Alpine Linux

Introductie

Alpine Linux (vanaf nu gerefereerd als Alpine) is een super light-weight Linux distributie met als voornaamste eigenschappen dat deze weinig resources nodig heeft, secure is en simpel in gebruik.

Een gemiddelde basis installatie van Alpine heeft ongeveer 130MB aan ruimte nodig.

Er zijn ook verschillende varianten van Alpine. Zo is er zelfs een speciale variant die volledig uitgekleed is voor het maken van Docker containers. Deze heeft maar **8MB** aan disk space nodig.

Alpine schuwt ook weg van alle extra complexiteit. Het hele init systeem maakt gebruik van OpenRC in plaats van systemd en er is een simpele package manager genaamd APK die quasi werkt zoals APT.

Alpine is ideaal als basis voor een server of project waarbij je niet een gigantische distributie wenst te voorzien. Een andere plek die baat heeft is wanneer je een Raspberry Pi gebruikt.

Musl libc

Een Linux distributie is nauw verwoven met de onderliggende c library. In de meeste gevallen is dit glibc.

Alpine maakt gebruik van musl libc in plaats van glibc. Musl ondersteunt geen `upgrade to tcp` optie voor dns requests. Een gevolg daarvan is dat dns requests boven 512 byte afgekort worden tot 512 byte.

In 99% van de gevallen ga je hier geen issues van ondervinden. Echter in speciale use cases waarbij je een gigantische dns response verwacht (zoals distributed host discoveries) kan je in de problemen komen wanneer je niet alle data ontvangt. Mocht je twijfelen: in die 512 byte kan je ~15A records of 8 AAAA records plaatsen als je de maximum lengte gebruikt van 255 karakters. Tegen dat je die limiet bereikt heb je normaal gezien al een ander probleem.

Echter heeft hier al redelijk wat digitale inkt over gevloeid. Maar de spec in RFC5966 laat dit toe: https://datatracker.ietf.org/doc/html/rfc5966#section-4:

```
Stub resolver implementations MAY omit support for TCP when
specifically designed for deployment in restricted environments
where truncation can never occur or where truncated DNS
responses are acceptable.
```

Installatie

Voor de installatie gebruiken we een nieuwe virtuele machine. Deze heeft niet veel resources nodig. Om te starten is het volgende *meer dan* voldoende:

- 1 cpu
- 1GB ram
- 16GB hard disk

> **Let op:** Alpine linux heeft problemen wanneer je gebruik maakt van secure boot. Vergeet dit dus niet uit te schakelen

Start op van het iso bestand. Zodra Alpine geboot is log je in met `root`.

Je kan nu de installatie starten met het `setup-alpine` commando. Dit gaat je alle vragen stellen die nodig zijn voor een Alpine op te zetten. Indien je tijdens de installatie `[antwoord]` ziet staan betekent dat dit de default waarde is.

Keyboard

De keymap is een persoonlijke keuze, ik typ qwerty dus in mijn geval is dit `us` en dan `us-intl`. Mocht je azerty typen dan heb je het type `be` en `be-period`.

Hostname

De hostname is ook weer een persoonlijke keuze. Je kiest hier de naam die je server krijgt in het netwerk. In dit geval kies ik voor `cursus-alpine`.

Networking

Je stelt `eth0` in. Je kan kiezen voor `dhcp` of voor een ip adres. Voor mijn netwerk opteer ik voor een statisch ip. Ik geef het ip adres in in CIDR notatie: `10.30.60.195/24`.

Als gateway kies ik `10.30.60.1`. Op de vraag voor manual network configuration antwoord ik `n`. Dit is enkel nodig indien je een speciale opstelling hebt waar je eventueel werkt met vlans etc...

Als dns domain name kies ik voor `lab.local`. En als dns nameservers `10.30.50.254,9.9.9.9`.

Het root wachtwoord

Het root wachtwoord is ook een persoonlijke keuze. Ik kies als wachtwoord `*********`.

Timezone

De timezone is ook een persoonlijke keuze. In mijn geval is dit `Europe/Brussels`. Als je twijfelt kan je een lijst opvragen met `?`.

Proxy

Geef je proxy server in indien je er een hebt. In mijn geval heb ik er geen dus antwoord ik `none`.

NTP client

Als ntp client is de standaard `chrony` daemon genoeg. Mocht je toch een andere daemon wensen dan kan je deze hier kiezen.

Packet mirror

Je krijgt nu een gigantische lijst met package mirrors. Kies gewoon er eentje die kortbij ligt. Ik kies voor optie `29`, deze is in dit geval `alpine.42.fr` omdat Frankrijk relatief kortbij is vergeleken met de andere mirrors. Elke mirror gaat ongeacht werken maar het komt je download/installatie snelheid alleen maar ten goede als je een mirror kiest die korterbij ligt.

Ssh server

De standaard `openssh` server is perfect voor ssh toegang. Deze configureren gaat ook het makkelijkste zijn indien je reeds ervaring hebt.

Installatiedisk kiezen

Je kan nu de installatiedisk kiezen en hoe je deze wenst in te delen. Ik kies als disk `sda`, type `lvm` en dan als doel `sys`. Je kan deze laatste 2 stappen ook in 1 keer doen door als type `lvmsys` op te geven.

> **Tip:** Je hoeft niet perse voor lvm te kiezen. Ik verkies dit persoonlijk om dat het mij later extra flexibiliteit biedt voor het uitbreiden en/of verplaatsen van mijn disks/volumes.

Alpine scant nu snel even alle disks. Als je de vraag krijgt of je zeker bent dat je wenst te installeren antwoord je `y`.

Wacht nu even tot Alpine op jouw disk geinstalleerd is. Als alles klaar is herstart je je Alpine met `reboot`.

Eerste configuratie

Zodra je server opnieuw opgestart is log je in met `root` en jouw wachtwoord. Je kan de server nog niet benaderen via ssh. De reden hiervoor is dat rechtstreeks inloggen als root niet toegelaten is.

We beginnen met de `nano` editor te installeren. Je kan dit doen met `apk`. Apk (Alpine Package Keeper) is de installatie tool van Alpine. Je installeert nano met `apk add nano`. Dit zou normaal heel snel moeten verlopen.

Een user aanmaken

Uit veiligheid gaan we nooit rechtstreeks met root aanmelden. We maken dus best een aparte gebruiker. De eerste stap is het installeren van sudo met `apk add sudo`. Dit laat je toe om je privileges te escaleren. Later wanneer je meer ervaring hebt met Linux kan je zelfs je sudoers file aanpassen zodat je deze rechten heel specifiek definieert. Echter in ons geval gaan we gewoon het simpel houden.

Maak een administrator user aan met `adduser administrator`. Voeg deze user dan toe aan de `wheel` group met `addgroup administrator wheel`.

Test dit door `groups administrator` uit te voeren. Dit commando gebruik je om alle groepen op te vragen waar een user zich in bevindt.

> **Tip:** In de meeste Debian based Linux distributies maakt sudo gebruik van de `sudo` groep. Echter gebruiken andere distributies de `wheel` groep. Dit komt van de term `big wheel` uit de jaren 60 en betekent iemand met veel macht. Iets moderner wordt er soms "The big cheese" gezegd.

De laatste stap is zorgen dat deze groep herkend wordt door sudo. Hiervoor heb je het `visudo` commando nodig. Dit comando maakt gebruik van de `vi` editor. Als je niet gewoon bent met deze editor te werken kan dit even verwarrend zijn. Echter geef ik je zometeen alle stappen die je moet uitvoeren.

De reden dat we `visudo` in plaats van `vi` gebruiken is omdat `visudo` de file volledig nakijkt voor dat deze opgeslaan wordt. Dit zorgt ervoor dat je je sudo configuratie niet kan stukmaken.

Voer `visudo` uit. Scroll naar beneden tot je de lijn `# %wheel ALL=(ALL) ALL` vindt. Als je je op deze lijn bevindt duw je op `i`. Je bevind je nu in insert modus. Verwijder de `#` en de spatie zodat je enkel `%wheel ALL=(ALL) ALL` over houdt. Duw nu op de `escape` toets en type dan `:wq` (write and quit) voor het bestand op te slaan en visudo te sluiten.

> **Tip:** Mocht je toch de sudo group willen gebruiken dan uncomment je de lijn met de sudo group. Je moet dan wel de sudo group aanmaken met `addgroup sudo` en de administrator ook in deze group plaatsen met `addgroup administrator sudo`.

Je kan dit nu alles testen door te switchen naar de administrator met `su administrator`. Wanneer je nu `sudo apk update` uitvoert moet je je wachtwoord even opgeven en zou alles moeten werken.

Je kan je nu via ssh verbinden. Dit werkt veel prettiger dan een console.

Update en upgrade

Voor dat we verder werken gaan we eerst zien dat we volledig up to date zijn. Doe dit even met `sudo apk update ; sudo apk upgrade`. Dit vernieuwt de lijst met packages en installeert een nieuwere versie indien deze beschikbaar is.

De configuratie aanpassen

Alpine voorziet makkelijke tools om snel aanpassingen te doen. Typ in je terminal even `setup-` en duw op de tab toets. Je krijgt dan een lijst met setup tools die beschikbaar zijn. Zo kan je makkelijk de tijdzone, keyboard, netwerkinstellingen etc... aanpassen

Repositories

Een belangrijk pakket wanneer je virtualiseert zijn de `open-vm-tools`. Deze laten jouw virtualisatie platform toe om te praten met jouw guest os en de nodige optimalisaties uit te voeren. Installeer deze met `sudo apk add open-vm-tools`.

Je krijgt nu de volgende foutmelding:

```
cursus-alpine:~$ sudo apk add open-vm-tools
ERROR: unable to select packages:
  open-vm-tools (no such package):
    required by: world[open-vm-tools]
```

De reden hiervoor is dat je niet alle repositories hebt ingeschakeld/toegevoegd. Surf even naar: https://pkgs.alpinelinux.org/packages. Deze website bevat alle packages die gekend zijn in de standaard Alpine repositories. Wanneer je zoekt op `open-vm-tools` zie je dat deze beschikbaar zijn in de `community` repository met als branch `edge`.

> **Tip:** Standaard zoekt de Alpine package website naar pakketten in de `edge` branch. Dit is is de bleeding edge ofwel ook de `latest` branch. Indien je enkel gegarandeerd stabiele pakketten wenst kan je je branch instellen op de versie van Alpine die je gebruikt.

Je kan deze activeren door je repositories file aan te passen. Voer hiervoor `sudo nano /etc/apk/repositories` uit. Je opent nu je repositories file met nano. Haal het `#` weg voor beide lijnen die `community` bevatten. Indien je enkele gegarandeerd stable pakketten wenst uncomment je enkel de lijn waarin jouw Alpine versie vermeld staat. Sla het bestand op met `ctrl + o en dan ctrl + x`.

Alpine weet nog niet welke nieuwe pakketten er beschikbaar zijn. Vernieuw de lijst dus even met `sudo apk update`.

Als alles goed verlopen is kan je nu zoeken met `apk search open-vm-tools`. Je krijgt dan een lijst met alle mogelijke pakketten:

```
open-vm-tools-openrc-11.3.5-r0
open-vm-tools-guestinfo-11.3.5-r0
open-vm-tools-plugins-all-11.3.5-r0
open-vm-tools-dev-11.3.5-r0
open-vm-tools-vmbackup-11.3.5-r0
```

```
open-vm-tools-static-11.3.5-r0
open-vm-tools-hgfs-11.3.5-r0
open-vm-tools-timesync-11.3.5-r0
open-vm-tools-11.3.5-r0
open-vm-tools-doc-11.3.5-r0
open-vm-tools-vix-11.3.5-r0
open-vm-tools-lang-11.3.5-r0
open-vm-tools-gtk-11.3.5-r0
open-vm-tools-deploypkg-11.3.5-r0
open-vm-tools-dbg-11.3.5-r0
```

De absolute basis die je nodig hebt is `open-vm-tools-guestinfo`. Wij gaan echter gewoon alles installeren. Je doet dit met `sudo apk add open-vm-tools`

> **Tip:** Indien je meer informatie wenst over een pakket kan je de `-v` flag gebruiken met apk search.

Soorten repositories

In de basis zijn er 3 repositories:

- **main**
  - de meest basic pakketten die je op elk systeem kan verwachten. Hebben een lange support cyclus van 2 jaar.
- **community**
  - community aangeleverde pakketten uit testing die aan de nodige tests zijn onderworpen en stabiel worden geacht. Hebben een support cyclus van 6 maanden.
- **testing**
  - nieuwe, kapotte en oude pakketten. Deze repository heeft geen support en is enkel beschikbaar in de `edge` mirror.

Services

In het vorige gedeelte hebben we de `open-vm-tools` geïnstalleerd. Deze werken echter nog niet. De reden is dat de service nog niet gestart is. Tevens moeten we er ook voor zorgen dat deze service na het booten opstart. Dit doen we met de volgende commando's:

```
rc-service open-vm-tools start
rc-update add open-vm-tools
```

Dit lijkt vreemd indien je van een andere distributie komt. Waar een service meestal automatisch begint te draaien. Wanneer je Alpine gebruikt moet je altijd expliciet zijn. Services moet je dus zelf starten en je moet zelf aangeven of deze automatisch mogen starten.

rc-service

Services beheer je met `rc-service`. Je kan makkelijk services starten, stoppen, herstarten en de status opvragen:

```
rc-service open-vm-tools start
rc-service open-vm-tools stop
rc-service open-vm-tools restart
rc-service open-vm-tools status
```

Dit zijn de standaard functies die geïmplementeerd moeten worden. Sommige services kunnen hierop uitbreiden en hun eigen functies aanbieden. Dit vind je het makkelijkst terug in de documentatie van de service.

runlevels

Voor dat leren hoe rc-update werkt is het belangrijk dat we kort even linux runlevels overlopen. In essentie is een runlevel niet meer dan een punt in de tijd van een linux systeem. Je kan alle gekende runlevels van Alpine opvragen met `rc-status --list`. Je krijgt dan een overzicht van alle gekende runlevels:

```
boot
default
sysinit
nonetwork
shutdown
```

Elk runlevel vindt plaats op een bepaald punt tijdens het opstarten en gebruik van jouw linux systeem:

- **sysinit**
    - Dit runlevel runt tijdens de initialisatie van het systeem. Dit runlevel initialiseert speciale folders zoals `/dev`, `/proc` en `/sys`. Dit runlevel wordt eenmalig gerund tijdens het opstarten.
- **boot**
    - Je bevindt je in dit runlevel tijdens het booten. Je voegt hier services toe die je wilt dat ze in orde zijn voor dat het systeem boot. Je plaatst hier meestal mount points die moeten gekoppeld zijn voor de rest van het systeem verder boot.
- **nonetwork**
    - Dit speciale runlevel kan je gebruiken wanneer je services wilt laten runnen indien er geen netwerk is
- **default**
    - Het standaard runlevel. Het systeem is volledig geboot en geinitialiseerd
- **shutdown**
    - Dit runlevel wordt aangeroepen tijdens het afsluiten van een systeem

Dit zijn de standaard runlevels. Je kan ook zelf je eigen runlevels definiëren door een sub map aan te maken in `/etc/runlevels`. Dit wordt zometeen behandeld

rc-update

Nu dat je weet wat een runlevel is en waar zich dit situeert kan je makkelijk zelf bepalen waar je services laat starten. Zo kan je zien dat we de `open-vm-tools` service laten starten in default. We zijn dan zeker dat het systeem volledig geboot en geinitialiseerd is. Eerder hebben we dit toch niet nodig.

Een service toevoegen of verwijderen aan een runlevel kan je met `rc-update`. Als je geen runlevel opgeeft wordt standaard `default` gebruikt:

```
1  rc-update add open-vm-tools
2  rc-update delete open-vm-tools
3  rc-update add open-vm-tools sysinit
4  rc-update delete open-vm-tools sysinit
```

Je kan een service ook aan meerdere runlevels toevoegen indien je dit wenst. Elk (standaard) runlevel heeft een map in `/etc/runlevels`. Wanneer je een service aan een runlevel toevoegt wordt voor deze service een symlink aangemaakt in de juiste map.

Voor een overzicht van al je geconfigureerde services kan je `rc-status` gebruiken:

```
Runlevel: default
 acpid                                                [  started  ]
 sshd                                                 [  started  ]
 open-vm-tools                                        [  started  ]
 chronyd                                              [  started  ]
 crond                                                [  started  ]
Dynamic Runlevel: hotplugged
Dynamic Runlevel: needed/wanted
 sysfs                                                [  started  ]
 fsck                                                 [  started  ]
 root                                                 [  started  ]
 localmount                                           [  started  ]
Dynamic Runlevel: manual
```

Je kan ook alle mogelijke services opvragen met `rc-status -s`.

In de lijst zie je enkele runlevels waarvoor er geen map bestaat in `/etc/runlevels`. Dit is omdat dit dynamic runlevels zijn. Deze zijn voor gedefinieerd in openrc. `hotplugged` wordt gebruikt wanneer je hardware inplugt op je toestel. `manual` zijn services die je manueel moet starten/stoppen en `needed/wanted` zijn services die door andere services vereist zijn. Zo kan je bijvoorbeeld `/etc/init.d/networking` openen en zien dat dit localmount` verwacht:

```
...
depend() {
        need localmount
        want dev-settle
        after bootmisc hwdrivers modules
        provide net
        keyword -jail -prefix -vserver -docker
}
...
```

Je kan ook je eigen runlevels maken en deze koppelen aan bestaande runlevels. Je hoeft hiervoor gewoon een map aan te maken onder de `/etc/runlevels` map. Stel dat je een apart runlevel wenst voor wanneer je gebruik maakt van vpn software. Je maakt dan een `vpn` map aan in de runlevels map.

Je kan nu de nodige pakketten installeren en deze toevoegen aan het vpn runlevel. Zodra je klaar bent kan je naar dit runlevel switchen met `openrc vpn`. Een ander voorbeeld zijn aparte runlevels voor wanneer een toestel gebruik maakt van batterij of stroom.

Je kan een runlevel ook **stacken** boven op een ander runlevel. Zo kan je het `vpn` runlevel alle services laten overerven van `default`. Dit spaart je een hoop configuratiewerk uit. Je kan dit doen met `rc-update -s add default vpn`.

Zelf service scripts schrijven

Sommige tools voorzien niet hun eigen service script. In andere gevallen kan het zijn dat je zelf enkele commando's wenst te runnen.

Je kan dan hiervoor een zelfgemaakt service script schrijven. Alle service scripts bevinden zich in `/etc/init.d`. Wanneer je dan het `rc-update` command gebruikt wordt er voor jou automatisch een symlink aangemaakt in het juiste runlevel.

De essentie

We beginnen met een nieuw service script genaamd `letsgo` aan te maken:

```
1  cd /etc/init.d
2  sudo touch letsgo
3  sudo chmod +x letsgo
```

Het is belangrijk dat je de `chmod` stap niet vergeet. Indien het script niet uitvoerbaar is kan openrc dit niet gebruiken.

Plaats nu de volgende content in dit script:

```
#!/sbin/openrc-run

depend() {
    need localmount
    need net
}

start() {
    /bin/echo "Let's go!"
}

stop() {
    /bin/echo "Cheerio!"
}
```

Dit is de absolute basis die je nodig hebt om zelf een service script te schrijven. Het script begint met de standaard `#!/sbin/openrc-run` shebang. Deze construct zorgt ervoor dat ons script met dit command wordt uitgevoerd.

De `depend` functie is om aan te geven wat deze service nodig heeft om te werken. In dit geval geven we aan dat deze service `localmount` en `net` nodig heeft. We gebruiken hiervoor het keyword `need`. Deze 2 services worden gebruikt om het file system te mounten en networking te starten. Deze zijn niet verplicht maar meestal een goeie basis.

Je kan eender welke service naam gebruiken achter een keyword.

Er zijn nog veel meer keywords zoals `use`, `want`, `after`, `before`,... Een overzicht kan je vinden op https://manpages.org/openrc-run/8 onder de sectie **"Dependencies"**. De belangrijkste keywords voor basis scripts zijn:

Keyword	Description
`need`	Hard dependency. De service kan niet starten zonder de aangegeven service
`use`	Soft dependency. Indien de aangegeven service(s) zich in hetzelfde runlevel bevinden worden deze ook gestart
`want`	Soft dependency. Probeert de services te starten en het maakt niet uit in welk runlevel de service(s) zich bevinden.
`after`	Start onze service na de opgegeven service(s).
`before`	Start onze service voor de opgegeven service(s).

Er is ook een `start` en een `stop` functie voorzien voor onze service te starten en te stoppen. Hier in plaats je alle commando's die nodig zijn om je service te starten en stoppen. Zie wel dat je altijd het volledige path mee geeft van elk command. Indien je twijfelt waar een command zich bevindt kan je altijd gebruik maken van het `which` command. Bijvoorbeeld `which echo`.

> **Let op:** Je moet niet zelf een `restart` functie voorzien. Openrc implementeert dit zelf door in de achtergrond `stop/start` te runnen.

Je kan dit nu makkelijk testen:

```
sudo /etc/init.d/letsgo start
Let's go!

sudo /etc/init.d/letsgo stop
Cheerio!
```

Een uitgebreid voorbeeld

Het vorige script werkt, en is perfect voor quick and dirty er voor te zorgen dat een service werkt. Maar het kan ook veel mooier en netter.

```
#!/sbin/openrc-run

description="Een service voor hallo en vaarwel."

depend() {
   need localmount
   need net
}

start() {
    ebegin "Starting ${RC_SVCNAME}"
    /bin/echo "Let's go!"
    eend $?
}

stop() {
    ebegin "Stopping ${RC_SVCNAME}"
    /bin/echo "Cheerio!"
    eend $?
}
```

Als eerste hebben we een `description` toegevoegd. Dit omschrijft wat onze service doet.

Je kan ook zien dat we in onze functies gebruik maken van `ebegin` en `eend`. Deze commando's zorgen er voor dat we een mooie output krijgen wanneer we onze service starten of stoppen:

```
sudo /etc/init.d/letsgo start
 * Starting letsgo ...
Let's go!                       [ ok ]

sudo /etc/init.d/letsgo stop
 * Stopping letsgo ...
Cheerio!                        [ ok ]
```

Enkele extra dingen die we opmerken is het gebruik van `${RC_SVCNAME}`. Dit is een vaste variabele in een openrc service script dat gelijk staat aan de naam van de service (meerbepaald de naam van de service file). Zo hoef je niet elke keer de naam van de service in te vullen.

Er zijn nog extra variablen beschikbaar. Deze kan je vinden op https://manpages.org/openrc-run/8 onder de sectie **"Environment"**.

Achter de `eend` functie zie je dat we gebruik maken van `$?`. In linux staat dit gelijk aan de exit status van het vorige commando. `eend` maakt hiervan gebruik om al dan niet een error weer te geven mocht dit niet 0 zijn.

Er zijn nog extra functies om alles mooi weer te geven. Deze kan je vinden op https://manpages.org/openrc-run/8 onder de sectie **"Builtins"**.

Je eigen functies voorzien

Het is perfect mogelijk om buiten de standaard functies ook je eigen functies te voorzien. In het onderstaande (uitgeklede) voorbeeld hebben we een `reload`, `checkconfig` en `clean` functie toegevoegd:

```
#!/sbin/openrc-run

extra_commands="checkconfig"
```

```
4  extra_started_commands="reload"
5  extra_stopped_commands="clean"
6
7  description="Een service voor hallo en vaarwel."
8
9  description_reload="Alle informatie herladen"
10 description_checkconfig="Test de config"
11 description_clean="Alle tijdelijke files uitkuisen"
12
13 depend() {
14    need localmount
15    need net
16 }
17
18 start() {
19     /bin/echo "Let's go!"
20 }
21
22 stop() {
23     /bin/echo "Cheerio!"
24 }
25
26 checkconfig() {
27     /bin/echo "Getting all the new info!"
28 }
29
30 reload() {
31     /bin/echo "Tijd voor een muntje!"
32 }
33
34 clean() {
35     /bin/echo "Cleaning!"
36 }
```

De functies maak je aan net zoals je andere functies al hebt aangemaakt. De volgende stap is aan te geven welke extra commando's er zijn en wanneer je deze mag gebruiken:

- `extra_commands` : Extra commando's die altijd mogen gebruikt worden
- `extra_started_commands` : Extra commando's die enkel mogen gebruikt worden als de service gestart is.
- `extra_stopped_commands` : Extra commando's die enkel mogen gebruikt worden als de service gestopt is.

Wanneer je **meerdere** commando's wenst te definieren dan moet je deze gewoon scheiden met een **spatie**: `extra_commands="checkconfig reload clean"` .

Je kan ook makkelijk een description geven voor elk command door `description_<commando>` te voorzien.

Wanneer je nu dit test zie je dat dit werkt zoals het hoort:

```
sudo /etc/init.d/letsgo stop
Cheerio!

sudo /etc/init.d/letsgo reload
 * letsgo: cannot `reload' as it has not been started

sudo /etc/init.d/letsgo describe
 * Een service voor hallo en vaarwel.
 * checkconfig: Test de config
 * reload: Alle informatie herladen
 * clean: Alle tijdelijke files uitkuisen
```

Er zijn nog veel extra mogelijkheden, zoals `required_files` , `required_dirs` , `chroot` ,... Je kan dit allemaal leren door je even te verdiepen in de manpage: https://manpages.org/openrc-run/8.

Een GUI installeren

Indien je wenst kan je voor Alpine ook makkelijk een window manager installeren. Let op dat je dit enkel doet als je echt een grafische omgeving nodig hebt. Voor een server is dit echt afgeraden.

De eerste stap is zorgen dat de X server (een onderliggend mechanisme van een window manager) de kernel kan gebruiken. Je doet dit met `sudo setup-xorg-base`.

Voor de rest moet je enkel nog je grafische omgeving installeren, deze starten en zorgen dat deze mee opstart met Alpine. In dit geval kiezen we voor de lightweight xfce window manager. Er zijn echter legio mogelijkheden.

Begin met xfce te installeren samen met de lightdm login manager zodat we een login scherm hebben:

```
apk add xfce4 xfce4-terminal xfce4-screensaver lightdm-gtk-greeter
```

De volgende stap is zorgen dat ons systeem met meerdere processen tegelijk kan communiceren alsook device events registreert:

```
rc-service dbus start
rc-service udev start

rc-update add dbus
rc-update add udev
```

Alles is nu klaar om de window manager te starten:

```
rc-service lightdm start
rc-update add lightdm
```

Wanneer we nu opvragen hoeveel geheugen gebruiken met `free -h` zien we dat we ~200MB geheugen in gebruik hebben. Netjes.

```
               total        used        free ...
Mem:           1.9G      199.3M        1.3G ...
Swap:          3.9G           0        3.9G
```

Een wireguard server opzetten

Een projectje waar we alles samen kunnen brengen is het opzetten van een Wireguard server. Wireguard laat ons toe om een simpele en veilige vpn server op te zetten. Voor dit project hebben we de volgende componenten nodig:

- Een Alpine server, met de main en community repositories
- Iptables
- Wireguard als vpn
- Wgdashboard om wireguard te beheren
- UFW als firewall

Een alpine server

Installeer een nieuwe Alpine. Voer alle voorbereidende stappen uit zonder een gui te installeren. Indien nodig installeer je ook de open-vm-tools.

Iptables

We gaan gebruik maken van iptables als firewall voor ons systeem. Dit is een simpele manier om een firewall te beheren en aan te maken. Dit ligt ook meer in lijn met alle documentatie die je online kan vinden. Het is echter aan te raden om nftables te gebruiken indien je hier mee bekend bent.

Je installeert iptables en zorgt er voor dat deze mee opstart met Alpine:

```
apk add iptables ip6tables

modprobe -v ip_tables
modprobe -v ip6_tables
modprobe -v iptable_nat

rc-service iptables save
rc-service iptables start
rc-update add iptables
```

We moeten dan nog even ip forwarding inschakelen. Dit doen we zodat we onze wireguard server als router kunnen gebruiken. Open hiervoor `/etc/conf.d/iptables` en zorg er voor dat `IPFORWARD="yes"` is. De inhoud ziet er dan als volgt uit:

```
# /etc/conf.d/iptables

# Location in which iptables initscript will save set rules
on
# service shutdown
IPTABLES_SAVE="/etc/iptables/rules-save"

# Options to pass to iptables-save and iptables-restore
SAVE_RESTORE_OPTIONS="-c"

# Save state on stopping iptables
SAVE_ON_STOP="yes"

# Enable/disable IPv4 forwarding with the rules
IPFORWARD="yes"
```

Herstart dan iptables en controleer of forwarding ingeschakeld is, je moet waarde `1` krijgen:

```
rc-service iptables restart
sysctl net.ipv4.ip_forward
```

Wireguard

Sinds Linux kernel v5.6 is wireguard een integraal onderdeel geworden van Linux. Wireguard is dan ook te vinden in de main repository van Alpine. Je kan wireguard makkelijk installeren:

```
sudo apk add wireguard-tools
```

De volgende stap is een basis configuratie aanmaken voor wireguard. Hiervoor heb je ook een public en private key nodig. Alle configuratie gebeurt in de `/etc/wireguard` map:

```
sudo su
cd /etc/wireguard/

umask 077
wg genkey | tee privatekey | wg pubkey > publickey
touch wg0.conf
```

Met `umask` stellen we de default rechten voor bestanden in. We genereren dan de public en private key voor wireguard te kunnen gebruiken. Als laatste maken we een lege file aan voor onze configuratie.

Jouw public en private key kan je vinden nu in `/etc/wireguard/publickey` en `/etc/wireguard/privatekey`.

Plaats in `wg0.conf` nu de volgende basis configuratie:

```
[Interface]
Address = 192.168.1.254/24
ListenPort = 51820
PrivateKey = jouwprivatekey
SaveConfig = true

PostUp = iptables -A FORWARD -i %i -j ACCEPT
PostUp = iptables -A FORWARD -o %i -j ACCEPT
PostUp = iptables -t nat -A POSTROUTING -o eth0 -j
MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT
PostDown = iptables -D FORWARD -o %i -j ACCEPT
PostDown = iptables -t nat -D POSTROUTING -o eth0 -j
MASQUERADE
```

Dit is alles wat we tot nu nodig hebben in onze initiele configuratie van wireguard. De rest van de configuratie wordt voorzien door ons dashboard. Het enigste wat ons nog rest is het voorzien van een service script dat onze wireguard interface automatisch activeert. Plaats de volgende inhoud in `/etc/init.d/wireguard`:

```
#!/sbin/openrc-run

description="Wireguard interfaces"

depend() {
    need localmount
    need net
}

start() {
     ebegin "Starting ${RC_SVCNAME}"
     /usr/bin/wg-quick up wg0
     eend $?
}

```

```
stop() {
    ebegin "Stopping ${RC_SVCNAME}"
    /usr/bin/wg-quick down wg0
    eend $?
}
```

We zorgen er ook voor dat wireguard automatisch de interface inschakelt wanneer Alpine opgestart is:

```
chmod +x /etc/init.d/wireguard
rc-update add wireguard
```

WgDashboard installeren

Voor het beheer van onze wireguard server gaan we gebruik maken van WgDashboard: https://github.com/donaldzou/WGDashboard. Dit is een gratis open source project gebaseerd op Python.

We beginnen met alle dependencies te installeren:

```
apk update ; apk upgrade
apk add python3 py3-pip py3-qrcode py3-cffi git
python3 -m ensurepip

pip3 install --upgrade pip setuptools wheel
```

Je kan testen of alles correct verlopen is door even jouw Python en pip versie na te kijken:

```
python3 -V
pip3 -V
```

Als jouw python versie hoger is dan 3.8 is ben je klaar voor de volgende stappen.

Voor wgdashboard gaan we in de `/srv` map een map voorzien waar we alle software gaan plaatsen. dit is de meest correcte en propere manier om te werken. We downloaden ook alle software met git:

```
cd /srv
mkdir wgdashboard
cd wgdashboard
git clone https://github.com/donaldzou/WGDashboard.git .
```

> **Tip:** In dit voorbeeld halen we de `main/latest` branch binnen. Indien je een specifieke stable release wenst kan je deze met `-b <release>` opgeven. Alle releases kan je vinden op https://github.com/donaldzou/WGDashboard/releases

Nadat git alle bestanden heeft binnengehaald kunnen we starten met de installatie van wgdashboard:

```
cd src
chmod +x wgd.sh
./wgd.sh install

-----------------------------------------------------------
-
| Starting to install WGDashboard
|
| Upgrading pip
|
| Installing latest Python dependencies
|
| WGDashboard installed successfully!
|
| Enter ./wgd.sh start to start the dashboard
|
-----------------------------------------------------------
-
```

De installatie duurt even. Als alles klaar is kan je de server opstarten:

```
./wgd.sh start

-----------------------------------------------------------
-
| Starting WGDashboard with Gunicorn in the background.
|
| Log files is under log/
|
-----------------------------------------------------------
-
```

Je kan nu surfen naar jouw Alpine server op poort `10086`. De standaard login is `admin/admin`.

Belangrijk is dat je in de settings jouw **dashboard port aanpast naar 80**. We doen zodat je niet elke keer de poort moet ingeven. De rest van de configuratie kan je zelf naar wens aanpassen.

De laatste stap is het voorzien van een service script voor wgdashboard. Plaats de volgende inhoud in `/etc/init.d/wgdashboard`:

```
#!/sbin/openrc-run

description="WgDashboard"

required_dirs="/etc/wireguard"

depend() {
   need localmount
   need net
   need wireguard
}

start() {
    ebegin "Starting ${RC_SVCNAME}"
    cd /srv/wgdashboard/src
    ./wgd.sh start
    eend $?
}

stop() {
    ebegin "Stopping ${RC_SVCNAME}"
    cd /srv/wgdashboard/src
    ./wgd.sh stop
    eend $?
}
```

Maak dit script uitvoerbaar en zorg er voor dat dit automatisch mee start met Alpine:

```
chmod +x /etc/init.d/wgdashboard
rc-update add wgdashboard
```

UFW

Wanneer je een server opzet die bereikbaar is via het internet (eigenlijk doe je dit best altijd) maak je gebruik van een firewall. Een tool om makkelijk met de linux firewall te werken is UFW (Uncomplicated FireWall). We installeren deze eerst:

```
apk add ufw
```

Onze server maakt gebruik van de volgende poorten en protocollen:

- `22/tcp` voor ssh
- `80/tcp` voor wgdashboard
- `51820/udp` voor wireguard

We stellen onze server zo in dat standaard alle binnenkomende verbindingen behalve wat we nodig hebben geblokkeerd is. Alle buitengaande verbindingen zijn wel ok:

```
ufw default deny incoming
ufw default allow outgoing

ufw allow 22/tcp
ufw allow 80/tcp
ufw allow 51820/udp
```

Het enige wat dan nog rest is je firewall inschakelen en de status opvragen:

```
ufw enable
ufw status

Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
51820/udp                  ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
51820/udp (v6)             ALLOW       Anywhere (v6)
```