Instalace DNS server?

Instalace DNS serverů, různé varianty, DOH, DOT, Unbound..

- 1 - DNS servery s DoH na Raspberry Pi: Kompletní průvodce bez unbound
- Změna hesla do Pihole
- Testování DNS serveru, kompletní příkazy
- DNS server, Pi-hole, DoH + záložní DNS2 server na Raspberry PI5

1 - DNS servery s DoH na Raspberry Pi: Kompletní pr? vodce bez unbound

Úvod do problematiky DNS a bezpe? nosti

DNS (Domain Name System) je základní součást internetu, která překládá doménová jména (např. zelina.eu) na IP adresy (např. 192.168.54.2). Tradiční DNS komunikace však probíhá nešifrovaně, což představuje několik bezpečnostních rizik:

1. **Sniffing (odposlech)** - Kdokoli s přístupem k síťovému provozu může vidět, jaké domény navštěvujete
2. **DNS Spoofing/hijacking** - Útočník může přesměrovat DNS dotazy na podvržené servery
3. **Man-in-the-middle útoky** - Zachycení a manipulace s DNS dotazy mezi klientem a serverem

Proto vznikly technologie jako **DNS over HTTPS (DoH)**, které šifrují DNS komunikaci pomocí HTTPS protokolu. DoH nabízí:

- Šifrování DNS provozu
- Ochranu před odposlechem
- Zvýšenou soukromost uživatelů
- Odolnost proti DNS manipulacím

Poskytovatelé rekurzivních DoH server ?

Nejen Cloudflare nabízí DoH služby. Mezi významné poskytovatele patří:

1. **Cloudflare**: `https://cloudflare-dns.com/dns-query` (1.1.1.1)
2. **Google**: `https://dns.google/dns-query` (8.8.8.8, 8.8.4.4)
3. **Quad9**: `https://dns.quad9.net/dns-query` (9.9.9.9) - zaměřený na bezpečnost

4. **AdGuard**: `https://dns.adguard.com/dns-query` - s funkcí blokování reklam
5. **OpenDNS**: `https://doh.opendns.com/dns-query` (208.67.222.222)
6. **NextDNS**: `https://dns.nextdns.io` - nabízí mnoho konfiguračních možností

Implementace dvou nezávislých DNS server? s DoH na Raspberry Pi 5

Vytvoříme řešení založené na `dnscrypt-proxy` v Dockeru, které nám umožní používat DoH s možností zálohy mezi poskytovateli.

P?íprava pro první server (DNS1)

Připojte se k prvnímu Raspberry Pi a vytvořte adresářovou strukturu:

```
# Vytvořte adresáře pro DNS server
sudo mkdir -p /opt/docker/dns1/config
cd /opt/docker/dns1
```

Vytvořte soubor `docker-compose.yml`:

```
sudo nano docker-compose.yml
```

Vložte následující obsah:

```
version: '3'
services:
  dnscrypt-proxy:
    image: klutchell/dnscrypt-proxy:latest
    container_name: dns1_dnscrypt
    restart: always
    network_mode: host
    volumes:
      - ./config:/config
    environment:
      - TZ=Europe/Prague
    cap_add:
      - NET_ADMIN
```

Nyní vytvořte konfigurační soubor:

```
sudo nano config/dnscrypt-proxy.toml
```

Vložte tuto konfiguraci:

```
# Naslouchejte na lokální adrese a definovaném portu
listen_addresses = ['0.0.0.0:53']

# Použijte IPv4 i IPv6
ipv4_servers = true
ipv6_servers = false

# Povolte logování
log_level = 2
log_file = '/config/dnscrypt-proxy.log'

# Timeout a maximální počet pokusů
timeout = 5000
keepalive = 30

# Seznam serverů k použití, v preferovaném pořadí
server_names = ['cloudflare', 'google', 'quad9-recommended', 'adguard-dns']

# Pokud jeden server selže, zkusí automaticky další
fallback_resolvers = ['1.1.1.1:53', '8.8.8.8:53', '9.9.9.9:53']

# Cache konfigurace
cache = true
cache_size = 4096
cache_min_ttl = 2400
cache_max_ttl = 86400
cache_neg_min_ttl = 60
cache_neg_max_ttl = 600

# Anonymizované DNS (pro zvýšení soukromí)
routes = [
    { server_name='cloudflare', via=['anon-cs-usca', 'anon-cs-nl'] },
    { server_name='google', via=['anon-cs-de', 'anon-cs-fr'] }
]

# Blokování malwaru (volitelné)
```

```
block_malware = true
block_ads = false
block_undelegated = true

# Bezpečnostní nastavení
require_dnssec = true
require_nolog = true
require_nofilter = true

# Ukládání certifikátů a klíčů
cert_refresh_delay = 240
dnscrypt_ephemeral_keys = true
tls_disable_session_tickets = true

# Ověřování odezvy
reject_ttl_below = 10
```

P?íprava pro druhý server (DNS2)

Připojte se k druhému Raspberry Pi a opakujte podobný postup:

```
sudo nano docker-compose.yml
```

Vložte podobný obsah jako u prvního serveru, jen změňte název kontejneru:

```
version: '3'
services:
  dnscrypt-proxy:
    image: klutchell/dnscrypt-proxy:latest
    container_name: dns2_dnscrypt
    restart: always
    network_mode: host
    volumes:
      - ./config:/config
    environment:
      - TZ=Europe/Prague
    cap_add:
      - NET_ADMIN
```

Vytvořte konfigurační soubor pro druhý server:

```
sudo nano config/dnscrypt-proxy.toml
```

Vložte podobnou konfiguraci, ale změňte pořadí serverů pro lepší redundanci:

```
# Naslouchejte na lokální adrese a definovaném portu
listen_addresses = ['0.0.0.0:53']

# Použijte IPv4 i IPv6
ipv4_servers = true
ipv6_servers = false

# Povolte logování
log_level = 2
log_file = '/config/dnscrypt-proxy.log'

# Timeout a maximální počet pokusů
timeout = 5000
keepalive = 30

# Seznam serverů k použití, v preferovaném pořadí
# Záměrně změněné pořadí oproti DNS1 pro lepší redundanci
server_names = ['quad9-recommended', 'adguard-dns', 'cloudflare', 'google']

# Pokud jeden server selže, zkusí automaticky další
fallback_resolvers = ['9.9.9.9:53', '1.1.1.1:53', '8.8.8.8:53']

# Cache konfigurace
cache = true
cache_size = 4096
cache_min_ttl = 2400
cache_max_ttl = 86400
cache_neg_min_ttl = 60
cache_neg_max_ttl = 600

# Anonymizované DNS (pro zvýšení soukromí)
routes = [
    { server_name='quad9-recommended', via=['anon-cs-de', 'anon-cs-fr'] },
    { server_name='cloudflare', via=['anon-cs-nl', 'anon-cs-usca'] }
]
```

```
# Blokování malwaru (volitelné)
block_malware = true
block_ads = false
block_undelegated = true

# Bezpečnostní nastavení
require_dnssec = true
require_nolog = true
require_nofilter = true

# Ukládání certifikátů a klíčů
cert_refresh_delay = 240
dnscrypt_ephemeral_keys = true
tls_disable_session_tickets = true

# Ověřování odezvy
reject_ttl_below = 10
```

Spušt?ní DNS server?

Na obou Raspberry Pi spusťte kontejnery:

```
# Na prvním RPi
cd /opt/docker/dns1
sudo docker-compose up -d

# Na druhém RPi
cd /opt/docker/dns2
sudo docker-compose up -d
```

Automatické aktualizace

Vytvořte skripty pro automatickou aktualizaci každých 14 dní. Na každém Raspberry Pi vytvořte skript:

```
# Na prvním RPi
sudo nano /opt/update_dns1.sh
```

Vložte následující obsah:

```
#!/bin/bash
# Log file
LOG_FILE="/opt/dns_update.log"

echo "$(date) - Začátek aktualizace DNS1" >> $LOG_FILE

# Aktualizace systému
echo "Aktualizace systému..." >> $LOG_FILE
sudo apt update && sudo apt upgrade -y >> $LOG_FILE 2>&1

# Přejděte do adresáře DNS serveru
cd /opt/docker/dns1

# Aktualizace Docker image
echo "Aktualizace Docker image..." >> $LOG_FILE
sudo docker-compose pull >> $LOG_FILE 2>&1

# Restart kontejneru
echo "Restart služby..." >> $LOG_FILE
sudo docker-compose down >> $LOG_FILE 2>&1
sudo docker-compose up -d >> $LOG_FILE 2>&1

echo "$(date) - Aktualizace DNS1 dokončena" >> $LOG_FILE
```

Nastavte spouštěcí práva:

```
sudo chmod +x /opt/update_dns1.sh
```

Podobně na druhém Raspberry Pi:

```
sudo nano /opt/update_dns2.sh
```

S podobným obsahem (upravte cesty pro DNS2):

```
#!/bin/bash
# Log file
LOG_FILE="/opt/dns_update.log"

echo "$(date) - Začátek aktualizace DNS2" >> $LOG_FILE
```

```
# Aktualizace systému
echo "Aktualizace systému..." >> $LOG_FILE
sudo apt update && sudo apt upgrade -y >> $LOG_FILE 2>&1

# Přejděte do adresáře DNS serveru
cd /opt/docker/dns2

# Aktualizace Docker image
echo "Aktualizace Docker image..." >> $LOG_FILE
sudo docker-compose pull >> $LOG_FILE 2>&1

# Restart kontejneru
echo "Restart služby..." >> $LOG_FILE
sudo docker-compose down >> $LOG_FILE 2>&1
sudo docker-compose up -d >> $LOG_FILE 2>&1

echo "$(date) - Aktualizace DNS2 dokončena" >> $LOG_FILE
```

Nastavte spouštěcí práva:

```
sudo chmod +x /opt/update_dns2.sh
```

Nakonec přidejte cronjob pro automatické spouštění:

```
# Na prvním RPi
sudo crontab -e
```

Přidejte řádek (spouští se každých 14 dní ve 2:00 ráno):

```
0 2 1,15 * * /opt/update_dns1.sh
```

Na druhém RPi (spouští se každých 14 dní ve 3:00 ráno - o den později než první server):

```
0 3 2,16 * * /opt/update_dns2.sh
```

Ov??ení funk?nosti

Pro ověření, že DNS servery fungují správně, můžete použít:

```
# Test pomocí dig
dig @192.168.54.2 example.com
dig @192.168.54.3 example.com

# Kontrola logů
sudo docker logs dns1_dnscrypt
sudo docker logs dns2_dnscrypt
```

Další zabezpe?ení DNS server?

Kromě použití DoH doporučuji tyto další bezpečnostní opatření:

1. **DNSSEC** - Zajišťuje autenticitu DNS odpovědí (už zahrnuto v konfiguraci)
2. **DNS Query Name Minimization** - Zvyšuje soukromí tím, že posílá jen potřebné části doménového jména
3. **IP Tables pravidla** - Pokud chcete omezit, kdo může používat vaše DNS servery:

```
sudo iptables -A INPUT -p udp --dport 53 -s 192.168.54.0/24 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 53 -s 192.168.54.0/24 -j ACCEPT
sudo iptables -A INPUT -p udp --dport 53 -j DROP
sudo iptables -A INPUT -p tcp --dport 53 -j DROP
```

Monitoring - Můžete přidat monitoring pomocí Prometheus a Grafana:

```
# Přidejte do docker-compose.yml
  prometheus-exporter:
    image: prom/node-exporter
    container_name: dns_prometheus
    restart: always
    ports:
      - "9100:9100"
```

Zálohování mezi DNS servery

Když máte nastavené dva DNS servery, zálohování funguje přirozeně na úrovni klienta. Když klientská zařízení nakonfigurujete tak, aby používala oba servery (192.168.54.2 a 192.168.54.3), budou automaticky využívat druhý server, pokud první nereaguje.

Navíc, každý ze serverů má vlastní záložní mechanismus - pokud primární DoH poskytovatel (např. Cloudflare na prvním serveru) není dostupný, přepne se na další v seznamu (Google, Quad9 atd.).

Záv?rem

Tento návod ti umožní vytvořit dva nezávislé, redundantní DNS servery s DoH na Raspberry Pi 5 s použitím adresáře `/opt/` pro umístění všech souborů. Díky kombinaci různých poskytovatelů DoH a jejich různému pořadí na každém serveru získáš maximální dostupnost a bezpečnost.

DoH významně zlepšuje soukromí a bezpečnost tím, že šifruje DNS provoz, zatímco dnscrypt-proxy nabízí pokročilé funkce jako redundance, blokování malwaru a DNSSEC. Konfigurace je nastavena tak, aby servery byly pravidelně aktualizovány a mohly bezpečně sloužit tvé interní síti.

Zm?na hesla do Pihole

```
docker exec -it pihole pihole setpassword 'noveheslo'
```

Testování DNS serveru, kompletní p?íkazy

1. Test lokálního resolveru (Pi-hole)

```
# Otestujte základní dotaz přes Pi-hole
dig @127.0.0.1 example.com
# Test rekurze + DNSSEC
dig @127.0.0.1 example.com +dnssec +multiline
# Test blokování reklam (mělo vrátit 0.0.0.0)
dig @127.0.0.1 doubleclick.net
```

2. Test Unbound (lokální cache)

```
# Dotaz přímo na Unbound (port 5053)
dig @127.0.0.1 -p 5053 example.com
# Zkontrolujte cache (TTL by se měl snižovat)
dig @127.0.0.1 -p 5053 example.com +short
```

3. Test DNSCrypt-proxy (šifrovaný DNS)

```
# Dotaz přímo na DNSCrypt (port 5300)
dig @127.0.0.1 -p 5300 example.com
# Test DoH (DNS-over-HTTPS)
curl -H 'accept: application/dns-json' 'http://127.0.0.1:5300/dns-
query?name=example.com&type=A'
```

4. Test z externích za?ízení

```
# Z Synology (IP 192.168.54.254)
nslookup example.com 192.168.54.X  # (IP Pi-hole)
# Z jiného zařízení v síti
dig @192.168.54.X example.com  # (IP Pi-hole)
```

5. Pokro?ilé testy

```
# Test rychlosti (měří odezvu)
time dig @127.0.0.1 example.com
# Test DNSSEC (mělo být "ad" flag)
dig @127.0.0.1 sigok.verteiltesysteme.net +dnssec
# Test netestovaného doménového jména (mělo selhat)
dig @127.0.0.1 test.dnssec-failed.org
```

Ideální výsledek

- Všechny dotazy by měly vracet **rychlé odpovědi** (pod 50 ms)
- DNSSEC dotazy by měly mít **"ad" flag** (ověřené)
- Blokované domény by měly vracet **0.0.0.0**
- V logách by neměly být **chyby "ignoring query"**

DNS server, Pi-hole, DoH + záložní DNS2 server na Raspberry PI5

P?íprava prost?edí

Nejprve vytvoříme potřebnou adresářovou strukturu na obou serverech:

```
# Vytvoření adresářové struktury - spusť postupně každý příkaz
sudo mkdir -p /opt/dns-server
sudo mkdir -p /opt/dns-server/pihole
sudo mkdir -p /opt/dns-server/pihole/etc-pihole
sudo mkdir -p /opt/dns-server/pihole/etc-dnsmasq.d
sudo mkdir -p /opt/dns-server/unbound
sudo mkdir -p /opt/dns-server/unbound/conf
sudo mkdir -p /opt/dns-server/cloudflared
sudo mkdir -p /opt/dns-server/backups

# Nastavení správných oprávnění
sudo chown -R 1000:1000 /opt/dns-server/pihole
```

Docker Compose pro DNS1 + DNS2 server

Vytvořme soubor `/opt/dns-server/docker-compose.yml` pro každý server. Začneme s první instancí (dns1.zelina.eu - 192.168.54.2):

```
# Vytvoření souboru docker-compose.yml
sudo nano /opt/dns-server/docker-compose.yml
```

Ale ja nepoužívám docker-compose, takže vytvořím stack server-DNS1 (server-DNS2) a vložím do web editoru:

```
version: '3'

services:
  # Unbound pro rekurzivní DNS a keš
  unbound:
```

```
    container_name: unbound
    image: klutchell/unbound:latest
    restart: unless-stopped
    hostname: unbound
    volumes:
      - /opt/dns-server/unbound/conf:/opt/unbound/etc/unbound/
    networks:
      dns_net:
        ipv4_address: 172.20.0.2

  # Cloudflared pro DNS-over-HTTPS
  cloudflared:
    container_name: cloudflared
    image: cloudflare/cloudflared:latest
    restart: unless-stopped
    command: proxy-dns --address 0.0.0.0 --port 5053 --upstream https://1.1.1.1/dns-query --
upstream https://1.0.0.1/dns-query
    networks:
      dns_net:
        ipv4_address: 172.20.0.3

  # Pi-hole jako DNS filtrovací nástroj
  pihole:
    container_name: pihole
    image: pihole/pihole:latest
    restart: unless-stopped
    hostname: 192.168.54.2    # Změnit na 19*2.168.54.3 - dns2.zelina.eu pro druhý server
    ports:
      - "53:53/tcp"
      - "53:53/udp"
      - "80:80/tcp"
    environment:
      TZ: 'Europe/Prague'
      WEBPASSWORD: 'BezpecneHeslo2024'    # Změnit na silné heslo
      DNS1: '172.20.0.2#53'    # Unbound
      DNS2: '172.20.0.3#5053'  # Cloudflared DoH
      ServerIP: '192.168.54.2' # Změnit na 192.168.54.3 pro druhý server
      HOSTNAME: 'dns1.zelina.eu' # Změnit na dns2.zelina.eu pro druhý server
      DNSMASQ_LISTENING: 'all'
      CACHE_SIZE: 10000
```

```
    volumes:
      - '/opt/dns-server/pihole/etc-pihole:/etc/pihole'
      - '/opt/dns-server/pihole/etc-dnsmasq.d:/etc/dnsmasq.d'
    depends_on:
      - unbound
      - cloudflared
    cap_add:
      - NET_ADMIN
    networks:
      dns_net:
        ipv4_address: 172.20.0.4

networks:
  dns_net:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/24
```

Uložíš pomocí klávesové zkratky `Ctrl+X`, poté `Y` a potvrdíš pomocí `Enter`.

Pro druhý server (192.168.54.3) upravíš následující hodnoty:

- `hostname: dns2.zelina.eu`
- `HOSTNAME: 'dns2.zelina.eu'`
- `ServerIP: '192.168.54.3'`

Konfigurace Unbound pro kešování:

Potřebujeme vytvořit konfigurační soubor pro Unbound. Vytvoříme `/opt/dns-server/unbound/conf/unbound.conf`:

```
# Vytvoření konfiguračního souboru pro Unbound
sudo nano /opt/dns-server/unbound/conf/unbound.conf
```

Zkopíruj do souboru následující obsah:

```
server:
    # Obecná nastavení
    verbosity: 1
    num-threads: 4
    interface: 0.0.0.0
```

```
port: 53
do-ip4: yes
do-ip6: no
do-udp: yes
do-tcp: yes

# Bezpečnost
hide-identity: yes
hide-version: yes
harden-glue: yes
harden-dnssec-stripped: yes
use-caps-for-id: yes

# Nastavení keše
cache-min-ttl: 300
cache-max-ttl: 86400
prefetch: yes
prefetch-key: yes
msg-cache-size: 128m
rrset-cache-size: 256m
neg-cache-size: 32m

# Výkon
so-rcvbuf: 4m
so-sndbuf: 4m
edns-buffer-size: 1232
max-udp-size: 1232

# Kořenové servery
root-hints: "/opt/unbound/etc/unbound/root.hints"

# Přístup z lokální sítě
access-control: 0.0.0.0/0 allow

# DNSSEC validace
auto-trust-anchor-file: "/opt/unbound/etc/unbound/root.key"

# Optimalizace pro slabý hardware (RPi)
outgoing-range: 512
num-queries-per-thread: 512
```

```
    # Vylepšení soukromí
    qname-minimisation: yes

    # Zápis statistik
    statistics-interval: 600
    extended-statistics: yes
    statistics-cumulative: yes

remote-control:
    control-enable: no
```

Aktualizace root.hints

Pro správnou funkci Unbound potřebujeme stáhnout nejnovější root.hints soubor:

```
wget -O /opt/dns-server/unbound/conf/root.hints https://www.internic.net/domain/named.root
```

Cloudflared - DNS over HTTPS

Pro Cloudflared nepotřebujeme speciální konfigurační soubor, jelikož nastavení předáváme přímo jako parametry v docker-compose.yml.

Skript pro kontrolu a restart služeb

Vytvoření kontrolního skriptu pro DNS server:

```
# Vytvoření skriptu pro kontrolu DNS
sudo nano /opt/dns-server/check-dns.sh
```

Vytvoříme skript, který bude kontrolovat funkčnost DNS serveru a případně jej restartovat:

```
#!/bin/bash
# Script pro kontrolu DNS serveru
# Umístit do /opt/dns-server/check-dns.sh

LOG_FILE="/opt/dns-server/dns-check.log"
HOSTNAME=$(hostname)
TIMESTAMP=$(date "+%Y-%m-%d %H:%M:%S")

# Funkce pro záznam do logu
log_message() {
```

```
    echo "${TIMESTAMP} - ${1}" >> ${LOG_FILE}
    echo "${1}"
}

# Kontrola, zda Pi-hole běží
pihole_check() {
    if docker ps | grep -q pihole; then
        log_message "Pi-hole běží správně."
    else
        log_message "Pi-hole neběží! Pokus o restart..."
        cd /opt/dns-server && docker-compose restart pihole
        sleep 10
        if docker ps | grep -q pihole; then
            log_message "Pi-hole byl úspěšně restartován."
        else
            log_message "Restart Pi-hole selhal! Spouštím kompletní restart DNS stacku..."
            cd /opt/dns-server && docker-compose down && docker-compose up -d
        fi
    fi
}

# Kontrola funkčnosti DNS
dns_check() {
    if dig @localhost google.com +short > /dev/null; then
        log_message "DNS resolving funguje správně."
    else
        log_message "DNS resolving nefunguje! Pokus o restart DNS stacku..."
        cd /opt/dns-server && docker-compose down && docker-compose up -d
    fi
}

# Hlavní kontrolní rutina
log_message "======= Začátek kontroly DNS serveru ($HOSTNAME) ======="
pihole_check
dns_check
log_message "======= Konec kontroly DNS serveru ======="
```

Nastavíme správná oprávnění a vytvoříme plánovanou úlohu cron:

```
sudo chmod +x /opt/dns-server/check-dns.sh
# Přidání do cronu pro spouštění každou hodinu
(crontab -l ; echo "0 * * * * /opt/dns-server/check-dns.sh") | crontab -
```

Záloha konfigurace

Vytvoření zálohovacího skriptu

```
# Vytvoření skriptu pro zálohování
sudo nano /opt/dns-server/backup-dns.sh
```

Vytvoříme skript pro pravidelné zálohování konfigurace:

```
#!/bin/bash
# Script pro zálohování DNS serveru
# Umístit do /opt/dns-server/backup-dns.sh

BACKUP_DIR="/opt/dns-server/backups"
TIMESTAMP=$(date "+%Y%m%d-%H%M%S")
HOSTNAME=$(hostname)
BACKUP_FILE="${BACKUP_DIR}/dns-backup-${HOSTNAME}-${TIMESTAMP}.tar.gz"

# Vytvoření adresáře pro zálohy, pokud neexistuje
mkdir -p ${BACKUP_DIR}

# Zápis do logu
echo "Zahajuji zálohu DNS konfigurace - ${TIMESTAMP}" >> ${BACKUP_DIR}/backup.log

# Vytvoření zálohy
tar -czf ${BACKUP_FILE} \
    /opt/dns-server/docker-compose.yml \
    /opt/dns-server/pihole \
    /opt/dns-server/unbound/conf \

# Výsledek
if [ $? -eq 0 ]; then
    echo "Záloha úspěšně vytvořena: ${BACKUP_FILE}" >> ${BACKUP_DIR}/backup.log

    # Odstranění starých záloh (starších než 30 dnů)
    find ${BACKUP_DIR} -name "dns-backup-*.tar.gz" -mtime +30 -delete
```

```
else
    echo "CHYBA: Vytvoření zálohy selhalo!" >> ${BACKUP_DIR}/backup.log
fi
```

Nastavíme oprávnění a přidáme do cronu pro týdenní zálohování:

```
# Nastavení spustitelných práv pro zálohovací skript
sudo chmod +x /opt/dns-server/backup-dns.sh

# Přidání do crontab pro spouštění každou neděli ve 2:00
(crontab -l 2>/dev/null; echo "0 2 * * 0 /opt/dns-server/backup-dns.sh") | crontab -
```

Vytvo?ení testovacího skriptu

```
# Vytvoření testovacího skriptu
sudo nano /opt/dns-server/test-dns.sh
```

Zkopíruj do souboru následující obsah:

```
#!/bin/bash
# Skript pro testování DNS serveru

# Barvy pro výstup
GREEN='\033[0;32m'
RED='\033[0;31m'
NC='\033[0m' # No Color

# Testovací domény
TEST_DOMAINS=("google.com" "facebook.com" "youtube.com" "github.com" "zelina.eu")
# Test blokované domény (známé reklamní domény)
BLOCKED_DOMAINS=("doubleclick.net" "googleadservices.com" "advertising.com")

echo "==== Testování DNS serveru ===="
echo "Testovací server: $(hostname)"
echo ""

# Test základních DNS dotazů
echo "1. Test základních DNS dotazů"
for domain in "${TEST_DOMAINS[@]}"; do
    echo -n "Dotaz na $domain: "
    if dig @localhost $domain +short > /dev/null; then
```

```
        echo -e "${GREEN}OK${NC}"
    else
        echo -e "${RED}SELHALO${NC}"
    fi
done
echo ""

# Test blokování reklam
echo "2. Test blokování reklam"
for domain in "${BLOCKED_DOMAINS[@]}"; do
    echo -n "Kontrola blokování $domain: "
    RESULT=$(dig @localhost $domain +short)
    if [[ "$RESULT" == *"0.0.0.0"* ]] || [[ -z "$RESULT" ]]; then
        echo -e "${GREEN}BLOKOVÁNO${NC}"
    else
        echo -e "${RED}NENÍ BLOKOVÁNO${NC}"
    fi
done
echo ""

# Test DNS over HTTPS
echo "3. Test DNS over HTTPS"
echo -n "Kontrola funkčnosti Cloudflared: "
if docker logs cloudflared 2>&1 | grep -q "Starting DNS over HTTPS proxy"; then
    echo -e "${GREEN}OK${NC}"
else
    echo -e "${RED}SELHALO${NC}"
fi
echo ""

# Test výkonu DNS
echo "4. Test výkonu DNS"
echo "Měření rychlosti DNS dotazů:"
time dig @localhost google.com > /dev/null
echo ""

# Test kešování
echo "5. Test DNS kešování"
echo "První dotaz (nekešovaný):"
time dig @localhost wikipedia.org > /dev/null
```

```
echo "Druhý dotaz (měl by být kešovaný):"
time dig @localhost wikipedia.org > /dev/null
echo ""

# Test zátěže
echo "6. Test zátěže (10 paralelních dotazů)"
for i in {1..10}; do
    dig @localhost example.com > /dev/null &
done
wait
echo -e "${GREEN}Test zátěže dokončen${NC}"
echo ""

echo "==== Test DNS serveru dokončen ===="
```

Nastavení práv pro testovací skript

```
# Nastavení spustitelných práv pro testovací skript
sudo chmod +x /opt/dns-server/test-dns.sh
```

Vytvo?ení skriptu pro synchronizaci mezi servery (pouze na primárním serveru)

```
# Vytvoření synchronizačního skriptu (pouze na serveru dns1.zelina.eu)
sudo nano /opt/dns-server/sync-dns.sh
```

Zkopíruj do souboru následující obsah:

```
#!/bin/bash
# Skript pro synchronizaci mezi primárním a sekundárním DNS serverem

PRIMARY_SERVER="dns1.zelina.eu"
SECONDARY_SERVER="dns2.zelina.eu"
SECONDARY_IP="192.168.54.3"
SSH_KEY="/root/.ssh/id_rsa"
SYNC_LOG="/opt/dns-server/sync.log"

# Kontrola, zda běží na primárním serveru
if [ "$(hostname)" != "$PRIMARY_SERVER" ]; then
```

```
    echo "Tento skript musí běžet na primárním serveru ($PRIMARY_SERVER)!"
    exit 1
fi

timestamp() {
    date "+%Y-%m-%d %H:%M:%S"
}

log() {
    echo "$(timestamp) - $1" >> "$SYNC_LOG"
    echo "$1"
}

log "Začátek synchronizace DNS serverů"

# Export Pi-hole nastavení a blokačních listů
log "Export nastavení Pi-hole..."
docker exec pihole pihole -a teleporter > /opt/dns-server/pihole-teleporter.tar.gz

# Synchronizace souborů na sekundární server
log "Synchronizace souborů na sekundární server..."
rsync -avz -e "ssh -i $SSH_KEY" \
    --exclude="backups" \
    --exclude="*.log" \
    /opt/dns-server/pihole-teleporter.tar.gz \
    /opt/dns-server/unbound/conf/ \
    root@$SECONDARY_IP:/opt/dns-server/

# Import nastavení na sekundárním serveru
log "Import nastavení na sekundárním serveru..."
ssh -i $SSH_KEY root@$SECONDARY_IP "docker exec -i pihole pihole -a -t /opt/dns-server/pihole-
teleporter.tar.gz"

# Restart služeb na sekundárním serveru
log "Restart služeb na sekundárním serveru..."
ssh -i $SSH_KEY root@$SECONDARY_IP "cd /opt/dns-server && docker-compose restart"

log "Synchronizace DNS serverů dokončena"
```

Nastavení práv pro synchronizační skript a přidání do cronu (pouze na primárním serveru)

```
# Nastavení spustitelných práv pro synchronizační skript
sudo chmod +x /opt/dns-server/sync-dns.sh

# Přidání do crontab pro spouštění každý den ve 3:00
(crontab -l 2>/dev/null; echo "0 3 * * * /opt/dns-server/sync-dns.sh") | crontab -
```

Vytvoření monitorovacího skriptu

```
# Vytvoření monitorovacího skriptu
sudo nano /opt/dns-server/monitor-dns.sh
```

Zkopíruj do souboru následující obsah:

```
#!/bin/bash
# Skript pro monitorování DNS serveru

MONITOR_LOG="/opt/dns-server/monitor.log"
HOSTNAME=$(hostname)
TIMESTAMP=$(date "+%Y-%m-%d %H:%M:%S")

# Záznam do logu
echo "===== DNS Monitor Report - $TIMESTAMP =====" >> $MONITOR_LOG

# Kontrola stavu kontejnerů
echo "Docker kontejnery:" >> $MONITOR_LOG
docker ps -a | grep -E 'pihole|unbound|cloudflared' >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Pi-hole statistiky
echo "Pi-hole statistiky:" >> $MONITOR_LOG
docker exec pihole pihole -c >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Využití systémových prostředků
echo "Vytížení CPU:" >> $MONITOR_LOG
top -bn1 | head -n 5 >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

echo "Využití paměti:" >> $MONITOR_LOG
free -h >> $MONITOR_LOG
```

```
echo "" >> $MONITOR_LOG

echo "Využití disku:" >> $MONITOR_LOG
df -h /opt >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Kontrola DNS dotazů
echo "Počet DNS dotazů za poslední hodinu:" >> $MONITOR_LOG
docker exec pihole sqlite3 /etc/pihole/pihole-FTL.db "SELECT COUNT(*) FROM queries WHERE
timestamp >= strftime('%s', 'now', '-1 hour');" >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Top 10 blokovaných domén
echo "Top 10 blokovaných domén:" >> $MONITOR_LOG
docker exec pihole sqlite3 /etc/pihole/pihole-FTL.db "SELECT domain, COUNT(*) FROM queries
WHERE status IN (1,4,5,6,7,8,9,10,11) GROUP BY domain ORDER BY COUNT(*) DESC LIMIT 10;" >>
$MONITOR_LOG
echo "" >> $MONITOR_LOG

# Top 10 povolených domén
echo "Top 10 povolených domén:" >> $MONITOR_LOG
docker exec pihole sqlite3 /etc/pihole/pihole-FTL.db "SELECT domain, COUNT(*) FROM queries
WHERE status = 2 GROUP BY domain ORDER BY COUNT(*) DESC LIMIT 10;" >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

echo "===== Konec DNS Monitor Report =====" >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Omezení velikosti logu na posledních 100 kB
if [ -f "$MONITOR_LOG" ] && [ $(stat -c%s "$MONITOR_LOG") -gt 102400 ]; then
    tail -c 102400 "$MONITOR_LOG" > "$MONITOR_LOG.tmp" && mv "$MONITOR_LOG.tmp" "$MONITOR_LOG"
fi
```

Nastavení práv pro monitorovací skript a přidání do cronu

```
# Nastavení spustitelných práv pro monitorovací skript
sudo chmod +x /opt/dns-server/monitor-dns.sh

# Přidání do crontab pro spouštění každou hodinu v půl
(crontab -l 2>/dev/null; echo "30 * * * * /opt/dns-server/monitor-dns.sh") | crontab -
```

Nastavení SSH klíčů pro komunikaci mezi servery (pouze na primárním serveru)

```
# Generování SSH klíčů na primárním serveru
ssh-keygen -t rsa -b 4096 -f /root/.ssh/id_rsa -N ""

# Kopírování veřejného klíče na sekundární server
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.54.3
```

Kontrola běžících kontejnerů

```
# Kontrola, zda všechny kontejnery běží
docker ps

# Kontrola logů jednotlivých kontejnerů
docker logs pihole
docker logs unbound
docker logs cloudflared
```

Otestování DNS serveru po spuštění

```
# Spuštění testovacího skriptu
/opt/dns-server/test-dns.sh
```

Pro zajišt?ní stability:

```
# Vytvořit skript pro pravidelnou kontrolu stavu
cat > /opt/dns-server/check-services.sh << 'EOF'
#!/bin/bash
echo "Kontrola služeb DNS serveru $(date)"
docker ps | grep -q pihole || (echo "Pi-hole neběží!" && docker start pihole)
docker ps | grep -q unbound || (echo "Unbound neběží!" && docker start unbound)
docker ps | grep -q cloudflared || (echo "Cloudflared neběží!" && docker start cloudflared)
EOF

chmod +x /opt/dns-server/check-services.sh

# Přidat do cronu
(crontab -l 2>/dev/null; echo "*/15 * * * * /opt/dns-server/check-services.sh >> /var/log/dns-
```

```
check.log 2>&1") | crontab -
```

Skript pro zajištění stability, který jsem navrhl, je jednoduchý monitorovací nástroj, který pomáhá udržet vaše DNS služby v provozu. Pojďme si ho rozebrat:

1. **Co tento skript konkrétně dělá:**
   - Kontroluje, zda všechny tři klíčové kontejnery (pihole, unbound a cloudflared) běží
   - Pokud některý z kontejnerů neběží, automaticky se ho pokusí znovu spustit
   - Zaznamenává informace o kontrolách do logovacího souboru
2. **Princip fungování:**
   - Příkaz `docker ps | grep -q pihole` kontroluje, zda v seznamu běžících kontejnerů je kontejner "pihole"
   - Pokud není nalezen (vrací chybový kód), spustí se příkaz za `||` - tedy vypíše upozornění a spustí kontejner pomocí `docker start pihole`
   - Stejná kontrola se provádí pro unbound a cloudflared
3. **Nastavení automatického spouštění:**
   - Vytváří se záznam v crontabu, který zajistí spouštění tohoto skriptu každých 15 minut
   - Výstup skriptu je přesměrován do souboru `/var/log/dns-check.log` pro případnou analýzu

Tento přístup je užitečný pro zajištění vysoké dostupnosti DNS serveru, protože automaticky obnoví služby v případě neočekávaného selhání některého z kontejnerů. Jedná se o jednoduchý způsob "samohojení" systému, který může zachytit a vyřešit mnoho běžných problémů bez nutnosti manuálního zásahu.

Konfigurace pro druhý server (dns2.zelina.eu - 192.168.54.3)

Pro druhý server postupujeme stejně, jen upravíme následující hodnoty v docker-compose.yml:

- `hostname: dns2.zelina.eu` (pro kontejner pihole)
- `HOSTNAME: 'dns2.zelina.eu'` (v environment pro pihole)
- `ServerIP: '192.168.54.3'` (v environment pro pihole)

Pro testování DNS je pot?eba nainstalovat balí?ek dnsutils.

```
apt-get update && apt-get install -y dnsutils
```

Testování funkcí DNS serveru

Pro testování funkčnosti vytvořme jednoduchý testovací skript:

```
#!/bin/bash
# Skript pro testování DNS serveru
# Umístit do /opt/dns-server/test-dns.sh

# Barvy pro výstup
GREEN='\033[0;32m'
RED='\033[0;31m'
NC='\033[0m' # No Color

# Testovací domény
TEST_DOMAINS=("google.com" "facebook.com" "youtube.com" "github.com" "zelina.eu")
# Test blokované domény (známé reklamní domény)
BLOCKED_DOMAINS=("doubleclick.net" "googleadservices.com" "advertising.com")

echo "==== Testování DNS serveru ===="
echo "Testovací server: $(hostname)"
echo ""

# Test základních DNS dotazů
echo "1. Test základních DNS dotazů"
for domain in "${TEST_DOMAINS[@]}"; do
    echo -n "Dotaz na $domain: "
    if dig @localhost $domain +short > /dev/null; then
        echo -e "${GREEN}OK${NC}"
    else
        echo -e "${RED}SELHALO${NC}"
    fi
done
echo ""

# Test blokování reklam
echo "2. Test blokování reklam"
for domain in "${BLOCKED_DOMAINS[@]}"; do
    echo -n "Kontrola blokování $domain: "
    RESULT=$(dig @localhost $domain +short)
    if [[ "$RESULT" == *"0.0.0.0"* ]] || [[ -z "$RESULT" ]]; then
        echo -e "${GREEN}BLOKOVÁNO${NC}"
    else
        echo -e "${RED}NENÍ BLOKOVÁNO${NC}"
    fi
```

```
done
echo ""

# Test DNS over HTTPS
echo "3. Test DNS over HTTPS"
echo -n "Kontrola funkčnosti Cloudflared: "
if docker logs cloudflared 2>&1 | grep -q "Starting DNS over HTTPS proxy"; then
    echo -e "${GREEN}OK${NC}"
else
    echo -e "${RED}SELHALO${NC}"
fi
echo ""

# Test výkonu DNS
echo "4. Test výkonu DNS"
echo "Měření rychlosti DNS dotazů:"
time dig @localhost google.com > /dev/null
echo ""

# Test kešování
echo "5. Test DNS kešování"
echo "První dotaz (nekešovaný):"
time dig @localhost wikipedia.org > /dev/null
echo "Druhý dotaz (měl by být kešovaný):"
time dig @localhost wikipedia.org > /dev/null
echo ""

# Test zátěže
echo "6. Test zátěže (10 paralelních dotazů)"
for i in {1..10}; do
    dig @localhost example.com > /dev/null &
done
wait
echo -e "${GREEN}Test zátěže dokončen${NC}"
echo ""

echo "==== Test DNS serveru dokončen ===="
```

Nastavíme správná oprávnění:

```
sudo chmod +x /opt/dns-server/test-dns.sh
```

Dodate?né zabezpe?ení a vylepšení

Povolení DNSSEC v Pi-hole

Pro zapnutí DNSSEC v Pi-hole:

```
# Připojení do kontejneru Pi-hole
docker exec -it pihole bash

# Editace konfiguračního souboru
echo "dnssec-validate=yes" >> /etc/dnsmasq.d/01-pihole.conf

# Restart dnsmasq
pihole restartdns
```

P?idání bloka?ních list?

Můžeme přidat další blokační listy do Pi-hole pro lepší filtrování:

```
# Připojení do kontejneru Pi-hole
docker exec -it pihole bash

# Přidání blokačních listů (můžeš přidat přímo v rozhraní Pi-hole)
pihole -a adlist add https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts
pihole -a adlist add https://s3.amazonaws.com/lists.disconnect.me/simple_tracking.txt
pihole -a adlist add https://s3.amazonaws.com/lists.disconnect.me/simple_ad.txt

# Aktualizace blokačních listů
pihole -g
```

Nastavení replikace mezi servery

Pro zajištění synchronizace mezi oběma DNS servery vytvoříme skript pro replikaci nastavení z primárního na sekundární server:

```
#!/bin/bash
# Skript pro synchronizaci mezi primárním a sekundárním DNS serverem
# Umístit na primární server do /opt/dns-server/sync-dns.sh

PRIMARY_SERVER="dns1.zelina.eu"
SECONDARY_SERVER="dns2.zelina.eu"
SECONDARY_IP="192.168.54.3"
SSH_KEY="/root/.ssh/id_rsa"
SYNC_LOG="/opt/dns-server/sync.log"

# Kontrola, zda běží na primárním serveru
if [ "$(hostname)" != "$PRIMARY_SERVER" ]; then
    echo "Tento skript musí běžet na primárním serveru ($PRIMARY_SERVER)!"
    exit 1
fi

timestamp() {
    date "+%Y-%m-%d %H:%M:%S"
}

log() {
    echo "$(timestamp) - $1" >> "$SYNC_LOG"
    echo "$1"
}

log "Začátek synchronizace DNS serverů"

# Export Pi-hole nastavení a blokačních listů
log "Export nastavení Pi-hole..."
docker exec pihole pihole -a teleporter > /opt/dns-server/pihole-teleporter.tar.gz

# Synchronizace souborů na sekundární server
log "Synchronizace souborů na sekundární server..."
rsync -avz -e "ssh -i $SSH_KEY" \
    --exclude="backups" \
    --exclude="*.log" \
    /opt/dns-server/pihole-teleporter.tar.gz \
    /opt/dns-server/unbound/conf/ \
    root@$SECONDARY_IP:/opt/dns-server/
```

```
# Import nastavení na sekundárním serveru
log "Import nastavení na sekundárním serveru..."
ssh -i $SSH_KEY root@$SECONDARY_IP "docker exec -i pihole pihole -a -t /opt/dns-server/pihole-teleporter.tar.gz"

# Restart služeb na sekundárním serveru
log "Restart služeb na sekundárním serveru..."
ssh -i $SSH_KEY root@$SECONDARY_IP "cd /opt/dns-server && docker-compose restart"

log "Synchronizace DNS serverů dokončena"
```

Nastavíme oprávnění a přidáme do cronu pro denní synchronizaci:

```
sudo chmod +x /opt/dns-server/sync-dns.sh
# Přidání do cronu pro spouštění každý den ve 3:00
(crontab -l ; echo "0 3 * * * /opt/dns-server/sync-dns.sh") | crontab -
```

Pro automatickou synchronizaci musíme také nastavit SSH klíče mezi servery:

```
# Generování SSH klíčů na primárním serveru
ssh-keygen -t rsa -b 4096

# Kopírování klíče na sekundární server
ssh-copy-id root@192.168.54.3
```

Monitorování DNS serveru

Pro monitorování vytvoříme jednoduchý skript:

```
#!/bin/bash
# Skript pro monitorování DNS serveru
# Umístit do /opt/dns-server/monitor-dns.sh

MONITOR_LOG="/opt/dns-server/monitor.log"
HOSTNAME=$(hostname)
TIMESTAMP=$(date "+%Y-%m-%d %H:%M:%S")
```

```
# Záznam do logu
echo "===== DNS Monitor Report - $TIMESTAMP =====" >> $MONITOR_LOG

# Kontrola stavu kontejnerů
echo "Docker kontejnery:" >> $MONITOR_LOG
docker ps -a | grep -E 'pihole|unbound|cloudflared' >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Pi-hole statistiky
echo "Pi-hole statistiky:" >> $MONITOR_LOG
docker exec pihole pihole -c >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Využití systémových prostředků
echo "Vytížení CPU:" >> $MONITOR_LOG
top -bn1 | head -n 5 >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

echo "Využití paměti:" >> $MONITOR_LOG
free -h >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

echo "Využití disku:" >> $MONITOR_LOG
df -h /opt >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Kontrola DNS dotazů
echo "Počet DNS dotazů za poslední hodinu:" >> $MONITOR_LOG
docker exec pihole sqlite3 /etc/pihole/pihole-FTL.db "SELECT COUNT(*) FROM queries WHERE
timestamp >= strftime('%s', 'now', '-1 hour');" >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Top 10 blokovaných domén
echo "Top 10 blokovaných domén:" >> $MONITOR_LOG
docker exec pihole sqlite3 /etc/pihole/pihole-FTL.db "SELECT domain, COUNT(*) FROM queries
WHERE status IN (1,4,5,6,7,8,9,10,11) GROUP BY domain ORDER BY COUNT(*) DESC LIMIT 10;" >>
$MONITOR_LOG
echo "" >> $MONITOR_LOG
```

```
# Top 10 povolených domén
echo "Top 10 povolených domén:" >> $MONITOR_LOG
docker exec pihole sqlite3 /etc/pihole/pihole-FTL.db "SELECT domain, COUNT(*) FROM queries
WHERE status = 2 GROUP BY domain ORDER BY COUNT(*) DESC LIMIT 10;" >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

echo "===== Konec DNS Monitor Report =====" >> $MONITOR_LOG
echo "" >> $MONITOR_LOG

# Omezení velikosti logu na posledních 100 kB
if [ -f "$MONITOR_LOG" ] && [ $(stat -c%s "$MONITOR_LOG") -gt 102400 ]; then
    tail -c 102400 "$MONITOR_LOG" > "$MONITOR_LOG.tmp" && mv "$MONITOR_LOG.tmp" "$MONITOR_LOG"
fi
```

Nastavíme oprávnění a přidáme do cronu pro hodinové monitorování:

```
sudo chmod +x /opt/dns-server/monitor-dns.sh
# Přidání do cronu pro spouštění každou hodinu
(crontab -l ; echo "30 * * * * /opt/dns-server/monitor-dns.sh") | crontab -
```

P?ístup k webovému rozhraní Pi-hole

Pro přístup k Pi-hole rozhraní:

- Primární server: http://192.168.54.2/admin
- Sekundární server: http://192.168.54.3/admin

Výchozí heslo: `BezpecneHeslo2024` (doporučuji změnit na silnější heslo v konfiguračním souboru docker-compose.yml a následně restartovat kontejner)

Nastavení klient?

Pro používání tvého DNS serveru je třeba nastavit DNS servery na klientech nebo přímo v routeru:

- Primární DNS: 192.168.54.2
- Sekundární DNS: 192.168.54.3

Testování funk?nosti

Po nastavení a spuštění obou serverů můžeš otestovat funkčnost pomocí následujících příkazů:

1. Základní DNS dotaz:

```
dig @192.168.54.2 google.com
dig @192.168.54.3 google.com
```

1. Test blokování reklam:

```
dig @192.168.54.2 doubleclick.net
```

1. Test failover (vypnutí primárního serveru):

```
# Vypnout primární server nebo Docker na něm
# Testovat z klienta, který má nastaveny oba DNS servery
ping google.com
```

1. Komplexní test pomocí vytvořeného testovacího skriptu:

```
/opt/dns-server/test-dns.sh
```

Doporu?ení pro dlouhodobý provoz

1. **Pravidelné aktualizace**: Nastav automatické aktualizace kontejnerů pomocí Watchtower:

```
docker run -d \
  --name watchtower \
  --restart unless-stopped \
  -v /var/run/docker.sock:/var/run/docker.sock \
  containrrr/watchtower \
  --cleanup \
  --interval 86400 \
  pihole unbound cloudflared
```

2. **Monitorování výkonu**: Přidej do monitorovacího skriptu odeslání upozornění při překročení limitů systémových prostředků.
3. **Rotace logů**: Zajisti automatickou rotaci logů pro všechny vytvořené logy.

4. **Hardening serveru**: Zabezpeč samotný Raspberry Pi správným nastavením firewallu a SSH přístupu.
5. **Zálohování**:
   - Pravidelně zálohuj Pi-hole nastavení pomocí teleporter funkce
   - Zálohuj SD kartu nebo systémový disk pomocí nástroje jako je `dd` nebo `rpi-clone`

Tímto máš kompletní řešení pro dva zabezpečené DNS servery s Pi-hole, Unbound a Cloudflared pro DNS-over-HTTPS. Celé řešení běží v Dockeru, je redundantní, automaticky se zálohuje a monitoruje.