[1] Příloha dokumentace

Návod k NAS

Obsah

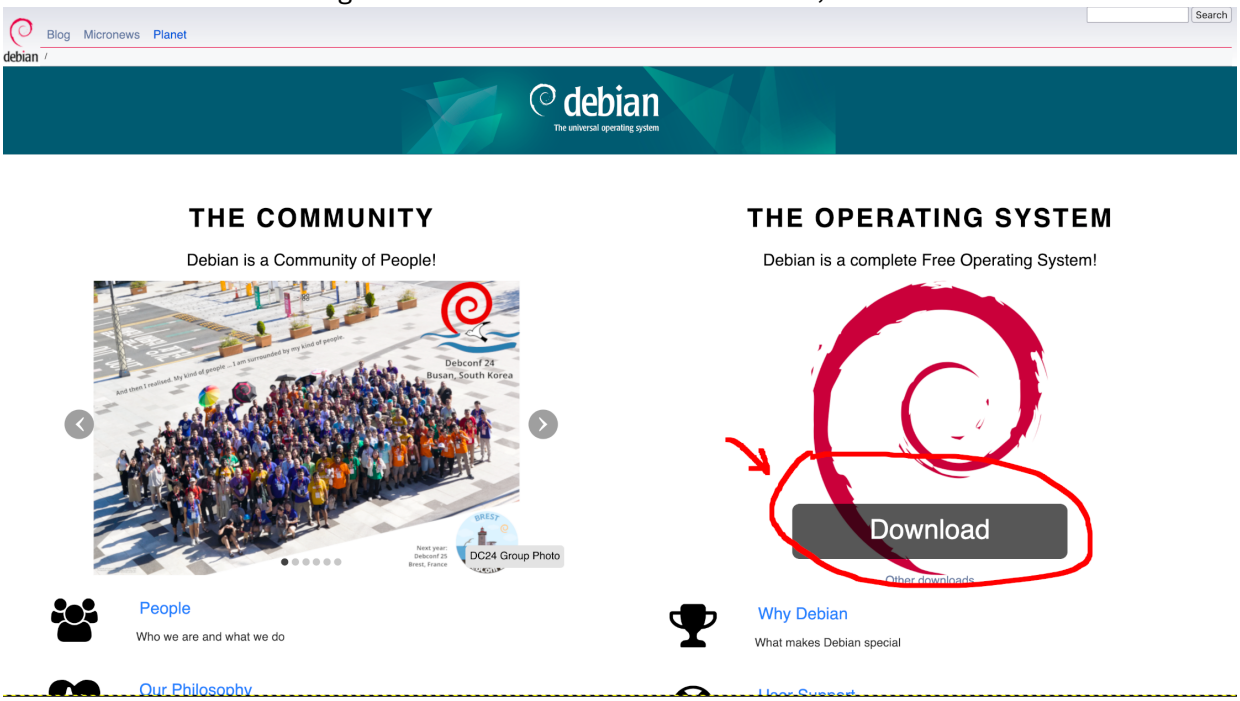
1 Hardware

Hardware mé NAS vypádá takto, ale je možné si vybrat jakýkoliv, klidně starší počítač.

Mini PC jsem si vybral čistě z praktických důvodů. NAS budu muset přinést do školy, takže tahat velký počítač by bylo nepříjemné. Výkonostně toto mini PC úplně stačí, na všechny vybrané služby takže, pokud nechcete využít hardware starší 12 let tak si myslím že nenarazíte na problém.

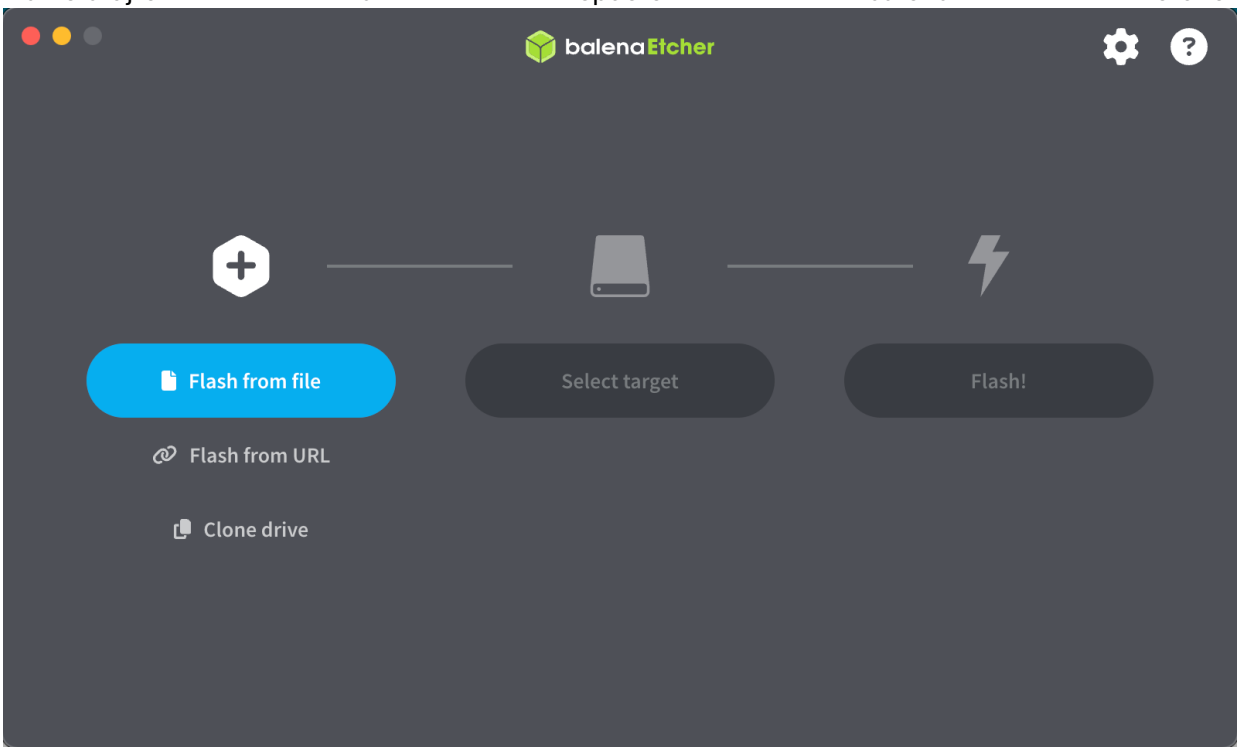
2 Instalace systému

Na stránce www.debian.org klieknete na velké tlačítko Download, to stáhne instalační iso Debianu

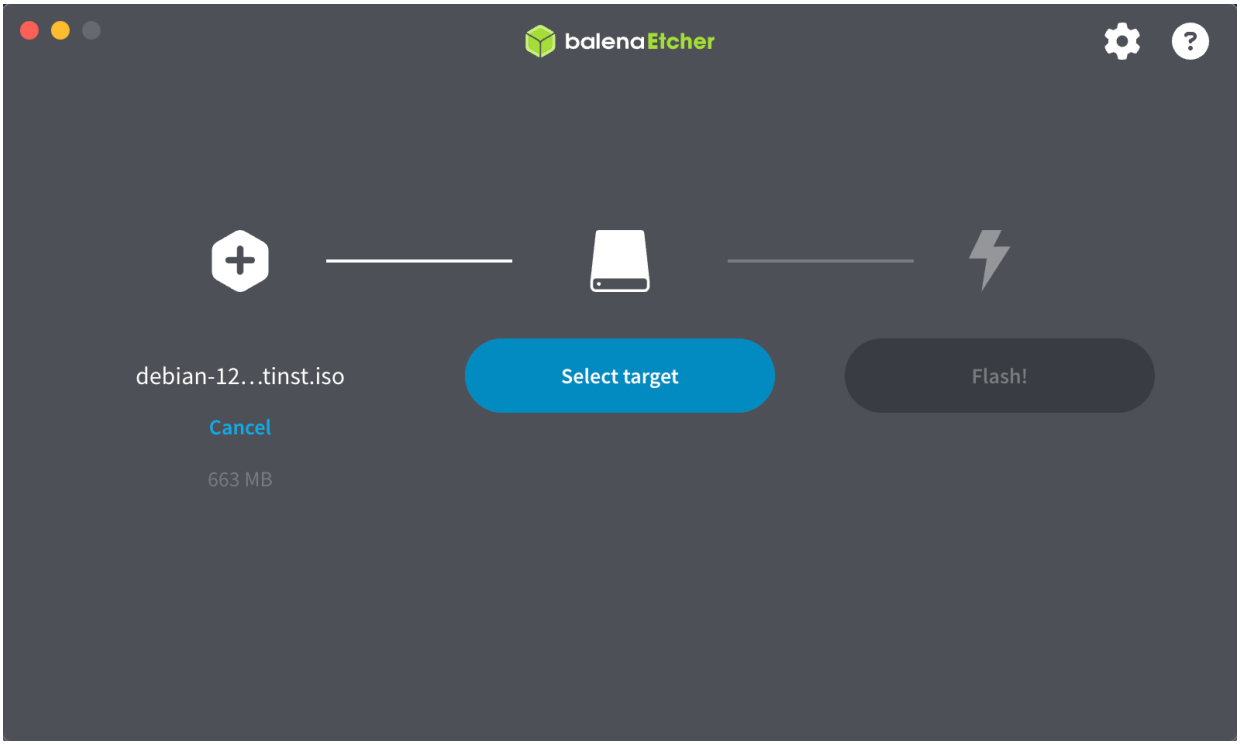


Ze stránky https://etcher.balena.io/#download-etcher stáhněte verzi pro svůj operační systém. Balena etcher slouží k vytvoření installační flešky z ISO souboru.

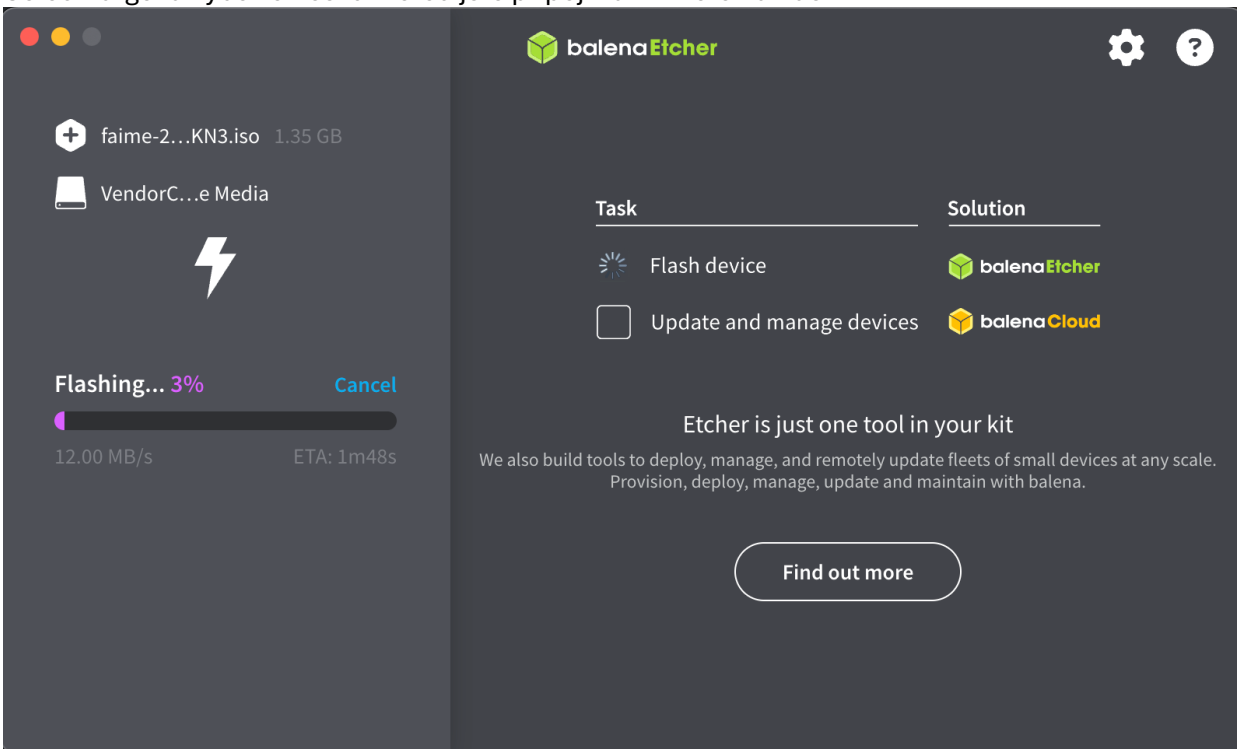
Nainstalujte a spusťe balena etcher



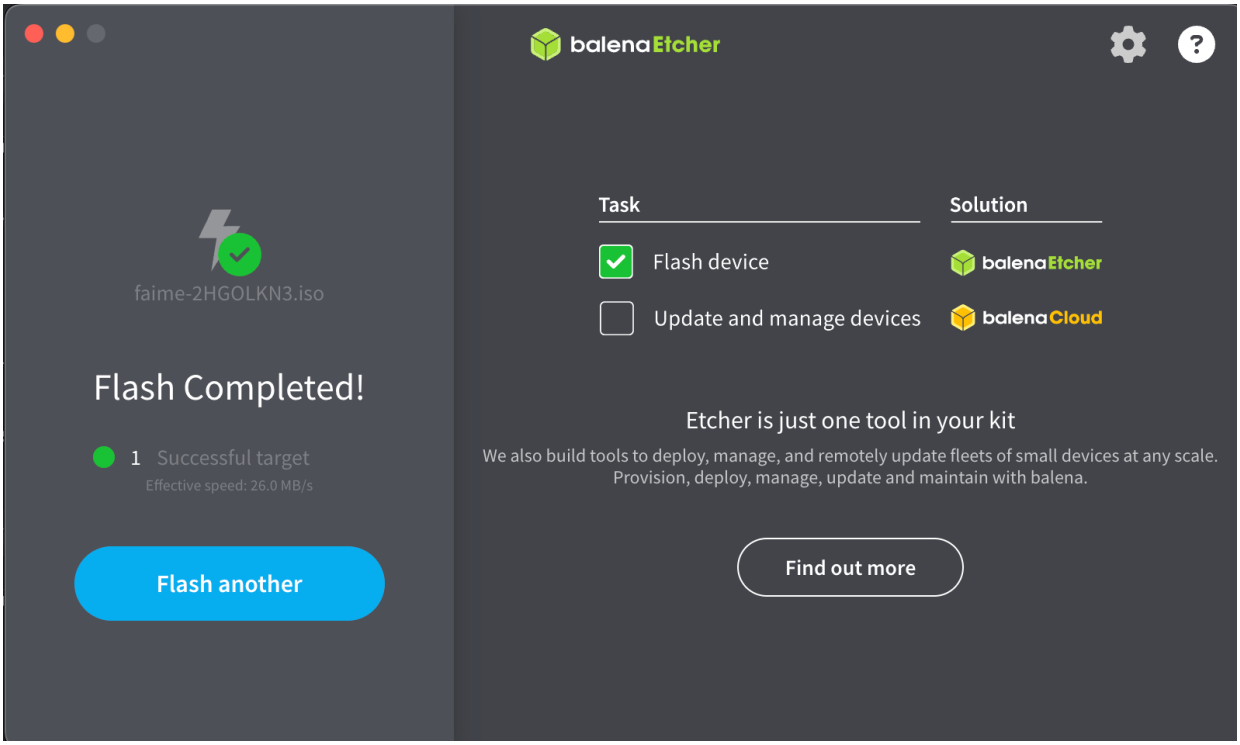
Klikněte na Flash from file
Vyberte ze stažených ISO debianu stažené v prvním kroku
Připojte USB flešku k počítači



Select target a vyberte flešku kterou jste připojili a klikněte na flash



vyčkejte až proces skončí



nyní je na USB flešce uložený intalátor debianu 12

Vložte USB disk do počítače kam chcete nainstallovat debian, a restartujte počítač, při zapínání počítače mačkejte na klávesnici klávesu která vás dostane do menu ve kterém vyberete disk ze kterého chcete bootovat (USB fleška kterou jste vložily před restartem), pokud nevíte která klávesa to je, použijte google, většinou to jeden výrobce má stejné a málokdy se tato klávesa mění.
Poté vyberte graphical install a nainstallujte debian dle svých potřeb, instalátor se vás na vše důležité zeptá. Doporučuji nenastavovat uživatele root, když v instalátoru nenastavíte uživatele root, váš uživatel bude automaticky přidán do sudo užitelů takže bude moci spouštět příkazy jako správce.

3 Instalace Dockeru

naprostou většinu aplikací budeme spouštět v dockeru takže je důležité ho nainstallovat. Nebudu sem vkládat konkrétní příkazy protože by se návod mohl stát velice rychle zastaralím a nepoužitelným. Otevřete https://docs.docker.com/engine/install/debian/ a postupujte podle pokynů. Tímto si nainstallujete docker engine který vám umožní provozovat containery.

4 RAID-Disků

NAS běží na debianu takže použiji apt k installaci balíčku mdadm který obsahuje vše co potřebuji

Installace:
```
apt install mdadm -y
```

mdadm mi dovolí vytvořit pole které má nastavený specifický typ raidu a jdou v něm něáké disky
```
mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/sdX
/dev/sdY
```

Zde je více informací o raid levlech https://en.wikipedia.org/wiki/Standard_RAID_levels
raid-devices určuje kolik zařízení v raid pool bude
a dále příkaz obsahuje disky ze kterých se raid pool bude skládat

pole nám je niní přístupné ve /dev/md0 jako standartní disk a mohu ho zformátovat a připojit

protože chceme využívat ext4 zformátujeme pool na ext4

```
mkfs.ext4 /dev/md0
```

nyní je disk zformátovaný ale nemám ho nikde připojený takže na něj nemůžeme zapisovat a číst

do souboru /etc/fstab vložím tento řádek, tato úprava fstab zařídí aby se disk automaticky připojil po každém restartu systému na dané místo (/mnt/disk1),

```
UUID=27857625-cb43-494b-afc2-c620b8ae3ae2       /mnt/disk1        ext4
      defaults,nofail         0       2
```

pokud po vložení tohoto řádku nechcete nas restartovat můžete použít příkaz

```
mount /mnt/disk1
```

tento příkaz vezme informaci z fstab pro /mnt/disk1 a připojí disk s uuid z fstab souboru.

5 Vysvětlení obsahu fstab souboru

SLOUPEC 1 UUID je unikátní označení disku aby system věděl který disk a nezáleželo na tom když ho připojím třeba do jiného slotu a bude se jmenovat jinak a UUID poolu zjistím takto: blkid /dev/md0

Výstup: /dev/md0: UUID="27857625-cb43-494b-afc2-c620b8ae3ae2" BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="eb49729f-8165-4b89-9702-a5e55794a802"

SLOUPEC 2 mountpoint je složka do které se disk připojí, disky které připojuji každy start systému se většinou připojují do složky /mnt/

SLOUPEC 3 typ filesystemu označuje jaký filesystem má systém očekávat, ext4 v tomto případě

SLOUPEC 4 jsou parametry připojení disku, použiju defaults a k nim přidám parametr nofail který nastaví aby když se disk při startu nepovede připojit aby systém I tak nabootoval, když by tam parametr nebyl, pool bychom třeba odebrali ale neodstranily ho z fstab byl by to zbytečný problém k řešení. Kdyby tam parametr nofail nebyl systém nám bez tohoto disku nenabootuje.

SLOUPEC 5 nastavuje jestli bude zálohován filesystem

SLOUPEC 6 kontrola filesystemu na disku, 0 žádná, 1 kontroluje se prvni (systémový disk), 2 kontroluje se poslední (ostatní disky)

6 swap file

Swap file je důležitou součástí naší NAS, je používám při větším obsazení ram pro procesy které momentálně nejsou aktivní (nepoužívají obsah uložený v RAM). Je důležité mít swapfile nastavený i přesto že se vám zdá že ram není zas tolik využívána, do RAM se totiž cachují soubory které systém často používá, aby k nim měl rychlejší přístup, a zpravidla toto cachování zaplní RAM celou i když to uživatel nevidí, například v programu htop je vidět v kategorii Mem žlutá část která označuje

cachování ale tato část není započítáná do celkově využívané paměti, takže v ukazovaných 2.04G není započítáná žlutá část, která z mojí zkušenosti časem poroste.

Běžně používaná velikost swapu je polovina kapacity RAM
možnosti na využití swapu jsou dvě, swap soubor/file a nebo swap oddíl na disku, já mám mnohem radši swap file kvůli větší variabilitě, nemusím vytvářet samostatný oddíl a při zvětšení nebo zmenčení také je jednoduší zmenšovat nebo zvětšovat soubor místo oddílu na disku.
Pro vytvoření swap souboru použijeme tento příkaz, vytvoří prázdný soubor o velikostí 2 gigabajty

```
sudo fallocate -l 4G /swapfile
```

nastavíme pro tento soubor práva tak aby z něj nikdo kromě systému, nemohl číst ani zapisovat, protože v ram jsou uložená kritická data

```
chmod 600 /swapfile
```

následně můžeme na tento swap soubor začít swapovat pomocí příkazu.

```
swapon /swapfile
```

Toto ale není trvalá změna protože po restartu systému se na swap swapovat automaticky nezačne.
Aby tomu tak bylo musíme swap přidat souboru /etc/fstab, tento řádek přidejte do fstab souboru

```
/swapfile none swap sw 0 0
```

7 SSH

SSH je služba které mi umožní připojení k NAS a provádění nastavení tak že to je bezpečné.
Základní konfigurace ssh v souboru /etc/sshd.conf

Doporučuji v konfiguraci nastavit řádek PermitRootLogin no, protože tím zabráníte přihlášení uživatele root přes ssh, root je častým terčem brute force útoků protože je jedoduší hádat pouze heslo než hádat uživatele a heslo.

Přihlašování:

SSH má vice možností přihlášení, rozdělil bych to na 3 kategorie heslo, klíč(např.: ed25519 nebo ecdsa) a třetí možnost, pro mě nejzajímavější je klíč pomocí bezpečnostního klíče (ed25519-sk,ecdsa-sk)

Používat ssh s heslem může být bezpečné ale pouze pokud používáte silné heslo a ostatní uživatelé taky, což né vždy je ovlivnitelné, můžete nastavit minimální požadavky ale to nepovede vždy k tomu k čemu chcete, můžete se dostat do stavu kdy bude heslo uživatele vypadat jako něco z tohoto:
Aa123456.
1Aaaaaaa.

Můžeme přidat délku hesla ale tím se také nedostanem moc daleko, něco jako
Aa123456789101112.
To taky není to co chceme

Řešení které je až spásné jsou ssh klíče, kdy uživatel má na svém zařízení uložený ssh klíč a když se chce k NAS připojit klíčem podepíše a je přihlášen, k ssh klíčum lze přidat passphrase přes kterou je klíč šifrován.
Tyto řádky v konfiguraci jsou zajímavé

```
PasswordAuthentication no # zakáže přihlášení heslem
```

```
PubkeyAuthentication yes # povolí přihlášení klíčem
PubkeyAcceptedAlgorithms ssh-ed25519 # vynucuje použití klíče ed25519 protože oproti rsa a ecdsa je bezpečnější
AuthorizedKeysFile .ssh/authorized_keys # nastavuje kde bude soubor s povolenými klíči pro uživatele hledán.
```

Ale co když uživatel bude mít na počítači třeba keylogger? To je problém který řeší ed25519-sk.
Který vyžaduje připojení fido2 zařízení a podepsání připojení k NAS pomocí fido2 klíče a klíče uloženého na počítači, tudíž když by někdo získal přístup k fido klíči, bude mu k ničemu protože nemá zbytek který je na počítači a když by někdo získal přístup k počítači musel by získat přístup ještě k fido2.

Zde ukáži jak vytvořit klasický ed25519 klíč s passphrase

Nejdřív klíč vygenerujeme, na počítači.
(Ujistěte se že používáte nejnovější verzi ssh)

```
ssh-keygen -t ed25519
```

```
root@bekucera-macbook:~# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/var/root/.ssh/id_ed25519):
Enter passphrase for "/var/root/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/root/.ssh/id_ed25519
Your public key has been saved in /var/root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:vWj4ELYD2WOSxmlTF8UJF+8AVfqIn2UfKL+O4fxpU1I root@bekucera-macbook.local
The key's randomart image is:
+--[ED25519 256]--+
|        +==+.    |
|         +oo     |
|      . . o .    |
|   . * . o = .E  |
|    @ * S + *..  |
|   o * = o B.... |
|      = o = .o.  |
|       = o ooo   |
|        . ++=.   |
+----[SHA256]-----+
root@bekucera-macbook:~#
```

Teď se nám klíč uložil do ~/.ssh/id_ed25519 k tomuto souboru se nesmí nikdo dostat a jediný kdo by měl mít právo z něho číst byste měl být vy.

```
root@bekucera-macbook:/var/root# cat .ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABDj+XID1b
Fo6y1aitC/rby0AAAAGAAAAAEAAAAzAAAAC3NzaC1lZDI1NTE5AAAAIJWIQ0iPU4ZG+0C+
lfajwqL9xwmAJD/xz17K/fG+imp8AAAAoG5NnFBmrw/JCnJcL6pTh8hm9w1nJcJfn580R2
qAESlu08c+4Pf1eHeXLZqwyMCKqv8OEN5AKC7KIZZ1KFGfRPo0Qn1gPXv11R84wuRUtS/L
r5R/fdJ633yTy/F6C42xXAqKd+bRxT+ZwkyAck7iX6xYgk7QcdoFBaa5QFq7KGwz1wRr8t
V9pueQ4FrIPWU/69KJ6Jdbfc2PcXPO/QsRolU=
-----END OPENSSH PRIVATE KEY-----
root@bekucera-macbook:/var/root#
```

Public část klíče se uložila do ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJWIQ0iPU4ZG+0C+lfajwqL9xwmAJD/xz17K/fG+imp8 root@bekucera-macbook.local
v mém případě vypadá takto
těď musíme klíč přidat na server aby ho server akceptoval, otevřeme .ssh/authorized_keys na NAS
nano .ssh/authorized_keys
a na nový řádek vložíme ed25519.pub klíč.

```
ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIJWIQ0iPU4ZG+0C+lfajwqL9xwmAJD/xz17K/fG+imp8
root@bekucera-macbook.local
```

Nyní se zkusíme připojit a všechno by mělo fungovat bez problému
systemctl reload sshd

reload místo restart zde použijte protože ho sshd podporuje a když použiji reload a v konfiguraci je chyba sice přikaz zobrazí chybu ale nespadne ssh server což by třeba když jsem od NAS fyzicky daleko mohl být problém.
Dalším dobrým způsobem omezení prostoru na útok je omezení ssh pouze na konkrétní uživatele
groupadd ssh_users
tím vytvoříme skupinu ssh_users
nyní do skupiny přidáme našeho uživatele

```
usermod -aG ssh_users bekucera
```

a nyní můžeme přidat řádek omezující připojení přes ssh do konfiguračního souboru ssh (/etc/ssh/sshd_config)

```
AllowGroups ssh_users
```

zase restartujeme ssh a zkusíme se připojit v novém okně

```
systemc    tl reload ssh
```

8 Firewall

Pro jednoduchost zde použiji UFW který neumožňuje dělat plnohodnotná pravidla ale pro tyto účely nám bude stačit.
První co budeme chtít udělat předtím než firewall zapneme bude povolení ssh a blokování všeho ostatního

```
      ufw allow from any to any port 22 proto tcp comment "allow ssh"
```

ufw allow - specifikuje že přidáváme pravidlo na povolení komunikace
ufw kromě allow umí také:
limit - limituje počet připojení a 3 za půl minuty
reject - zahodí packet a dále už nic neřeší
deny - zahodí packet a pošle zpátku ICMP error

from any - znamená z jakých hostů se pravidlo má uplatnit, v tomto případě kdo se může na ssh připojit
lze specifikovat ip addressu nebo subnet ip address
10.10.15.0/24
10.10.16.15

to any - pro jaký cíl se má pravidlo uplatnit, přes jakou addressu se mouhou připojit na ssh
port 22 - na jaký port se pravidlo uplatňuje, ssh má port 22, kdybychom chtěli přidat více portů do jednoho pravidla je to možné, napíšeme je za sebe oddělené pouze čárkou příklad 25,465,993

proto tpc - jaký protokol bude použit ssh pracuje na protokolu tcp další možnost je udp
comment "allow ssh" - nám umožní přehledněji s pravidli pracovat

teď jdeme nastavit aby ufw defaultně komunikaci odmítal

```
ufw default incoming reject
```

nyní když se máme již jak připojit tak můžeme ufw aktivovat.

```
ufw enable
```

Když chceme si zobrazit stav ufw použijeme příkaz

```
ufw status
```

mazání nebo editování pravidel na ufw může být relativně složité
jedna z možností mazání je napsat

```
ufw delete %cele_puvodni_pravidlo%
ufw delete allow from any to any port 22 comment "allow ssh"
```

ale mnou oblíbená cesta je

```
ufw status numbered
ufw delete 2
```

ufw editování pravidel neumožňuje takže jediná možnost je pravidlo odstranit a vytvořit znovu tak aby vám vyhovovalo

docker působí problém že všechny porty které jsou publikované z docker containeru obcházejí pravidla firewallu takže jsou dostupné i když pro ně není přidané pravidlo, tento problém lze vyřešit tím že v compose souboru na řádek na kterém publikuji port přidám před port adresu 127.0.0.1 takto:
před úpravou

```
name: uptime-kuma
services:
  uptime-kuma:
    image: louislam/uptime-kuma:1
    container_name: uptime-kuma
    volumes:
        - './data:/app/data'
    ports:
```

```
        - '8031:3001'
    restart: always
    mem_limit: 512m
    cpus: "0.5"
```

po úpravě

```
name: uptime-kuma
services:
  uptime-kuma:
    image: louislam/uptime-kuma:1
    container_name: uptime-kuma
    volumes:
        - './data:/app/data'
    ports:
        - '127.0.0.1:8031:3001'
    restart: always
    mem_limit: 512m
    cpus: "0.5"
```

toto zapříčiní že port je publikován pouze na localhost:8031 a není dostupný z ostatních počítačů. '8031:3001' Se rovná - '0.0.0.0:8031:3001' takže původně byl port publikován na všechny ip adresy zařízení.

9 Aplikace

Nyní když máme nastavený OS můžeme začít spouštět aplikace která chceme využívat
Nebudu sem vkládat konkrétní compose.yml soubory ze stejného důvodu jako příkazy na installaci Dockeru, staly by se brzo zastaralými.

9.1 SMB

Pro sambu jsem použil vlastní docker image protože image které jsem našel byli zastaralé a neudržované.
Tento image bude mít vždy nejnovější verzi samby protože pokaždé když se builduje, stáhne nejnovější sambu která je na repozitáři alpine linuxu.
Samba kvůli svému provázání se systémem, nedovolí spravovat uživatele kteří nejsou přidané v systému takže musím nedříve přidat uživatele do systému v kontejneru a až potom s ním pracovat s rámci samby. Přidávám volume ./etc:/etc protože samba je závislá na systému do takové míry že tuto persistanci potřebuje, kvůli souborům /etc/passwd /etc/shadow atd. Pro vytvoření uživatele je tedy potřeba nejdříve vytvořit uživatele na linuxu v containeru pomocí
`docker compose exec -it samba sh`
a do tohoto terminálu napište
`adduser %username%`
tímto se vytvoří linux uživatel s vámi vybraným jménem.
teď už zbývá pouze přidat uživatele do samby a nastavit mu heslo v sambě
`smbpasswd -a %username%`
nyní už užavetele můžeme napsat do konfigurace samby tak aby měl přístup ke konkrétní složce.

Nechci sambě věnovat moc času a výrazně nedoporučuji je používat, kvůli stabilitě, bezpečnosti, a složistosi na nastavení.

9.2 Immich

Aplikace sloužící k zálohování fotek z telefonu na NAS, má webové rozhraní ve kterém se mohu dívat na fotky, má aplikace pro android i IOS. Server Immich je jednoduchý k nastavení, zde je tento postup hezky popsaný a vysvětlený https://immich.app/docs/install/docker-compose.

9.3 Jellyfin

Využívá se k ukládání a přehrávání filmů.
https://jellyfin.org/docs/general/installation/container v oficiální dokumentaci se udává že se má použít network-mode=host což bych ale nedoporučovat a sám používám pouze reverse proxy směřující na port 8096 v kontejneru.

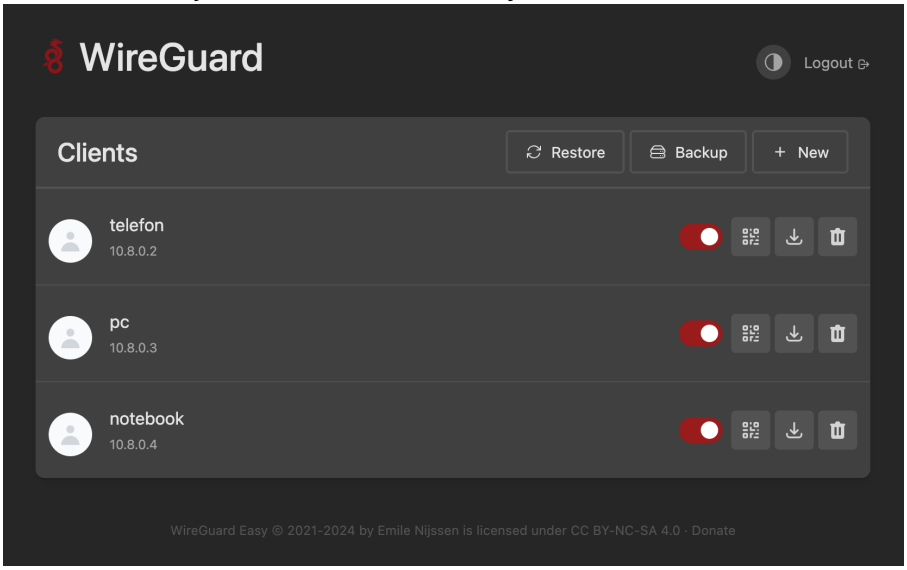
9.4 Searxng

https://docs.searxng.org/admin/installation-docker.html#installation-docker
Searxng ve svojí dokumentaci ukazuje pouze možnost spuštění pomocí docker run kterou ale z důvodů zmíněných v dokumentaci nechci využívat, takže využiji nástroj jako **https://it-tools.tech/docker-run-to-docker-compose-converter** který umí přepsat docker run na docker compose, umím to také a často výstup z toho programu upravuji protože nesedí mým požadavkům ale je jednodušší vložit docker run příkaz to stránky jako je tato a potom upravit výsledek než přepisovat docker run na compose ručně. Z v searxng doporučuji vypnout vyhledávací enginy které budou odpovídat příliš dlouho jinak dotaz na vyhledávání může trvat až 7 sekund, 7 sekund čekat

na zobrazení výsledku vyhledávání je otravně, já osobně mám zapnuté pouze největší vyhledávací enginy jako google, bing, seznam, brave, duckduckgo a wikipedia

9.5 Wireguard

Wireguard je složitý na správu z terminálu, je udělaný tak že malá změna vyžaduje nejméně 3 příkazy což není rychlé a effektivní. Z tohoto důvodu využívám wg-easy který je jednoduše spustitelný v dockeru https://github.com/wg-easy/wg-easy/blob/master/docker-compose.yml
a má jednoduché a rychlé rozhraní na správu ve webu.



9.6 Minecraft JAVA server

Na Minecraft server jsem si také musel vytvořit vlatní docker image protože nikde nebyl žádný který by mi vyhovoval https://gitea.bekucera.uk/bekucera/mc-server_public.git
stačí stáhnout spustit docker compose up, poté schválit eulu a můžete jít hrát.

9.7 Uptime Kuma

https://github.com/louislam/uptime-kuma
Uptime kuma také poskytuje pouze docker run příkaz ale tento problém už umíme vyřešit.
https://github.com/louislam/uptime-kuma/wiki/Reverse-Proxy
Uptime kuma má složitý popis jak nastavit reverse proxy ale veskutečnosti stačí standartní reverse proxy v apache2 jednoduše nastavená.

```
<VirtualHost *:80>
        ServerName kuma.nas.local

        ProxyPreserveHost On
        ProxyPass / http://uptime-kuma:3001/
        ProxyPassReverse / http://uptime-kuma:3001/

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

9.8 Reverse proxy

Jako reverse proxy využívám apache2 který běží v kontejneru, tento apache2-rp přidám do sítě s každým kontejnerem který má být dostupný a na apache2 nastavím aby dotazy například na jellyfin.nas.local odesílal na kontejner jellyfin. V popisku uptime kuma je příklad konfigurace reverse proxy.
Reverse proxy je jedinou vyjímkou kde vložím compose.yml, kvůli tomu aby mohl ukázat jak vypadá přidání do sítí a jak ukládám configuraci apache

```
name: apache2-rp
services:
    apache2-rp:
        image: 'ubuntu/apache2:latest'
        ports:
            - '80:80'
        volumes:
          - ./config:/etc/apache2
        environment:
            - TZ=UTC
        container_name: apache2-rp
        networks:
          jellyfin_default:
          immich_default:
          adguard_default:
          searxng_default:
          wg-easy_default:
          uptime-kuma_default:

networks:
  jellyfin_default:
    external: true
  immich_default:
    external: true
  adguard_default:
    external: true
  searxng_default:
    external: true
  wg-easy_default:
    external: true
  uptime-kuma_default:
    external: true
```