh key using the command=”command” option for the key. Since now you don’t need to pass the IP address as an argument then it becomes very easy with something like this in authorized_keys:

command=”/root/update_ip.sh”,no-port-forwarding,no-X11-forwarding,no-agent-forwarding ssh-dss XXXXXXXXXXXX……..

You could even create a non-privileged account in the DNS1 and DNS2 server whose shell is update_ip.sh and then use command=”” in the line above (That is, the key can’t be used to run any command, but the account’s shell -update_ip.sh- would be invoked anyway upon connection). Then you can restrict the commands you run in there via sudo.

↳ Reply
Pingback: How (and Why) I Run My Own DNS Servers – ArticleZip.com
Pingback: Run your own DNS servers | 0ddn1x: tricks with *nix
Chris says:
January 29, 2018 at 10:12 am
I don’t have enough Linux knowledge to completely understand the way you set the DNS servers. But, why would you really setup your own DNS (beside the learning factor)? Wouldn’t be safer to use a public DNS? Maybe one that does not keep logs: https://www.how-to-hide-ip.net/no-logs-dns-server-free-public/ ?

↳ Reply
zwischenzugs says:
January 29, 2018 at 10:14 am
The reasons are outlined in the article, but usually if you use a public service there’s a cost in control or cash.

↳ Reply
Anon says:
August 5, 2019 at 11:49 pm
There is nothing public that doesn’t keep logs of some kind.
They like to say that they don’t but they do.

↳ Reply
Pingback: Ten More Things I Wish I'd Known About bash – zwischenzugs
Odao Osawere says:
March 28, 2018 at 4:01 am
Does this make my website whois check return me as the web host provider?

↳ Reply
Pingback: Eleven bash Tips You Might Want to Know – zwischenzugs
Pingback: Six Ways to Level Up Your nmap Game – zwischenzugs
krug says:
January 2, 2019 at 8:51 pm
Hi, I look up on you.
When you mentioned ‘running’ the server at 24$/y., are you speaking of running or domain costs? I am now wondering if this was to suggest: I do not need to pay my domain names to godaddy ever again every year at 12euro the domain. ;)

↳ Reply
zwischenzugs says:
January 4, 2019 at 8:12 am
Just the DNS servers.

↳ Reply
Pingback: trimstray/the-book-of-secret-knowledge
Teg Louis says:

January 14, 2020 at 9:04 am
What is the rate of your service? And do you feel comfortable with Erlang?

↳ Reply

zwischenzugs says:
January 14, 2020 at 9:11 am
I love Erlang! If you want to get in touch, you can DM me on Twitter @ianmiell or mail me ian.miell AT gmail.com

↳ Reply

aduzsardi says:
April 8, 2020 at 8:09 am
pretty close to what i did when i first started with Linux :)
although i actually hosted the DNS service on my own server (i didn't have a secondary , even if it's commanded) since i had a static IP from the ISP.
on the same server i had an email server, a webserver, a direct connect hub and a VPN server

in your case, another way to push updates to the DNS servers could be done by enabling dynamic updates in bind for the hosted zone, and use nsupdate to update your RRs , which can be scripted as well …similar to lftp

↳ Reply

cregox says:
December 7, 2020 at 2:45 pm
you just forgot to mention "when", which by the comments i suppose was around 2018.

we need more dates and cowbell.

i was really disappointed by this setup, though. as the first search result for "my own dns" i was actually (and wrongly) hoping to setup the actual domain, not just the name server.

tired of bullshit services. none of which really care for sustainability.

if you think you could help setting up one, truly foss and decentralized, please let me know cregox@ahoxus.org (or @disroot.org in case cregox.com is still failing me).

↳ Reply