Freifunk S"u+00f"dholstein Documentation

Release 2.2

Grotax, ffsh

09.07.2023

Inhalte

KAPITEL 1

Infrastruktur

1.1 Netzwerk

Network IPv4: `10.144.0.0/16`

Network IPv6: `fddf:0bf7:80::/48`

1.2 Gateways

Name	ULA	IPv6	RFC1918	DHCP	ICVPN-Transit	Mesh MAC(s)	B.A.T.M.A.N. adv. MAC(s)	Standort	Betreuer	Exit/VPN-Dienst	Status
Barnitz	fddf:0bf7:80[illegible]	[illegible]	10.144.48.[illegible]	10.144.48.2-10.144.63.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:00:[illegible]	Hetzner (nbg)	ul	VPN	online
Beste	fddf:0bf7:80[illegible]	[illegible]	10.144.64.[illegible]	10.144.64.2-10.144.79.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:00:[illegible]	Hetzner (fsn)	ul	VPN	online
Bille	fddf:0bf7:80[illegible]	[illegible]	10.144.80.[illegible]	10.144.80.2-10.144.95.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:00:[illegible]	N80-cup	fr	VPN	online
Brunsbach	fddf:0bf7:80[illegible]	[illegible]	10.144.96.[illegible]	10.144.96.2-10.144.111.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:00:[illegible]	N96-cup	be	VPN	online
Heilsau	fddf:0bf7:80[illegible]	[illegible]	10.144.112.[illegible]	10.144.112.2-10.144.127.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:01:[illegible]	Hetzner (hel)	ul	VPN	online
Hopfenbach	fddf:0bf7:80[illegible]	[illegible]	10.144.128.[illegible]	10.144.128.2-10.144.143.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:01:28				
Krummbach	fddf:0bf7:80[illegible]	[illegible]	10.144.144.[illegible]	10.144.144.2-10.144.159.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:01:44				
Piepenbek	fddf:0bf7:80[illegible]	[illegible]	10.144.160.[illegible]	10.144.160.2-10.144.175.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:01:60				
Strusbek	fddf:0bf7:80[illegible]	[illegible]	10.144.176.[illegible]	10.144.176.2-10.144.191.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:01:76				
Sylsbek	fddf:0bf7:80[illegible]	[illegible]	10.144.192.[illegible]	10.144.192.2-10.144.207.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:01:92	N192-cup	ul	VPN	online
Trave	fddf:0bf7:80[illegible]	[illegible]	10.144.208.[illegible]	10.144.208.2-10.144.223.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:01:[illegible]	Hetzner (fsn)	ul	VPN	online
Viehbach	fddf:0bf7:80[illegible]	[illegible]	10.144.224.[illegible]	10.144.224.2-10.144.239.254	n/a	00:5b:27:80[illegible]	00:5b:27:81:02:24				

1.3 Karte

Die Karte kann unter https://map.freifunk-suedholstein.de erreicht werden. Die Karte wird auf dem Gateway Hopfenbach betrieben. Sie basiert auf dem meshviewer von Freifunk Regensburg. Unsere Konfiguration findet man in unserem fork auf GitHub.

Unsere Grafana instanz ist unter https://map.freifunk-suedholstein.de/grafana erreichbar.

```
# meshviewer.json
https://map.freifunk-suedholstein.de/data/meshviewer.json
# nodes.json (v2)
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
https://map.freifunk-suedholstein.de/data/nodes.json
# nodelist.json
https://map.freifunk-suedholstein.de/data/nodelist.json
```

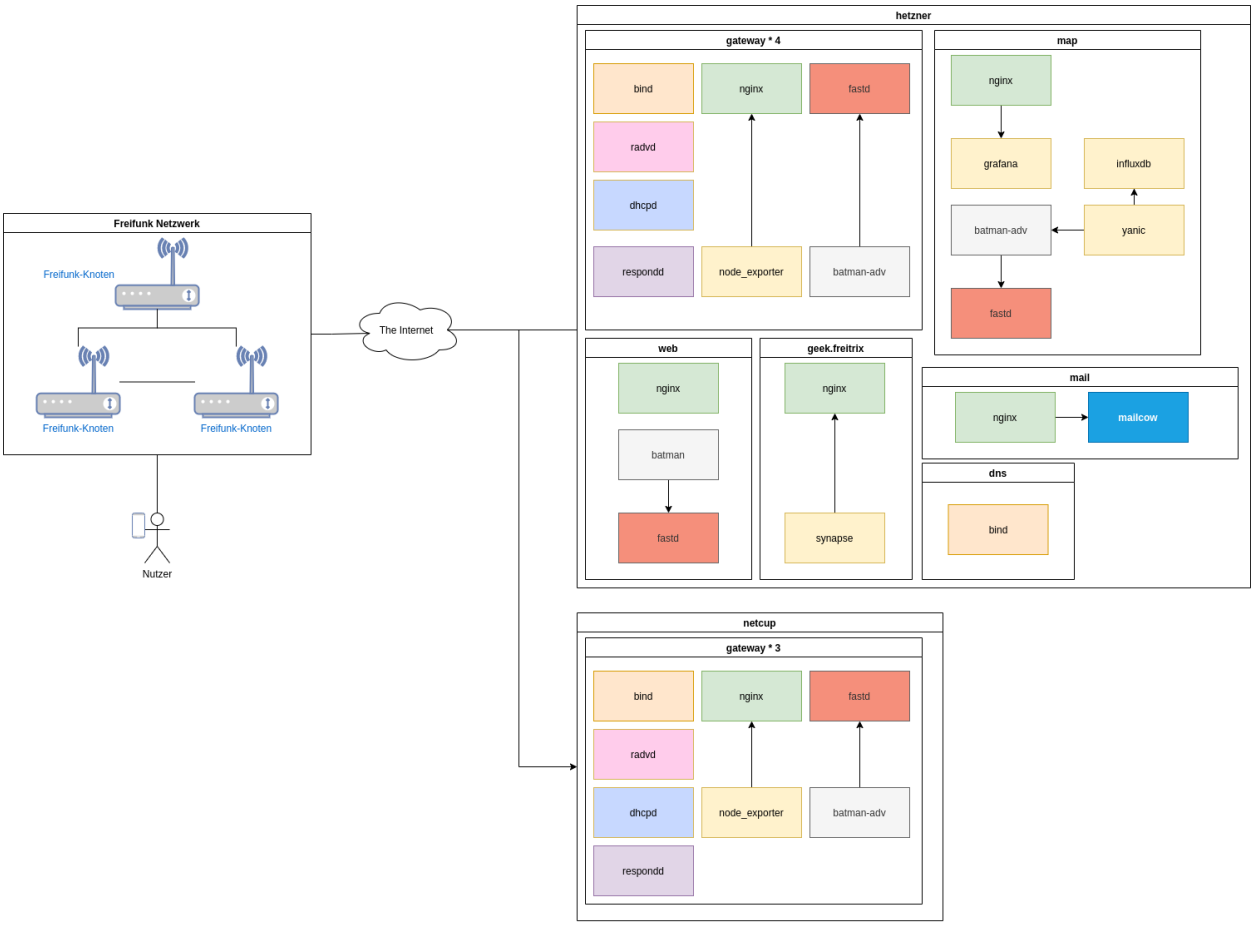
1.4 GitHub

Fast alle Daten, welche für den Betrieb notwendig sind, werden unter https://github.com/ffsh gespeichert.

KAPITEL 2

Gateway Konfiguration

Dies ist die Dokumentation für die Gateways von Freifunk Südholstein.



In 2021 haben wir für die Einrichtung der Gateways auf Ansible umgestellt. Davor war die Einrichtung eines Gateways

ein manueller Prozess, der bereits bei kleineren Fehlern zu stundenlangen Rätseln führen konnte.

Ansible verkürzt die Einrichtung eines Gateways auf etwa 30min.

Unser Ansible Repository liegt auf GitHub: https://github.com/ffsh/ansible

2.1 Ansible installieren

Ansible muss auf deinem lokalen System installiert werden. In den meisten Linux Distributionen wirst du ein oder mehrere Ansible Pakete finden. Unter Manjaro ist das „ansible“ Paket ausreichend. Mehr Optionen findest du hier: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

2.2 Struktur

Unser Ansible Repository ist relativ einfach gehalten, dadurch bleibt es übersichtlich. Es ist allerdings auch nicht perfekt.

2.2.1 setup.yml

Hier werden neben ein paar Einstellungen vor allem die Rollen definiert und in welcher Reihenfolge sie auf dem Gateway angewendet werden sollen.

2.2.2 hosts.yml

Hier werden alle variablen, welche von den Rollen benötigt werden gespeichert. Abgesehen von den Geheimen natürlich. Die YAML Struktur empfiehlt sich zur besseren Übersichtlichkeit.

2.2.3 host_vars/

Hier gibt es pro Gateway eine Datei mit verschlüsselten Inhalt. Dazu gehören der private fastd key sowie optional eine Wiregaurd Konfiguration für einen „exit“ VPN.

2.2.4 roles/

Hier werden die Rollen gespeichert, wenn etwas nicht funktioniert liegt der Fehler vermutlich hier. Die Rollen fokussieren sich in der Regel auf genau eine Komponente und ihre Abhängigkeiten.

2.3 Secrets anlegen und bearbeiten

Für unser Gateway brauchen wir natürlich auch geheime Daten, welche wir nicht öffentlich im git Repository ablegen wollen. Dafür bietet Ansible eine integrierte Funktion, welche eine verschlüsselte Datei mit den gewünschten Daten erstellt. Die Readme Datei im Repository erklärt, wie neue Secrets erstellt werden und wie man sie bearbeitet.

Mehr zu diesem Thema: https://docs.ansible.com/ansible/latest/user_guide/vault.html

2.4 Hinweise zur Nutzung

Ansible ist relativ einfach zu benutzen, in der Readme des Repository findest du die üblichen nützlichen Befehle. Standardmäßig wird Ansible eine Änderung auf allen konfigurierten Gateways einspielen. Manuelle Änderungen auf einem Gateway sind ok, sie sollten allerdings so schnell wie möglich in der entsprechenden Ansible Rolle festgehalten werden. Es empfiehlt sich, diese Anpassungen zu testen. Es ist ratsam sich beim Schreiben von Rollen bei den Standard Ansible Rollen zu bedienen, das ist allerdings nicht immer möglich.

2.5 Rollen

Die Rollen enthalten die Skripte welche auf den Gateways ausgeführt werden. Sie sollen den Server durch eine bestimmte Konfiguration in ein Gateway verwandeln.

2.6 Architektur

Dieser Abschnitt soll die Architektur unserer Gateways beschreiben.

2.6.1 Hardware

Für die „Hardware“ setzen wir aktuell die kleinsten virtuellen Maschinen von Hetzner und Netcup ein und fahren mit diesem System sehr bisher sehr gut. Die Knoten verteilen sich mehr oder weniger gleichmäßig über alle Gateways.

2.6.2 VPN

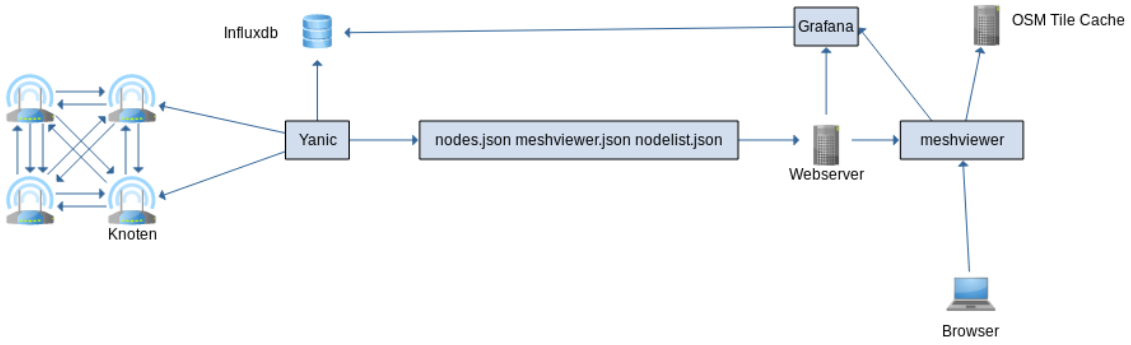
Wir setzen als VPN Lösung zwischen Knoten und Gateway und auch zwischen den Gateways sowie den Servern für die Karte und die Firmware aktuell auf fastd. fastd ist nicht die beste VPN Lösung und zeigt auf den üblichen Routern keine besonders gute Performance, ist aber relativ einfach in der Handhabung und funktioniert reibungslos mit Batman Advanced.

Als „Exit VPN“ setzen wir Wireguard ein, um uns vor urheberrechtlichen Problemen zu schützen. Wireguard ist sehr performant und ist für diesen Fall sehr einfach zu konfigurieren.

KAPITEL 3

map

Diese Dokumentation behandelt das Aufsetzen von https://map.freifunk-suedholstein.de



3.1 Vorraussetzungen

Wie für all unsere Service Server verwenden wir Ubuntu als Basis. Als Hardware verwenden für den gesammten Stack eine VM von Hetzner (CPX21) 2vCores 4GB RAM 40GB disk. Für unsere Community reicht dieses Setup bisher aus.

3.2 yanic

yanic sammelt von den Knoten Daten, welche dann auf einer Karte angezeigt werden können, früher wurde hierfür Alfred benutzt. yanic ist in go geschrieben also installieren wir eine neue Version von go. golang

```
wget https://dl.google.com/go/go1.17.5.linux-amd64.tar.gz
# Bitte sha256 vergleichen https://golang.org/dl/
tar -C /usr/local -xzf go1.17.5.linux-amd64.tar.gz
rm go1.17.5.linux-amd64.tar.gz
```

In `~/.bashrc`

```
GOPATH=/opt/go
PATH=$PATH:/usr/local/go/bin:$GOPATH/bin
```

Hier musst du dich einmal abmelden und neu anmelden damit die Variablen auch gesetzt werden.

Nach dem Anmelden kann man prüfen ob die Variablen korrekt gesetzt wurden.

```
echo $GOPATH
/opt/go
```

Mit `whereis` go prüfen ob go gefunden wird:

```
go: /usr/local/go /usr/local/go/bin/go
```

Dann wird yanic installiert.

```
go get -v -u github.com/FreifunkBremen/yanic
```

Die Konfiguration von Yanic wird in `/etc/yanic.conf` angelegt:

```
[respondd]
enable           = true
synchronize      = "1m"
collect_interval = "1m"

[respondd.sites.ffsh]
domains            = ["ffsh", "ffod", "ffrz"]

[respondd.sites.ffod]
domains            = []

[respondd.sites.ffrz]
domains            = []

[[respondd.interfaces]]
ifname          = "bat0"

[[respondd.interfaces]]
ifname            = "bat0"
multicast_address = "ff02::2:1001"

[webserver]
enable  = false
bind    = "127.0.0.1:8080"
webroot = "/var/www/html/meshviewer"

[nodes]
state_path    = "/var/lib/yanic/state.json"
prune_after   = "14d"
save_interval = "2m"
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
offline_after = "10m"

[[nodes.output.geojson]]
enable   = true
path = "/var/www/map/data/nodes.geojson"

[[nodes.output.meshviewer-ffrgb]]
enable   = true
path = "/var/www/map/data/meshviewer.json"

[nodes.output.meshviewer-ffrgb.filter]
no_owner = true
blacklist = ["f4f26d4a2ecc"]

[[nodes.output.meshviewer]]
enable = true
version    = 2
nodes_path = "/var/www/map/data/nodes.json"
graph_path = "/var/www/map/data/graph.json"

[nodes.output.meshviewer.filter]
no_owner = true
blacklist = ["f4f26d4a2ecc"]

[[nodes.output.nodelist]]
enable   = true
path = "/var/www/map/data/nodelist.json"

[nodes.output.nodelist.filter]
no_owner = true
blacklist = ["f4f26d4a2ecc"]

[database]
delete_after    = "30d"
delete_interval = "1h"

[[database.connection.influxdb]]
enabled  = true
address  = "http://localhost:8086"
database = "ffsh"
username = ""
password = ""

[database.connection.influxdb.tags]

[[database.connection.graphite]]
enable   = false
address  = "localhost:2003"
prefix   = "freifunk"

[[database.connection.respondd]]
enable   = false
type     = "udp6"
address  = "stats.bremen.freifunk.net:11001"

[[database.connection.logging]]
enable   = false
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
path      = "/var/log/yanic.log"
```

Wir können testen ob yanic funktioniert in dem wir eine manuelle Anfrage stellen hier an das Gateway Hopfenbach:

```
yanic query --wait 5 br-ffsh "fddf:bf7:80::48:1"
```

Damit yanic auch als Deamon läuft legen wir noch einen Service an.

```
sudo cp /opt/go/src/github.com/FreifunkBremen/yanic/contrib/init/linux-systemd/yanic.
↪service /lib/systemd/system/yanic.service
sudo systemctl daemon-reload
```

3.3 influxdb

Influxdb dient als Datenbank für `yanic`

Achtung hier wird Influxdb 1.x (aktuell 1.8.10) installiert, die aktuelle version ist 2.0 (diese wird aktuell nicht von yanic unterstützt).

```
wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/lsb-release
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable" |␣
↪sudo tee /etc/apt/sources.list.d/influxdb.list
```

```
sudo apt install influxdb influxdb-client
```

Nun sichern wir die influxdb ab `/etc/influxdb/influxdb.conf`

Hier werden nur die empfohlenen Anpassungen beschrieben: Noch vor der `[meta]` Sektion setzen wir, sonst wäre der port 8088 überall offen.

```
bind-address = "localhost:8088"
```

Weiter unten bei `[admin]` das gleiche:

```
bind-address = "localhost:8083"
```

kurz danach in `[http]`

```
bind-address = "localhost:8086"
```

```
systemctl restart influxdb
```

Nun sollte influxdb nur noch auf localhost erreichbar sein, prüfen kann man dies mit `netstat -tlpn`

3.4 Grafana

Grafana kann Graphen erstellen welche im meshviewer eingebunden werden können. Hier wird Grafana über eine Repository installiert.

```
deb https://packagecloud.io/grafana/stable/debian/ stable main
curl https://packagecloud.io/gpg.key | sudo apt-key add -
sudo apt-get update
sudo apt-get install grafana
```

Da Grafana bei uns hinter einem Proxy laufen soll, setzen wir auch hier alle IPs auf `localhost`. Am besten einmal am Ende prüfen ob alles richtig konfiguriert ist mit `netstat -tlpn`.

Ein wichtiger Punkt ist der öffentliche Zugang, damit die Statistiken auch von Besuchern abgerufen werden können.

```
#################################### Anonymous Auth ##########################
[auth.anonymous]
# enable anonymous access
enabled = true

# specify organization name that should be used for unauthenticated users
org_name = Freifunk Südholstein

# specify role for unauthenticated users
org_role = Viewer
```

Die Organisation kann man als Admin in Grafana anlegen.

3.5 meshviewer

Für den Meshviewer installieren wir als erstes nodejs und yarn

3.5.1 nodejs

Wir brauchen ein aktuelles nodejs das finden wir auf nodejs.org Wir benutzen die LTS Variante 16.x

```
curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash -
sudo apt-get install -y nodejs
```

3.5.2 yarn

Dann installieren wir yarn folge einfach den Anweisungen der Dokumentation.

3.5.3 meshviewer-rgb

Nun installieren wir den meshviewer selbst. Im web Verzeichnis `/var/www/`

```
git clone https://github.com/ffsh/meshviewer.git
cd meshviewer
yarn
```

Nun muss die Konfiguration in `meshviewer/config.js` eventuell noch angepasst werden.

Danach `yarn run gulp` Nun muss nur noch ein Webserver meshviewer/build ausliefern.

3.6 Tile-cache mit nginx

Für den Meshviewer benötigt man einen Tile-Server, der die Karte als einzelne Kacheln ausliefert. Wir verwenden dabei das kostenlose und freie Angebot von OpenStreetMap. Damit die Server von OpenStreetMap weniger stark belastet werden verwenden wir einen Tile-Cache. Bei einer Anfrage für eine Karten-Kachel fragt der Browser den Cache, hat dieser die Datei bereits, so liefert er sie direkt aus. Hat er sie nicht, so fragt er bei den OpenStreetMap Servern und speichert die Datei in seinem Cache. Für die einfache Umsetzung haben ein paar Freifunker an einer Konfiguration für nginx gearbeitet, welche genau das umsetzt.

Voraussetzungen: - nginx erreichbar unter der entsprechenden Domain - TLS mit gültigem Zertifikat (Let's Encrypt) - ein wenig Speicherplatz

```
#
# Nginx >= 1.9.15 - 1.10.1 recommended
#
# Thanks to https://github.com/cbricart

proxy_cache_path /var/www/tilecache/osm
    levels=1:2 inactive=7d
    keys_zone=tilecache:64m
    max_size=500M;

upstream osm_tiles {
    server a.tile.openstreetmap.org;
    server b.tile.openstreetmap.org;
    server c.tile.openstreetmap.org;
    keepalive 16;
}

server {
    listen 80;
    listen [::]:80;

    server_name tiles.freifunk-suedholstein.de;

    # Redirect all HTTP requests to HTTPS with a 301 Moved Permanently response.
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;

    ssl_dhparam /etc/ssl/dhparam.pem;

    server_name tiles.freifunk-suedholstein.de;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-
↪ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-
↪SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
↪SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-
↪SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-
↪ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-
↪RSA-AES256-SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-
↪SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-
↪SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS';
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    ssl_prefer_server_ciphers on;

    add_header Strict-Transport-Security max-age=15768000;

    ssl_stapling on;
    ssl_stapling_verify on;

    ssl_certificate /etc/letsencrypt/live/tiles.freifunk-suedholstein.de/fullchain.
↪pem;
    ssl_certificate_key /etc/letsencrypt/live/tiles.freifunk-suedholstein.de/privkey.
↪pem;

    root /var/www/tilecache/html;

    location / {
        try_files $uri @osm;
    }

    location @osm {
        proxy_pass http://osm_tiles;
        proxy_http_version      1.1;
        proxy_set_header    Connection "";
        proxy_set_header    Accept-Encoding "";
        proxy_set_header    User-Agent "Mozilla/5.0 (compatible; OSMTileCache/1.0;␣
↪+mailto:noc@freifunk-suedhoslтein.de; +https://map.freifunk-suedholstein.de/)";
        proxy_set_header    Host tile.openstreetmap.org;

        proxy_temp_path     /var/www/tilecache/temp;
        proxy_cache         tilecache;
        proxy_store         off;
        proxy_cache_key     $uri$is_args$args;
        proxy_ignore_headers Expires Cache-Control;
        proxy_cache_valid   200 301 302 7d;
        proxy_cache_valid   404 1m;
        proxy_cache_valid   any 1m;
        proxy_next_upstream   error timeout invalid_header http_500 http_502 http_503␣
↪http_504 http_403 http_404;
        proxy_cache_use_stale error timeout updating invalid_header http_500 http_502␣
↪http_503 http_504 http_403 http_404;

        proxy_hide_header   Via;
        proxy_hide_header   X-Cache;
        proxy_hide_header   X-Cache-Lookup;

        expires 7d;
        }
}
```

3.7 Grafna cache mit nginx

Da Grafana ab Version 7.0 das rendern der images, welche wir auf der Karte einbetten, anders rendert als früher mussten wir einen Cache einrichten. Siehe https://github.com/ffrgb/meshviewer/issues/304

Basierend auf den Kommentaren haben wir auch eine Konfiguration zusammengestellt.

```
proxy_cache_path /var/cache/nginx/grafana
        keys_zone=grafana:10m
        max_size=128m;

server {
    listen 80;
    listen [::]:80;

    server_name grafana.freifunk-suedholstein.de;

    # Redirect all HTTP requests to HTTPS with a 301 Moved Permanently response.
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;

    ssl_dhparam /etc/ssl/dhparam.pem;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-
↪ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-
↪SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
↪SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-
↪SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-
↪ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-
↪RSA-AES256-SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-
↪SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-
↪SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS';
    ssl_prefer_server_ciphers on;

    add_header Strict-Transport-Security max-age=15768000;

    ssl_stapling on;
    ssl_stapling_verify on;

    ssl_certificate /etc/letsencrypt/live/grafana.freifunk-suedholstein.de/fullchain.
↪pem;
    ssl_certificate_key /etc/letsencrypt/live/grafana.freifunk-suedholstein.de/
↪privkey.pem;

    server_name grafana.freifunk-suedholstein.de;

    location / {
        proxy_pass http://127.0.0.1:3000;
    }

    location /render/ {
        proxy_http_version 1.1;
        proxy_pass http://127.0.0.1:3000/render/;

        proxy_cache grafana;
        proxy_cache_valid 300s;
        proxy_cache_lock on;
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        proxy_cache_lock_age 60s;
        proxy_cache_lock_timeout 60s;

        proxy_hide_header Cache-Control;
        proxy_hide_header Expires;
        proxy_hide_header Pragma;

        proxy_ignore_headers Cache-Control Expires;

        expires 150s;
    }

}
```

KAPITEL 4

Firmware

Die Firmware von Freifunk Südholstein basiert auf Gluon, wir verwenden dabei in der Regel keine Community-Packages, dadurch ist unsere Firmware sehr stabil und wir können stets auf dem neusten Stand bleiben.

Das site Repository findet man auf GitHub. Freifunk Südholstein hat drei update Kanäle, die regelmäßig bedient werden.

- testing
- rc
- stable

Eine neue Firmware wird stets zuerst im testing Kanal verbreitet, wir haben nur wenige Knoten, welche den testing oder rc Kanal benutzen. Wir hoffen jedoch, dadurch zumindest grobe Probleme rechtzeitig zu entdecken.

4.1 Firmware Releases

Ein neuer Release beginnt in der Regel mit einem neuen Release von Gluon, dabei können Änderungen an der site Konfiguration nötig werden. Gelegentlich gibt es auch neue Features, ob diese per Update ausgerollt werden sollen wird dabei mit einer Folgen-Nutzen abschätzung Abgewegt. Sind die Risiken für Fehlfunktionen zu groß, so wird die Funktion zunächst nicht eingeführt.

Sobald die site config angepasst wurde, wird ein neuer git tag angelegt, die Version setzt sich dabei aus der Gluon-Version sowie einer extra Stelle für Änderungen von FFSH zusammen.

- Gluon Release = 2020.2.3
- FFSH Release = 0
- FFSH+Gluon Release 2020.2.3.0

Durch die extra Stelle kann FFSH beliebig viele Releases auf Basis derselben Gluon-Version erstellen. Nachdem ein Release erfolgreich gebaut wurde, wird der Release signiert und per Webserver an die Knoten verteilt.

Dabei gilt die Reihenfolge testing, rc, stable. Nachdem die Knoten eines Kanals aktuallisiert wurden wird in der Regel etwa eine Woche gewartet bis die Firmware auf den folgenden Kanal ausgerollt wird. Aufgrund der geringen Anzahl an Knoten werden testing und rc gelegentlich parallel ausgerollt.

4.2 Firmware Pipeline

Für die Firmware Pipeline benutzt Freifunk Südholstein GitHub Actions, sowie abgewandelte Skripte vom Gluon Projekt.

Im site Repository gibt es ein actions Verzeichnis, dort liegen ein paar scripte.

- `generate-actions.py` ist ein leicht abgewandeltes script vom Gluon Projekt, welches den GitHub Workflow generiert.
- `install-dependencies.sh` installiert zusätzliche Pakete in dem Cointainer von GitHub.
- `run-build-local.sh` kann zum bauen auf der lokalen maschine verwendet werden, je nach Bedaf anpassen, für die Fehlersuche gedacht.
- `run-build.sh` ist das eigentliche build Skript, bei einem neuen release wird hier die Version angepasst.

Die vom `generate-actions.py` generierte `build-gluon.yml` Datei liegt in `.github/workflows`, das Skript muss in der Regel nur dann neu ausgeführt werden, wenn ein neuer Gluon Release ein *target* hinzufügt oder entfernt. Das Script muss angepasst werden, wenn es einen neuen major Release gibt z.B. 2020->2021, dann muss der tags liste ein neuer regex Ausdruck hinzugefügt werden.

Experimentell ist bisher der Workflow im `release.yml`, das Ziel ist es nach dem erfolgreichen Bauen der Firmware manuell einen Release auf GitHub anzulegen. Der Workflow soll dann über API ein Firmware Release-Archiv zusammenstellen und als Anhang an den Release beifügen.

Das Bauen der Firmware dauert aktuell etwas mehr als eine Stunde, wenn GitHub durchschnittlich ausgelastet ist. Sollte GitHub sich eines Tages dazu entscheiden, die Ressourcen zu verringern, gibt es die Möglichkeit einen self-hosted runner z.b. in der Hetzner Cloud zu installieren.

4.2.1 Lokal Bauen

Die Firmware kann auch lokal gebaut werden, dies ist vor allem nützlich, wenn man einen speziellen release erstellen möchte, oder zur Fehlersuche.

Dafür können natürlich ganz klassisch alle Abhängigkeiten installiert werden, Infos gibt es in der Gluon Dokumentation. Allerdings kann es dabei leicht zu Problemen kommen, alternativ liegt im site Repository ein Dockerfile.

```
docker build . --tag gluon
docker run --name gluon --mount type=bind,source=$(pwd),target=/gluon gluon
```

4.3 Site Repository

Das site Repository wird auf GitHub gehostet. Es gibt einen main branch, auf diesem Branch findet die hauptsächliche Entwicklung statt. Daneben kann es feature branches geben, die ausgehend vom main branch eine experimentelle oder abweichende Funktionalität mitbringen.

Es gibt kein Changelog im Repository, auf freifunk-suedholstein.de sind alle Beiträge zur Firmware getagt und man kann einen schnellen Überblick über die Veränderungen bekommen.

KAPITEL 5

monitoring

Diese Dokumentation behandelt die Konfiguration unseres monitoring servers https://stats.freifunk-suedholstein.de

5.1 Vorraussetzungen

Der aktuelle Server läuft mit Debian 12 in der Hetzner Cloud auf einer CX11 instanz (1VCPU 2GB RAM 20GB disk)

5.2 Prometheus

Prometheus ist das Herz des monitoring systems, Prometheus sammelt die Daten von den Gateways und anderen Servern und speichert sie in seiner eigenen TSDB. Prometheus stellt eine API bereit mit der Daten aus der TSDB abgefragt werden können.

Wir verwenden die Prometheus version aus dem Debian repository.

```
apt install prometheus
```

Die Konfiguration liegt in `/etc/prometheus/prometheus.yml`

Der systemd service kann per `systemctl status prometheus.service` geprüft werden.

Die Konfiguration:

```
# my global config
global:
scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every␣
↪1 minute.
evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1␣
↪minute.
# scrape_timeout is set to the global default (10s).

# Alertmanager configuration
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
alerting:
alertmanagers:
   - static_configs:
      - targets:
         # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_
↪interval'.
rule_files:
# - "first_rules.yml"
# - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
# The job name is added as a label `job=<job_name>` to any timeseries scraped from␣
↪this config.
- job_name: "prometheus"

   # metrics_path defaults to '/metrics'
   # scheme defaults to 'http'.

   static_configs:
      - targets: ["localhost:9090"]

- job_name: "gateways"

   metrics_path: stats/metrics
   scheme: https
   basic_auth:
      username: '...'
      password: '...'

   static_configs:
      - targets: ["barnitz.freifunk-suedholstein.de", "beste.freifunk-suedholstein.de
↪", "bille.freifunk-suedholstein.de", "brunsbach.freifunk-suedholstein.de", "heilsau.
↪freifunk-suedholstein.de", "sylsbek.freifunk-suedholstein.de", "trave.freifunk-
↪suedholstein.de"]

- job_name: "services"

   metrics_path: stats/metrics
   scheme: https

   static_configs:
      - targets: ["mail.freifunk-suedholstein.de", "map.freifunk-suedholstein.de",
↪"web.freifunk-suedholstein.de"]
```

Username und Passwort sind in unserer KeePass Datei.

Zusätzlich beschränken wir Prometheus darauf auf localhost zu lauschen.

`/etc/default/prometheus`

```
# Set the command-line arguments to pass to the server.
# Due to shell escaping, to pass backslashes for regexes, you need to double
# them (\\d for \d). If running under systemd, you need to double them again
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# (\\\\d to mean \d), and escape newlines too.
ARGS="--web.listen-address=127.0.0.1:9090"
```

Diese Einstellungen werden von systemd bei einem Neustart geladen.

```
systemctl restart prometheus.service
```

Die Daten von prometheus liegen in:

```
/var/lib/prometheus/metrics2
```

5.3 Grafana

Grafana wird verwendet um die Daten aus prometheus zu visualiseren. Die installation und Konfiguration ist hier nicht detailiert.

Die wesentlichen punkte sind.

- grafana installation https://grafana.com/docs/grafana/latest/setup-grafana/installation/
- ngninx reverse proxy, letsencrypt
- prometheus als Daten-Quelle konfigurieren
- grafana konfiguration anpassen, grafana sollte nur auf localhost lauschen
- dashboards einrichten, für den node_exporter gibt es fertige dashboards aus der Community

KAPITEL 6

Indices and tables

- genindex
- modindex
- search