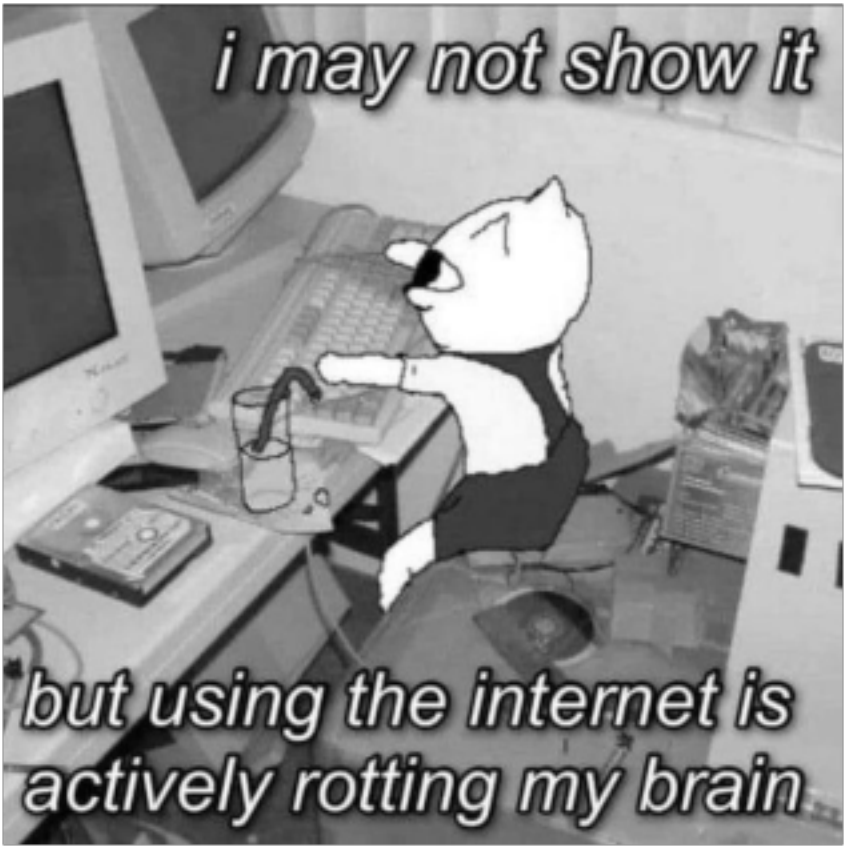
you have a homebrew server, make it public!

This publication contains my research ~~(as of 1st of December 2025)~~ as of March 2026! I am not an expert in self-hosting *in the slightest*, but that's what makes it so fun!



NOTE: my methods are not 100% cybersecure... I am but a newbie

Contents

What is a server?

↳ *my two server roomates; cerealbox (left) and chikorita (right). both are raspberry pi's <3*

A "server" is a heavily misunderstood term, in the sense that most people don't know what it entails and are often too afraid to ask. By doing a simple online image search, we end up with countless blue-tinted pictures of sterile rooms, filled with rows of gigantic machines that one would never dare, or even have access to, come close to— gatekept from whoever doesn't have the keys to the room. This conjured image signifies certain connotations of who has access to technological infrastructures, therefore who holds the power to information.

In reality, a server is but a computer, meant to route processes and services. It can take the form of a laptop, a raspberry pi, a phone [1], a microcontroller-turnt-charm hanging from a keychain [2] and be hosted in any place; a bedroom, a library, a shoebox; far removed from the widespread myth of the blue, sterile room.

Your IPs

> *IP stands for Internet Protocol*

Your homebrew server has two IP addresses, a **local** and a **public** one.

The **local IP** is static— but only devices in your network have access to it. Imagine the server as a boat, sailing the world. You have a bed in your boat and it's location is static, but only **relatively** to the boat.

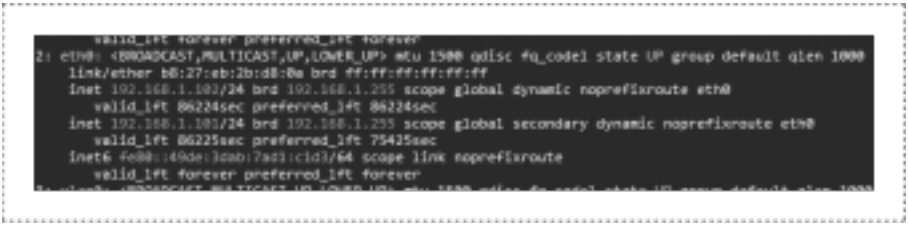
The **public IP** of your server is always unique and most likely not static, unless you have requested one (and paid for it). For example, I found out my provider changes mine once a day, probably during the night, and every time the router gets rebooted.

> ***UPDATE**: I found out that most routers have a semi-static public IP, that changes only when the router is restarted. I got the short end of the stick...*

Even though every network has a public IP, that doesn't mean that devices have access to it. More about that later!!!

Local

on the terminal, run `$ ifconfig` or `$ ip address` and look for the line saying `inet`. For example, my server's local IP is `192.168.1.101`



Public

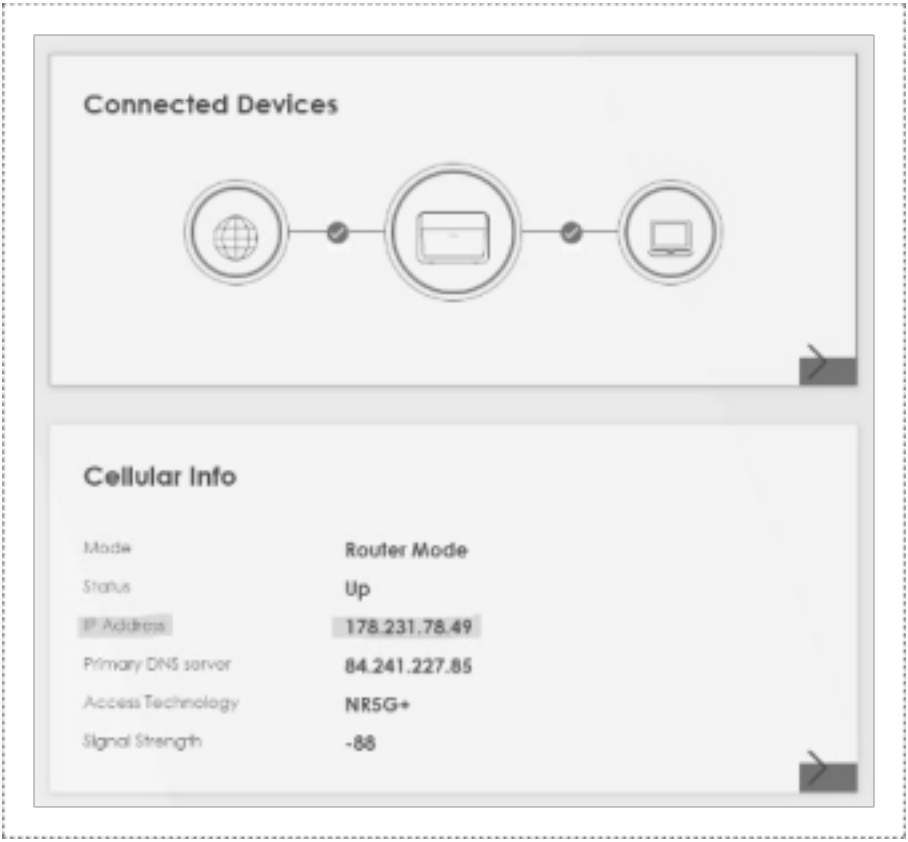
You can find it your public ip online, on websites like whatsmyip.com,or by running `curl https://ipinfo.io/ip` on the terminal of your server

IP address?

They know where I pee?

the Router

Chances are that through your router you have more control over your connection than you think!



↳ my public ip

On the bottom or back of your modem, you can find the admin config IP and some credentials. This IP is a local one and leads to a portal to your router's settings. Paste it on your browser search bar and

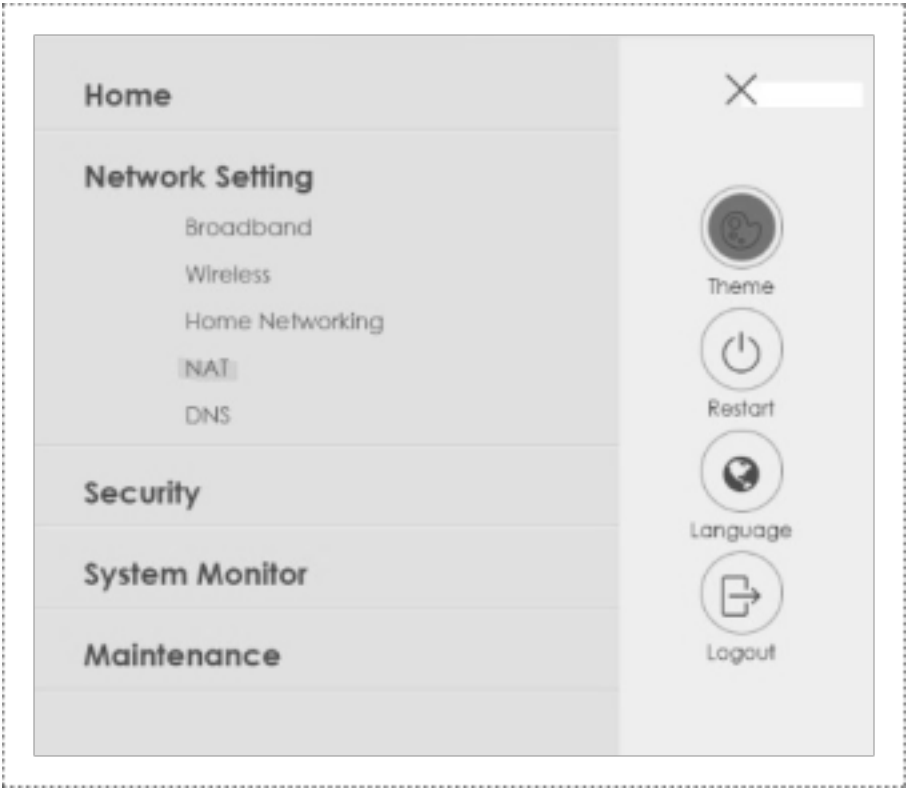
enter the credentials you have been given.

Static Local IP

Okay, I kind of lied before; the location of your bed in the boat is not static.

The local IP of your devices are also not static by design, but you can change that super easily under `Home Networking > Static DHCP` or some similar path. This is where I found it in my router admin settings. It is important to always know where your device is. It will make your life 200% easier, as you won't have to look for it every time.

Port Forwarding



↪ *where I found port forwarding settings*

Ports are numbered points of access to your server. Some of them are reserved for specific processes and to get traffic allowed into your server, you need to open up some of them. You can do this through the Router configuration portal (tables by homebrewserver.club[3])

`:80` for HTTP connection

```
Name: "HTTP"
Protocol: TCP
Device: 192.168.1.10
External Port: 80
Port to device: 80
```

`:443` for HTTPs,

```
Name: "HTTPS"
Protocol: TCP
Device: 192.168.1.10
External Port: 443
Port to device: 443
```

and `:22` for remote ssh access. For security reasons, this one should be set to a random port number, such as `:9999`, which you will then map to `:22` That is the secret door that only you and people you trust must know! 🤫

```
Name: "SSH"
      Protocol: TCP
      Device: 192.168.1.10
      External Port: 9999
      Port to device: 22
```

After this set of configurations, you should be able to access your browser remotely!

↳ *ssh set on a random port*

(Dyn)DNS

> *DNS stands for Domain Name System*

If the geographic credentials are a server's **public IP**, the **domain name** is the address.

Imagine having to go to school, when all the information you have are the geographic credentials of the entrance— `51.91843934978055, 4.488696626699141`. No one uses those numbers to find the building. We all go to `Wijnhaven 61, 3011 WJ Rotterdam` which is way more descriptive and memorable. So, unless you want people to look up your server by a series of numbers (that is, your **public IP**) you need to buy *(sigh)* a **domain name**, like `cerealbox.org`, from a domain name provider.

However, buying the domain name is not enough by itself. You have to create the connection between your **public IP** and the **domain name** so that they compliment each other. <3

This is what **DNS** is for! For now, as I run out of time, I set up the **DNS** through my domain name provider.

Dynamic DNS are basically DNS but for non-static IPs— they get regularly updated with the new IP.

~~I am still researching the best way to go about the~~

~~dynamic nature of my IP, but for now I just keep updating it on the DNS.~~

How I set up my DDNS

My intention with the server is to host a radio station, so I reached out to dot radio registry for a domain as it is a top-level one, ~~which means you can't just buy it from any domain provider (like godaddy or porkbun).~~ (**edit:** false, some top-level domains are available through those providers. Others however, like dot radio, are given selectively) They have their own DNS for their domains, ~~which at first was a good thing since I wanted to stay away from the huge, exploitative DNS providers resolvers.~~ (**edit:** they do dns resolution from an external service anyway) However, the non-static state of my IP, again, is a problem. I either need to find a Dynamic DNS service, which I plan to stay clear from, or use Cloudflare and their API[4].

Using Cloudflare API

API (Application Programming Interface) works as a middle-person between you and the software.

I made a script to call it from my server when the DNS configuration needs a change!

This is probably not the most efficient way to do it,

but I came up with it myself and it works<3

1. getip.sh to get IP

script that gets the current public IP and logs it into a txt file. then, I document the time when it happened.

```
curl ipinfo.io/ip >> /home/$USER/iplog.txt && echo >> /home/$USER/iplog.txt && date >> /home/$USER/timeip.txt
```

2. call API

`Profile > API Tokens > Create Token` & I gave it all DNS permissions

Then I wrote **ip.sh**, to run the API

```
#Gets last line of the ip log, so the
current IP
curip=$(tail -n 1  iplog.txt)
echo "ip is" $curip

#Gets last IP saved
previp=$(cat currentip.txt)
echo "prev is" $previp

#Checks if there has been a change
if [ "$previp" != "$curip" ]; then
#DNS Zone ID can be found on the
Overview
#DNS Record ID can be found under
"Inspect Element" [6]
        curl https://
api.cloudflare.com/client/v4/zones/
$DNS_ZONE_ID/dns_records/
$DNS_RECORD_ID \
        -X PATCH \
        -H 'Content-Type: application/
json' \
        #Your API Key
        -H "Authorization: Bearer
$API_TOKEN" \
        -d '{
                "name":
"$NAME_OF_RECORD",
                "ttl": 3600,
```

```
                "comment": "Domain
verification record",
                "content":
"'"$curip"'",
                "proxied": false
        }'
        #if you want Cloudflare to do
more than DNS for you, then "proxied":
true
        echo $curip > currentip.txt
        cat currentip.txt
else
        echo "all good!"
fi
```

3. setting up a cron

cronjob is a linux scheduler that runs on an interval that you set.

0 */6 * * * means that I set it to "*At minute 0 past every 6th hour*"[6], so every 6 hours.

look at your cron's code with `crontab -l`

and edit it with `crontab -e`

```
# m h  dom mon dow   command
0 */6 * * *     bash  /home/$USER/
getip.sh && sleep 10 && bash /home/
$USER/ip.sh >> /home/$USER/log.txt
```

this code runs the **getip.sh** script, waits 10 seconds, then runs the API script **ip.sh** and logs the result in **log.txt**

NGINX

```
sudo apt update -y
```

```
sudo apt install nginx
```

Creating Subdomains

Azuracast is running on port 90 and I want to access it through admin.fanon.radio

First, I go on the DNS records and add the following configuration, so that ALL subdomain traffic is redirected to the main IP

Type	Name	Content
A	fanon.radio	{{MY IP}}
CNAME	*	fanon.radio

`A` → Address, it indicates the IP of your domain

`CNAME` → Canonical Name. They point to a domain, never to an IP

`*` → wildcard domain, it refairs to ALL subdomains. I only have one now, which is *admin*, but if in the future I need more, I won't have to reconfigure the DNS record table.

Then, locally on my server I edit `/etc/nginx/sites-available` and nano `admin.fanon.radio`

```
server {

        listen 80;

        server_name admin.fanon.radio;

        location / {
                proxy_pass http://localhost:90;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
        }

}
```

then run `sudo ln -s /etc/nginx/sites-`

`available/admin.fanon.radio /etc/nginx/sites-enabled/`

after rebooting nginx, it should work!!!

SSL Certificate

I used certbot to get SSL certificates for my domains → https://certbot.eff.org/instructions?ws=nginx&os=snap

If you have already created subdomains, I suggest that you run `sudo certbot -d YOURMAINDOMAIN --nginx` for each one seperately, rather than using a batch option.

References

1. ↑ https://wiki.comphost.club/ — Server Charms Workshop, by actinomy
2. ↑ https://codeberg.org/actinomy/server-charms-workshop — Composting Mobile Phones - Hosting server on obsolete mobile phones, by Marie Verdeil and Rein van der Woerd
3. ↑ https://homebrewserver.club/fundamentals-port-forwarding.html — "Port Forwarding configuration for your home router" by homebrewserver.club
4. ↑ https://developers.cloudflare.com/api/
5. ↑ https://www.reddit.com/r/selfhosted/comments/1cle0oq/where_do_i_find_my_dns_record_id_in_cloudflare/
6. ↑ https://crontab.guru

Colophon

designed using **paged.js**

fonts in use:

FT88 Regular & **FT88 Italic** redesigned by Oriane Charvieux and Mandy Elbé
BBB Herthey Futural 40 by Bye Bye Binary Collective
BBB ReadMe SemiBold Italic by Bye Bye Binary Collective

by eleni fili<3