

mkosi i systemd-nspawn

La comanda *mkosi* (<https://github.com/systemd/mkosi>) -de "Make Operating System Image"- és una eina per generar imatges (generalment en forma de fitxers) que contenen al seu interior un sistema operatiu mínim, llest per ser arrencat ja sigui en forma de contenidor o màquina virtual.

Concretament, *mkosi* s'encarrega de particionar l'interior de la imatge -atenció, només admet el tipus de taula GPT...això vol dir que dins de la imatge generarà mínim dues particions: la ESP i la partició arrel-, de formatejar aquestes particions internes -el sistema arrel podrà estar, a escollir, en Ext4, Btrfs o també Squashfs (és a dir, només lectura)- i de descarregar-hi allà tots els paquets bàsics necessaris per tenir allà un minisistema operatiu funcional, també a escollir (les possibilitats són Debian, Ubuntu, Fedora, Suse, Arch o CentOS). Depenent de si s'escull generar una imatge arrencable o no (és a dir, preparada per fer-se servir com a màquina virtual autònoma o bé només com a simple contenidor), s'afegirà un gestor d'arranc i un kernel+initrd (ubicats a la partició ESP, muntada sota la carpeta /efi), o no.

La creació pròpiament dita de les imatges dels diferents sistemes operatius *mkosi* delega en eines especialitzades que tenen una llarga història dins la comunitat. Així per exemple, per crear imatges Debian/Ubuntu fa servir l'eina *debootstrap* (o *cdebootstrap* o *grml-debootstrap* o *vmdebootstrap*, segons les circumstàncies); per crear imatges Fedora/CentOS fa servir l'eina *dnf* amb el paràmetre *-installroot*; per crear imatges Arch fa servir l'eina *pacstrap* i per crear imatges Suse fa servir *zypper*. Això vol dir, no obstant, que depén d'on estigui instal·lat *mkosi*, aquest serà capaç de generar imatges de més o menys sistemes diferents: per exemple, a sistemes Debian/Ubuntu no existeix el paquet "dnf", amb la qual cosa serà impossible generar imatges Fedora/CentOS; en canvi, a sistemes Fedora/CentOS sí que existeix el paquet "debootstrap", i, per tant, en aquest cas sí que es pot generar imatges Debian/Ubuntu.

NOTA: Cadascuna d'aquestes eines (debootstrap, etc) ja ve configurada per accedir als repositoris oficials de la distribució corresponent i descarregar d'allà els paquets necessaris per tal d'omplir de contingut les imatges. No obstant, si es desitja, *mkosi* permet escollir explícitament els repositoris a usar gràcies al seu paràmetre *-m* per especificar el servidor-mirror a usar i/o *--repositories* per indicar un repositori concret

A continuació es presenta una taula amb els paràmetres més importants de *mkosi*. Tal com es pot veure allà, una comanda per exemple com...:

sudo mkosi -d debian -r stretch -t gpt_ext4 -b -o imatge.raw -p apt -p iproute2 --password 1234

...crearà un fitxer-imatge anomenat "imatge.raw" contenint un sistema Debian bàsic arrencable amb un usuari "root" amb contrasenya 1234 i amb un parell de paquets instal·lats addicionals.

Si no volguéssim indicar tants paràmetres, tal com es pot veure també a la taula, es pot crear un arxiu anomenat "[mkosi.defaults](#)" a la mateixa carpeta on executarem *mkosi* que contingui una sèrie de seccions i directives que estableixen els valors per defecte per tots els paràmetres que desitgem i així no caldrà especificar-los a mà (en el cas que s'especifiquessin, però, tindrien preferència) ; un exemple seria:

```
[Distribution]
Distribution=debian
Release=stretch
[Output]
Format=gpt_ext4
Bootable=yes
Output=imatge.raw
[Packages]
Packages=apt iproute2
[Validation]
Password=1234
```

NOTA: En qualsevol cas, si un paràmetre no s'especifica enlloc (ni a l'arxiu "mkosi.defaults" ni directament, es pot esbrinar quin és el seu valor per defecte executant la comanda: *mkosi summary*

NOTA: Altres opcions més específiques que no veurem són: *mkosi build*, *mkosi clean*, *mkosi shell*, *mkosi boot* o *mkosi qemu*

Paràmetre	Directiva equivalent a l'arxiu mkiso.defaults	Significat
-d ...	<i>Distribution=...</i> (sota la secció [Distribution])	Indica la distribució a instal·lar. Valors possibles: "fedora", "ubuntu", "debian", "opensuse", "arch", "centos"
-r ...	<i>Release=...</i> (sota la secció [Distribution])	Indica la versió de la distribució. Valors possibles: 28, "stretch", "bionic", etc. Si no s'indica, s'escull el valor més modern (per exemple, a Debian serà "unstable")
-t ...	<i>Format=...</i> (sota la secció [Output])	Indica el format de la imatge a crear. Valors possibles: <u>gpt_ext4</u> : la imatge serà un fitxer dins del qual hi haurà un sistema GPT amb una 1 ^a partició ESP (muntada a /efi) -si la imatge es marca com arrencable- i una 2 ^a partició (arrel del sistema) de tipus Ext4. Equivalent a <u>raw_gpt</u> <u>gpt_xfs</u> : la imatge serà un fitxer dins del qual hi haurà un sistema GPT amb una 1 ^a partició ESP (muntada a /efi) -si la imatge es marca com arrencable- i una 2 ^a partició (arrel del sistema) de tipus XFS. <u>gpt_btrfs</u> : la imatge serà un fitxer dins del qual hi haurà un sistema GPT amb una 1 ^a partició ESP (muntada a /efi) -si la imatge es marca com arrencable- i una 2 ^a partició (arrel del sistema) de tipus Btrfs. Equivalent a <u>raw_btrfs</u> <u>gpt_squashfs</u> : la imatge serà un fitxer dins del qual hi haurà un sistema GPT amb una 1 ^a partició ESP (muntada a /efi) -si la imatge es marca com arrencable- i una 2 ^a partició (arrel del sistema) de tipus Squashfs (és a dir, de només lectura). Equivalent a <u>raw_squashfs</u> <u>directory</u> : la imatge serà una carpeta dins de la qual hi haurà tot el seu contingut del sistema arrel <u>subvolume</u> : la imatge serà un subvolum Btrfs dins del qual hi haurà tot el seu contingut del sistema arrel <u>tar</u> : la imatge serà un arxiu tar dins del qual hi haurà tot el seu contingut del sistema arrel
-b	<i>Bootable=yes</i> (sota la secció [Output])	Fa que la imatge sigui arrencable (en afegir una partició ESP amb un bootloader i, a la partició arrel, una carpeta /boot amb un kernel+initrd). Només funciona pels formats "raw_XXX" però els altres formats poden ser arrencats (només) via <i>systemd-nspawn -b -D</i> NOTA: En aquest cas el "kernel+initrd" és un únic fitxer que inclou tant el kernel en sí com l'arxiu initrd integrat en ell a més dels paràmetres del kernel escrits "hard-coded". Aquest únic fitxer és generat per Dracut en ser invocat per mkosi durant els darrers moments de la seva execució.
-o ...	<i>Output=...</i> (sota la secció [Output])	Indica la ruta i nom de la imatge a crear (per defecte: "image.raw") NOTA: Si s'executa mkosi diverses vegades, no sobre escriurà la imatge generada si ja existia d'abans, a no ser que s'indiqui el paràmetre -f
--root-size n° --esp-size n°	<i>RootSize=n°</i> <i>EspSize=n°</i> (sota la secció [Output])	Estableix el tamany del sist. fitxers arrel (p. defecte 1G) Estableix el tamany de la partició ESP (p. defecte 256M) Només funcionen pels formats "raw_XXX"
-p ...	<i>Packages=...</i> (sota la secció [Packages])	Indica un paquet addicional que s'instal·larà a la imatge. Si es vol afegir més d'un cal indicar un paràmetre -p per cadascun. A l'arxiu de configuració basta amb una sola línia Packages= indicant els diferents paquets un rera l'altre separats per espai en blanc. Paquets a tenir en compte: "iproute2" i "apt/dnf"
--password ...	<i>Password=...</i>	Assigna la contrasenya a l'usuari root.

	(sota la secció [Validation])	Només funciona pels formats "raw_xxx"
--read-only		Fa el sistema arrel de només lectura (raw_squashfs ja és) Només funciona pels formats "raw_xxx"
--hostname ...		Estableix el nom de la màquina dins la imatge
--kernel-commandline ..		Estableix els paràmetres del kernel responsable d'arrencar la imatge (si aquesta és arrencable, és clar)

A la mateixa carpeta on executem mkosi també poden haver opcionalment els següents elements:

*Subcarpeta "mkosi.extra" : Tots els arxius que hi hagi al seu interior es copiaran dins de la partició arrel dins de la imatge, respectant l'arbre de subcarpetes (és a dir, l'arxiu "mkosi.extra/etc/unarxiu.conf" es copiarà sota la carpeta "/etc" del sistema arrel)

*Subcarpeta "mkosi.cache" : Usada com cache dels paquets descarregats, per no haver de tornar a descarregar-los si es repeteix l'execució de mkosi varies vegades

*Script executable "mkosi.postinst" : És invocat com a últim pas a la generació de la imatge i s'executa dins del context del sistema existent a la imatge. Permet poder alterar la imatge "a la carta"

NOTA: Si fos necessari que aquest script tingui accés a la xarxa, caldria afegir el paràmetre `--with-network` a la comanda `mkosi` (o bé afegir la línia `WithNetwork=yes` dins de la secció [Packages] de l'arxiu "mkosi.defaults").

*Fitxer "mkosi.rootpw" : Conté la contrasenya que s'establirà a l'usuari "root" de la imatge (si a l'arxiu apareix un salt de línia al final es treu automàticament, així que no és problema). Aquest arxiu ha de ser propietat de l'usuari "root" de la màquina host i tenir permisos 600 o menys. Si aquest arxiu no apareix i tampoc s'usa el paràmetre `--password`, mkosi establirà la contrasenya per defecte que tingui l'usuari "root" al sistema a construir, la qual normalment sol estar bloquejada).

Un cop generada la imatge, tenim diferents possibilitats d'ús:

*Si no és arrencable (és a dir, si no hem indicat el paràmetre `-b` a `mkosi`), podem entrar-hi fent-la servir com a contenidor Systemd amb la comanda (inclosa dins el paquet "systemd-container"):

systemd-nspawn -i imatge.raw

NOTA: Si volem iniciar Systemd dins del propi contenidor (això té certes ventatges a l'hora de comunicar el contenidor amb la màquina amfitriona) podem fer en canvi: ***systemd-nspawn -bi imatge.raw***

NOTA: Si la imatge fos una carpeta o subvolum en comptes d'un fitxer "raw", caldria executar llavors la comanda ***systemd-nspawn -D carpeta*** (o ***systemd-nspawn -bD carpeta***)

NOTA: L'ús d'imatges mkosi en contenidors LXC/LXD o Docker no està documentat

*Si és arrencable (és a dir, si sí hem indicat el paràmetre `-b` a `mkosi`), podem generar una màquina virtual Qemu amb una comanda similar a aquesta...:

qemu-system-x86_64 -m 512 -enable-kvm
-drive if=pflash,format=raw,readonly,file=/usr/share/OVMF/OVMF_CODE.fd
-drive if=pflash,format=raw,file=~/.copied_OVMF_VARS.fd
-drive if=virtio,format=raw,file=imatge.raw

...o, alternativament, fer servir Libvirt (millor perquè cockpit-machines pot treballar-hi amb ell), així:

virt-install -n nomMaquina -r 512 --disk path=imatge.raw --boot uefi --import

NOTA: És molt recomanable afegir a la comanda `virt-install` anterior el paràmetre `--graphics vnc,listen=0.0.0.0,port=5910` per tal de tenir permanentment activat un servidor VNC mentre la màquina estigui encesa i així poder veure la pantalla de la màquina virtual des d'un client VNC qualsevol

També es pot escriure aquesta imatge arrencable "imatge.raw" directament a un pendrive USB mitjançant `dd` i fer-lo servir com a sistema d'arranc.

Pràctiques *mkosi*, *systemd-nspawn* i *machinectl*

1.-a) Entra en una màquina Fedora que tinguis a mà, converteix-te en "administrador" i crea dins de la carpeta `/root` una subcarpeta anomenada `"mkosi.cache"`. Seguidament, dins d'aquesta carpeta `/root` executa la comanda *mkosi* adient per tal de generar una imatge Fedora 27 arrencable de tipus `"gpt_ext4"`, anomenada `"fedora.raw"`, amb els paquets `"dnf"`, `"nano"`, `"iputils"` i `"iproute"` extra instal·lats i amb un usuari `"root"` que tingui com a contrasenya `"1234"` (atenció, trigarà una estona!)

NOTA: Potser hauràs d'especificar (amb el paràmetre `--root-size`) un tamany de la partició del sistema més gran que l'assignat per defecte per *mkosi* ja que els paquets extra potser no hi cabran i per tant *mkosi* s'interrumpirà amb un error

b) Torna a executar la comanda *mkosi* des de la mateixa carpeta però ara per generar una imatge Ubuntu Bionic no-arrencable (de fet, a dia d'avui encara no se'n poden generar d'arrencables) de tipus `"gpt_ext4"` també, anomenada `"ubuntu.raw"`, amb els paquets `"apt"`, `"nano"`, `"iputils-ping"` i `"iproute2"` extra instal·lats i amb un usuari `"root"` que tingui com a contrasenya `"1234"` (atenció, trigarà una estona!)

2.-a) Executa la comanda *systemd-nspawn* adient per tal d'arrencar la imatge `"ubuntu.raw"` com un contenidor sense Systemd al seu interior. Un cop a dins del contenidor, executa la comanda *uname -r* per confirmar quin és el kernel que està fent servir el contenidor.

b) Llegeix aquest article: https://es.wikipedia.org/wiki/Loop_device i seguidament executa, dins del contenidor, la comanda *lsblk* per veure els dispositius de bloc muntats. Observa i raona en quin dispositiu està muntada la carpeta arrel del sistema. ¿Té coherència amb el que veus en executar *df -hT*?

c) Què és el que veus en fer *ls /* dins del contenidor? I en fer *ls /home*? Per què?

d) Executa la comanda *systemd-detect-virt* dins del contenidor. ¿Per a què serveix?

3.-a) Dins del contenidor anterior observa (amb la comanda *ip*, que haurà d'estar instal·lada) la direcció IP de les diferents tarjes de xarxa i la porta d'enllaç que té el contenidor. ¿Què veus? Prova (amb la comanda *ping*, que haurà d'estar instal·lada) si el contenidor té, efectivament, connexió de xarxa.

NOTA: Que comparteixin les mateixes IPs és una conseqüència de que, per defecte, el contenidor se "n'aprofita" de la configuració de xarxa de la màquina amfitriona. Això, òbviament, podria canviar-se, tal com veurem en propers apartats

b) Surt d'aquest contenidor (ho pots fer pulsant tres vegades seguides `"CTRL+]"` o bé `"CTRL+dret + 5"`) i torna-hi a entrar però ara afegint a la comanda *systemd-nspawn -i ubuntu.raw* un nou paràmetre anomenat `--private-network`. Observa (amb *ip -c a* i *ping*) què li passa a la configuració de xarxa del contenidor (pots consultar també la pàgina del manual de *systemd-nspawn*).

c) Consulta al manual de *systemd-nspawn* per esbrinar què fa, combinat amb el paràmetre `--private-network`, el paràmetre `--network-interface=enp0s8` ¿Per a què creus que podria ser útil aquest paràmetre?

NOTA: Existeixen altres opcions similars a `--network-interface` però que, en comptes de fer "desaparèixer" la tarja de xarxa de la màquina amfitriona per "posar-la" al contenidor, permeten crear dins del contenidor una tarja de xarxa virtual i connectar-la, com si estiguessin en un switch, a una tarja de xarxa determinada la màquina anfitriona Per exemple, `--network-macvlan=enp0s8` o `--network-ipvlan=enp0s8`, entre altres.

d) Sabent que un dispositiu `"veth"` és un conjunt de dues tarjes ethernet virtuals connectades entre si de forma que el que surt per una arriba a l'altra, surt del contenidor i torna-hi a entrar però ara afegint a la comanda *systemd-nspawn -i ubuntu.raw* el paràmetre `--network-veth` (o `-n`, és el mateix) combinat amb el paràmetre `--private-network`. Seguidament executa *ip -c a* tant a la màquina amfitriona com a dins del contenidor. ¿Quines tarjes de xarxa noves veus?

NOTA: If the name of the container is `foo`, the name of the virtual Ethernet interface on the host will be `ve-foo` and the name of the virtual Ethernet interface in the container will be `host0`. When examining the interfaces with `ip link`, interface names will be shown with a suffix, such as `ve-foo@if2` and `host0@if9`. The `@ifN` is not actually part of the name of the interface; instead, `ip link` appends this information to indicate which "slot" the virtual Ethernet cable connects to on the other end. For example, a host virtual Ethernet interface shown as `ve-foo@if2` will connect to container `foo`, and inside the container to the 2nd network interface -- the one shown with index 2 when running `ip link` inside the container. Similarly, in the container, the interface named `host0@if9` will connect to the 9th slot on the host.

dII) Assigna una direcció IP de la mateixa xarxa a la tarjes noves que han aparegut en fer l'apartat anterior, tant a la màquina anfitriona com al contenidor, respectivament. Fes *ping* entre elles. ¿Què passa? D'altra banda, ¿el contenidor té connexió a Internet? Per què?

NOTA: Respecte aquest apartat és molt interessant llegir l'apartat de la pàgina del manual de *systemd-nspawn* que parla sobre l'existència dels fitxers `"/usr/lib/systemd/network/80-container-ve.network"` i `"/usr/lib/systemd/network/80-container-host0.network"` (útils només, no obstant, si tant al contenidor com a la màquina anfitriona funciona *systemd-networkd*)

4.-a) Surt del contenidor utilitzat als exercicis anteriors i executa de nou la comanda *systemd-nspawn* però ara per arrencar la imatge "fedora.raw" amb el seu propi Systemd al seu interior. Un cop a dins del contenidor, executa la comanda *uname -r* per confirmar quin és el kernel que s'està fent servir.

b) Executa dins del contenidor la comanda *lsblk* ¿Quants dispositius "loop" veus ara? ¿On estan muntats cadascun? ¿Per què hi ha aquesta diferència amb el contenidor estudiat als exercicis anteriors?

c) ¿Què és el que veus en fer *ls /efi* dins del contenidor? I més concretament, en fer *ls /efi/Linux*?

5.-a) Mantenint en marxa el contenidor anterior, obre un terminal independent a la màquina anfitriona i executa-hi allà la comanda *machinectl list*. ¿Què veus?

b) ¿Què veus si executes *machinectl status fedora* ?

NOTA: La comanda *machinectl* és capaç de comunicar-se amb els contenidors gràcies a què aquests incorporen un servei especial anomenat D-Bus (a més del propi Systemd), que és automàticament instal·lat a totes les imatges generades per *mkosi*. No obstant, si s'utilitza una altra eina per generar imatges (com ara directament *debootstrap*, *dnf --installroot*, etc), cal tenir això en compte per instal·lar D-Bus (i Systemd) a mà.

c) ¿Què passa si executes *machinectl login fedora*? (has de ser "root" i el contenidor, tal com ja s'ha demanat, ha de tenir funcionant Systemd al seu interior)

NOTA: Si el contenidor és un Fedora, la comanda anterior potser no funciona. Per a què funcioni (i la resta de comandes d'aquest exercici) prèviament has d'executar (com administrador) la comanda *setenforce permissive* , la qual relaxa un mecanisme de seguretat anomenat SELinux responsable de bloquejar la gestió de contenidors per part de *machinectl* i que està activat per defecte a sistemes Fedora

NOTA: Si el contenidor és un Debian/Ubuntu, la comanda anterior potser no funciona. L'explicació tècnica es troba aquí: <https://github.com/systemd/systemd/issues/852#issuecomment-127652768> però bàsicament, el que cal fer és afegir dins de l'arxiu `/etc/securetty` del contenidor una línia que posi *pts/0* (i per si de cas, més línies que posin *pts/1*, *pts/2*, etc)

d) Consulta el manual de *systemctl* per esbrinar per a què serveix el seu paràmetre *-M*. ¿Què passa, per tant, si executes la comanda *systemctl -M fedora status systemd-journald* (has de ser "root")?

NOTA: Molts altres executables de Systemd tenen el paràmetre *-M*, com ara *journalctl*, *hostnamectl*, *systemd-analyze*, etc

e) ¿Què passa si executes *machinectl poweroff fedora* ? (has de ser "root" i el contenidor, tal com ja s'ha demanat, ha de tenir funcionant Systemd al seu interior)

NOTA: Un sinònim d'aquesta comanda és *machinectl stop nomContenidor*. D'altra banda, també existeix la comanda *machinectl terminate nomContenidor*, però aquesta atura el contenidor indicat de forma abrupta. D'altra banda, també existeix la comanda *machinectl reboot nomContenidor*

6.-a) Amb el/s contenidor/s apagat/s, en un terminal de la màquina amfitriona executa la comanda *machinectl list-images* . Per saber per què aquesta comanda no mostra res consulta a la pàgina del manual de *machinectl* què és el que fa realment aquesta comanda i quina importància té en aquest cas la carpeta */var/lib/machines*. Un cop après això, fes el necessari (que pot ser una còpia o la creació d'un enllaç simbòlic) per a què tant "ubuntu.raw" com "fedora.raw" apareguin a la sortida de *machinectl list-images* . No cal que tinguis cap contenidor en marxa per aconseguir-ho.

b) Executa ara *machinectl start fedora* (has de ser "root") i tot seguit *machinectl login fedora* ¿Què veus? Surt del contenidor amb "CTRL+J" i seguidament executa *machinectl status fedora* ¿Continua funcionant el contenidor?

NOTA: Fixa't en la diferència clau entre *systemd-nspawn* i *machinectl start* : la primera inicia un contenidor directament en primer pla i la segona ho fa en segon pla (havent d'utilitzar *machinectl login* per accedir al seu shell). Això fa que la segona sigui molt més adient per posar en marxa serveis dins de contenidors, tal com veurem en els exercicis següents. D'altra banda, *machinectl start* només funcionarà amb contenidors ubicats dins de */var/lib/machines*

NOTA: La comanda *machinectl start fedora* és equivalent a *systemctl start systemd-nspawn@fedora.service*. Igualment, la comanda *machinectl stop fedora* és equivalent a *systemctl stop systemd-nspawn@fedora.service*. Per tant, si volguéssim canviar algun aspecte de la manera d'iniciar els contenidors mitjançant *machinectl start*, caldria editar l'arxiu de configuració del service en qüestió, que és */usr/lib/systemd/system/systemd-nspawn@.service* (allà es poden canviar, per exemple, els paràmetres concrets passats a la comanda *systemd-nspawn* invocada per *machinectl start*, entre altres aspectes. Per exemple, l'ús del paràmetre *-b* -indicat per defecte-)

c) ¿Què passa si executes *machinectl shell root@fedora /bin/ls* ? (has de ser "root")

NOTA: Si el contenidor es va arrencar fent servir directament *systemd-nspawn* en comptes de *machinectl start* , la comanda *machinectl shell* donarà un error similar a "Failed to get shell PTY" degut a què *systemd-nspawn* ja ofereix un shell interactiu per treballar-hi (impedint la connexió de qualsevol altre shell) mentre que amb *machinectl start* no hi ha aquest problema perquè inicia el contenidor en segon pla

NOTA: Si el contenidor no està arrencat, es podria executar la comanda *systemd-nspawn -i fedora.raw /bin/ls* per iniciar-lo, executar la comanda indicada al final (mostrant la seva sortida per pantalla) i parar-lo, tot d'una tacada.

d) ¿Què veus si executes *machinectl image-status fedora* ?

NOTA: Una comanda similar és *machinectl show-image fedora* però està més enfocada a ser "machine-parseable"

e) ¿Què passa si executes *machinectl clone fedora fedora2*? (has de ser "root")

f) ¿Què passa si executes *machinectl rename fedora2 fedoraNou* ? (has de ser "root")

g) ¿Què passaria si executessis *machinectl read-only fedoraNou true*? (has de ser "root")

h) Executa finalment *machinectl stop fedora* i comprova que, efectivament, el contenidor ara ja no estigui funcionant.

i) ¿Per a què serviria la comanda *machinectl enable fedora*? (has de ser "root"). Per saber si funciona hauràs de reiniciar la màquina amfitriona

NOTA: Aquesta comanda és equivalent a *systemctl enable systemd-nspawn@fedora.service*

NOTA: En el cas hipotètic de necessitar iniciar un contenidor tenint un determinat punt de muntatge ja funcional, l'arxiu *.mount* adient haurà de tenir les línies *RequiredBy=systemd-nspawn@fedora.service* i *StopWhenUnneeded=true*

En el cas de voler executar la comanda *machinectl start ...* (o *machinectl enable ...*) és interessant disposar d'algun mètode per poder passar paràmetres personalitzats al contenidor en qüestió (com ara el mode de les seves tarja de xarxa, per exemple), i poder sobreescriure, per tant, els paràmetres per defecte que apareixen a la plantilla *"/usr/lib/systemd/system/systemd-nspawn@.service"*. Per aconseguir això es pot crear un arxiu dins de la carpeta *"/etc/systemd/nspawn"* (o també dins de la carpeta *"/var/lib/machines"* però llavors cal afegir a *systemd-nspawn* el paràmetre *--settings=trusted* per a què el reconegui) anomenat igual que el contenidor i amb extensió *".nspawn"* que tingui un contingut similar a aquest (per a més informació, consultar *man systemd.nspawn*):

[Exec]

Boot=on #Si val "on" és equivalent a haver escrit el paràmetre **-b** de *systemd-nspawn*

Parameters=/bin/ls -l #Equivalent a haver escrit la comanda indicada al final de *systemd-nspawn*

Environment=PATH=/bin #Estableix el valor d'una variable d'entorn al contenidor. Equivalent al paràmetre **--setenv**

LimitXXX=xxx #Estableixen diferents límits (*LimitCPU, LimitFSIZE, LimitNOFile...*) pel contenidor. Equival a **--rlimit**

ResolvConf=copy-host #Estableix com es gestionarà l'arxiu */etc/resolv.conf* del contenidor (si es copiarà de l'amfitrió, si no,...)

[Files]

ReadOnly=yes #Equivalent al paràmetre **--read-only** . Una altra opció similar és *Volatile*

Bind=/opt:/var/opt #Munta la carpeta */opt* de l'amfitrió a la carpeta */var/opt* del contenidor. Equival a **--bind**

BindReadOnly=/opt:/opt #Similar a *Bind* però el muntatge és de només lectura. Equival a **--bind-ro**

[Network]

Private=yes #Equivalent a **--private-network**

Interface=enp0s3 #Equivalent a **--network-interface**

VirtualEthernet=yes #Equivalent a **--network-veth (-n)**. Molt sovint es combina amb **-p** (veure més avall)

MACVLAN=enp0s8 #Equivalent a **--network-macvlan**

IPVLAN=enp0s8 #Equivalent a **--network-ipvlan**

Port=udp:x:y #Reenvia tot el tràfic del port n°x de l'amfitrió al port n° y del contenidor Equivalent a **--port (-p)**

j) Crea dins de la carpeta */etc/systemd/nspawn* un fitxer anomenat *"fedora.nspawn"* que faci que en iniciar el contenidor *"fedora"* amb *machinectl start/enable...*:

- 1.-Automàticament es munti la carpeta */var* de la màquina amfitriona a la carpeta */opt* del contenidor
- 2.-Que el mode de la tarja de xarxa del contenidor sigui de tipus IPVLAN (vinculat a la primera tarja de xarxa que tingui la màquina amfitriona).

...i, un cop iniciat, comprova que, efectivament, la seva carpeta */opt* estigui poblada i que disposi d'una tarja de xarxa (deshabilitada per defecte) anomenada *"iv-nomTarjaAmfitrió"* (pots provar d'activar-la i donar-li una IP a mà de la mateixa xarxa que la de la màquina amfitriona per comprovar que, efectivament, el contenidor començarà a tenir connexió amb l'exterior). Atura el contenidor.

7.-a) Executa la comanda *systemd-nspawn --network-ipvlan=enp0s3 -i fedora.raw* . Un cop dins del contenidor, configura la tarja *"ipvlan"* de la manera que necessitis (activa la tarja, dona-li una IP temporal, assigna una porta d'enllaç temporal, confirma el contingut de l'arxiu */etc/resolv.conf...*) per tal de poder tenir Internet i descarregar i instal·lar el paquet *"nmap-ncat"*. Un cop instal·lat, apaga el contenidor.

b) ¿Com creus que podries evitar-te haver de tornar a fer els passos anteriors cada cop que volguessis tornar a tenir la tarja de xarxa configurada en executar la comanda anterior? Pista: mira l'estat del servei *systemd-networkd* dins del contenidor

c) Executa ara la comanda *systemd-nspawn -p tcp:2000:3000 -n -i fedora.raw* . A partir d'aquí:

NOTA: Si no s'especifica el protocol al paràmetre *-p*, s'assumeix "tcp". Si no s'indica el port del contenidor, s'assumeix que és el mateix que l'indicat per la màquina amfitriona

- 1.-Executa les següents comandes al contenidor:

```
ip link set up dev host0
ip address add 192.168.55.2/24 dev host0
nc -l -p 3000
```

- 2.-Obre un terminal a la màquina amfitriona i executar-hi allà les següents comandes:

```
sudo ip link set up dev ve-fedora
sudo ip address add 192.168.55.1/24 dev ve-fedora
nc 192.168.55.1 2000
```

¿Què passa? Per què?

NOTA: Aquest apartat es podria haver fet amb qualsevol altre servei (Apache2, etc) en comptes del Netcat

d) Atura el contenidor que has posat en marxa a l'apartat anterior. ¿Què li passa al client Netcat de la màquina amfitriona?

Existeix una manera d'iniciar un contenidor (via *machinectl start...*) sense haver de ser root. Resulta que la comanda *machinectl* no és més que un client D-Bus d'un servei anomenat "systemd-machined", el qual és qui realment realitza la feina d'iniciar, parar, etc els contenidors (per tant, la comanda *machinectl* no funcionarà mai si aquest servei està apagat). Doncs bé, resulta que aquest servei utilitza el sistema de seguretat Polkit per autenticar els usuaris que en voler fer ús. Més en concret, al fitxer `/usr/share/polkit-1/actions/org.freedesktop.machine1.policy` trobem les accions concretes (moltes amb una relació directa amb un verb de *machinectl*) a les que podem establir restriccions, com ara "org.freedesktop.machine1.login", "org.freedesktop.machine1.shell", "org.freedesktop.machine1.manage-machines" (útil per *start* i *stop*) o "org.freedesktop.machine1.manage-images", entre d'altres. Així doncs, l'únic que caldrà fer per a què poguem realitzar l'acció escollida sense haver de ser administrador és canviar a "yes" (o "auth_self") el valor de la línia `<allow_active>` associada a aquesta acció (valor que segurament per defecte serà "auth_admin").