Automatiser la création d'images

Table of Contents

Dans un hyperviseur KVM, la création d'images disque et l'installation sur celles-ci peuvent être automatisées à l'aide de:

- virt-install
- packer

Le fichier kickstart

Le fichier **Kickstart** est un fichier texte contenant des mots-clés pour guider l'installation des distributions RHEL. Les sections de ce fichier doivent être indiquées dans l'ordre. Sauf spécification contraire, les éléments contenus dans les sections n'ont pas à être placés dans un ordre spécifique. Tel est l'ordre des sections:

1. La section des commandes
2. La section **%packages**.
3. Les sections **%pre** et **%post** — Ces deux sections n'ont pas à respecter un ordre précis

```
cat > /data/nfsshare/packer/rockylinux/http/8/ks.cfg <<'EOF'
%pre --erroronfail --interpreter=/usr/bin/bash --log=/tmp/kickstart_pre.log
while IFS=$"\n" read line
do
if [[ $line =~ '"u_' ]]; then
params="${line//\"/}"
params="${params//: /=}"
params="${params//,/}"
params="${params// /}"
eval $params;
echo "$params" >> /tmp/ks-network
fi
done < <(curl -s http://xx.xx.xxx.xxx:8023/8/network.json)
echo "=============== ks-network ========================="
cat /tmp/ks-network
```

```
%end

# License agreement
eula --agreed
# Use text mode install
text
# Keyboard layout
keyboard --vckeymap=fr --xlayouts='fr'
# Root password
rootpw --plaintext passw0rd
user --name=vagrant --plaintext --password vagrant
# System language
lang fr_FR.UTF-8
# Network information
network  --bootproto=static --ip=${u_net_ip} --netmask=${u_net_mask} --
gateway=${u_net_gateway} --nameserver=${u_net_dns} --device=link --
hostname=localhost.localdomain --onboot=on --activate
# Firewall configuration
firewall --disabled
# Use network installation
url --url https://dl.rockylinux.org/vault/rocky/8.4/BaseOS/x86_64/os/ --
proxy="http://proxy.infra.dgfip:3128" --noverifyssl
repo --name="Rocky-AppStream" --
baseurl=https://dl.rockylinux.org/vault/rocky/8.4/AppStream/x86_64/os/ --
proxy="http://proxy.infra.dgfip:3128" --install --noverifyssl
repo --name="Rocky-BaseOS" --
baseurl=https://dl.rockylinux.org/vault/rocky/8.4/BaseOS/x86_64/os/ --
proxy="http://proxy.infra.dgfip:3128" --install --noverifyssl
# Do not configure the X Window System
skipx
# Disable Initial Setup on first boot
firstboot --disable
# System timezone
timezone Europe/Paris --isUtc
# SELinux configuration
selinux --enforcing
# System services
services --enabled="cockpit.socket,NetworkManager,sshd"
# System bootloader configuration
bootloader --location=mbr --append="crashkernel=auto net.ifnames=0
biosdevname=0"
# Clear the Master Boot Record
zerombr
# Partition clearing information
clearpart --all --initlabel

# Disk partitioning information
autopart

# Reboot after successful installation
reboot
```

```
%packages --ignoremissing --excludedocs --instLangs=fr_FR.utf8
@core
dnf
kernel
yum
nfs-utils
dnf-utils
grub2-pc
grub2-efi-x64
shim

# Exclude unnecessary firmwares (aic=sas, alsa=audio, ivtv=tv,
iwl+libertas=wifi)
-aic94xx-firmware
-alsa-firmware
-alsa-lib
-alsa-tools-firmware
-ivtv-firmware
-iwl*-firmware
-libertas*-firmware
-fprintd-pam
-intltool
-microcode_ctl

# We need this image to be portable; also, rescue mode isn't useful here.
dracut-config-generic
dracut-norescue

# Needed initially, but removed below.
firewalld

# cherry-pick a few things from @base
tar
tcpdump
rsync

# Some things from @core we can do without in a minimal install
-biosdevname
-plymouth
NetworkManager
-iprutils

# Because we need networking
dhcp-client

# Minimal Cockpit web console
cockpit-ws
cockpit-system
subscription-manager-cockpit

# Exclude all langpacks for now
```

```
-langpacks-*
-langpacks-en
langpacks-fr

# We are building RHEL
rocky-release

# Add rng-tools as source of entropy
rng-tools

# vagrant needs this to copy initial files via scp
openssh-clients
sudo
selinux-policy-devel
wget
nfs-utils
net-tools
tar
bzip2
deltarpm
rsync
dnf-utils
redhat-lsb-core
elfutils-libelf-devel
network-scripts
%end

%post --nochroot --erroronfail --interpreter=/usr/bin/bash --
log=/mnt/sysimage/var/log/kickstart_post_nochroot.log
cat >> /mnt/sysimage/var/log/kickstart_post_nochroot.log << "EOL"
%include /tmp/kickstart_pre.log
EOL
%include /tmp/ks-network
rm -f /mnt/sysimage/etc/sysconfig/network-scripts/ifcfg-*
# simple eth0 config, again not hard-coded to the build hardware
cat > /mnt/sysimage/etc/sysconfig/network-scripts/ifcfg-eth0 <<EOL
DEVICE="eth0"
BOOTPROTO="static"
ONBOOT="yes"
TYPE="Ethernet"
USERCTL="yes"
DNS1="$u_net_dns"
DNS2="10.156.32.33"
GATEWAY="$u_net_gateway"
IPADDR="$u_net_ip"
NETMASK="$u_net_mask"
IPV4_FAILURE_FATAL="yes"
IPV6INIT="no"
NAME="System eth0"
NM_CONTROLLED="yes"
ONBOOT="yes"
```

```
EOL

%end

#
# Add custom post scripts after the base post.
#
%post --erroronfail
# Disable quiet boot and splash screen
sed --follow-symlinks -i "s/ rhgb quiet//" /mnt/sysimage/etc/default/grub
sed --follow-symlinks -i "s/ rhgb quiet//" /mnt/sysimage/boot/grub2/grubenv

# setup systemd to boot to the right runlevel
echo -n "Setting default runlevel to multiuser text mode"
rm -f /etc/systemd/system/default.target
ln -s /lib/systemd/system/multi-user.target
/etc/systemd/system/default.target
echo .

# this is installed by default but we don't need it in virt
echo "Removing linux-firmware package."
dnf -C -y remove linux-firmware

# Remove firewalld; it is required to be present for install/image building.
echo "Removing firewalld."
dnf -C -y remove firewalld --setopt="clean_requirements_on_remove=1"

echo -n "Getty fixes"
# although we want console output going to the serial console, we don't
# actually have the opportunity to login there. FIX.
# we don't really need to auto-spawn _any_ gettys.
sed -i '/^#NAutoVTs=.*/ a\
NAutoVTs=0' /etc/systemd/logind.conf

echo -n "Network fixes"
# initscripts don't like this file to be missing.
cat > /etc/sysconfig/network << EOL
NETWORKING=yes
NOZEROCONF=yes
EOL

# For cloud images, 'eth0' _is_ the predictable device name, since
# we don't want to be tied to specific virtual (!) hardware
rm -f /etc/udev/rules.d/70*
ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules

# set virtual-guest as default profile for tuned
echo "virtual-guest" > /etc/tuned/active_profile

# generic localhost names
cat > /etc/hosts << EOL
```

```
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6

EOL
echo .

cat <<EOL > /etc/sysconfig/kernel
# UPDATEDEFAULT specifies if new-kernel-pkg should make
# new kernels the default
UPDATEDEFAULT=yes

# DEFAULTKERNEL specifies the default kernel package type
DEFAULTKERNEL=kernel
EOL

# make sure firstboot doesn't start
echo "RUN_FIRSTBOOT=NO" > /etc/sysconfig/firstboot


# Disable subscription-manager yum plugins
sed -i 's|^enabled=1|enabled=0|' /etc/yum/pluginconf.d/product-id.conf
sed -i 's|^enabled=1|enabled=0|' /etc/yum/pluginconf.d/subscription-
manager.conf

echo "Cleaning old yum repodata."
dnf clean all

# clean up installation logs"
rm -rf /var/log/yum.log
rm -rf /var/lib/yum/*
rm -rf /root/install.log
rm -rf /root/install.log.syslog
rm -rf /root/anaconda-ks.cfg
rm -rf /var/log/anaconda*

echo "Fixing SELinux contexts."
touch /var/log/cron
touch /var/log/boot.log
mkdir -p /var/cache/yum
/usr/sbin/fixfiles -R -a restore

# remove random-seed so it's not the same every time
rm -f /var/lib/systemd/random-seed

# Remove machine-id on the pre generated images
cat /dev/null > /etc/machine-id

# Anaconda is writing to /etc/resolv.conf from the generating environment.
# The system should start out with an empty file.
```

```
truncate -s 0 /etc/resolv.conf

# Passwordless sudo for the user 'vagrant'
echo 'Defaults:vagrant !requiretty' > /etc/sudoers.d/vagrant
echo '%vagrant ALL=(ALL) NOPASSWD: ALL' >> /etc/sudoers.d/vagrant
chmod 440 /etc/sudoers.d/vagrant
%end
EOF
```

Les options **net.ifnames=0** et **biosdevname=0** dans la commande **bootloader** permettent de désactiver le rennomage des cartes ethernet. Le cartes seront donc présentées au système avec les noms habituels eth0, eth1, ...

La section **%post** est exécutée par défaut dans un chroot, les paramètres de la section **%pre** ne sont donc pas passé d'un environnement à l'autre sans utiliser une astuce.

Il faut charger les paramètres dans la section **%pre**, dans un fichier accessible à cette section, par exemple `/tmp/kickstart_pre.log`

Puis dans une section **%post** non chrooté on peut charger ce fichier dans le répertoire `/mnt/sysyimage/` accessible à la section **%post** chrooté:

```
%post --nochroot --erroronfail --interpreter=/usr/bin/bash --
log=/mnt/sysimage/var/log/kickstart_post_nochroot.log
cat >> /mnt/sysimage/var/log/kickstart_post_nochroot.log << "EOL"
%include /tmp/kickstart_pre.log
EOL
```

On peut alors utiliser ce fichier dans la section **%post** où sont exécutés les scriptes post-installation..

Les fichiers initrd.img et vmlinuz doivent être compatibles avec le kernel de l'iso. Si on utilise l'iso **Rocky-8.4-x86_64** il faut définir les repo `rocky/8.4/BaseOS/x86_64` sinon on obtient une erreur du type: `0000053: kickstart install: Service org.fedoraproject.Anaconda.Modules.Storage has failed to start (timed out)`

Le téléchargement des paquets en http est filtré par l'antivirus, ce qui ralentit trop le processus, il faut donc utiliser les dépôts en https avec l'option **–noverifyssl**

virt-install

virt-install est un outil de ligne de commande permettant de créer de nouveaux invités de conteneur KVM, Xen ou Linux à l'aide de la bibliothèque de gestion d'hyperviseur **libvirt**.

L'outil **virt-install** prend en charge les installations textuelles et graphiques, à l'aide de graphiques VNC ou SDL, ou d'une console série texte. L'invité peut être configuré pour utiliser un ou plusieurs disques virtuels, interfaces réseau, périphériques audio, périphériques physiques USB ou PCI, entre autres.

l'installation depuis **virt-install** utilise l'interface **ovs br-int**, il faut donc utiliser une adresse en dehors du sous-réseau réservé à qemu **xx.xx.xxx.xxx/29** et la passerelle par défaut **yy.yy.yyy.y**.

Le fichier contenant les paramètres réseau, se présente ainsi:

```
cat > http/8/network.json <<'EOF'
{
    "u_net_ip": "xx.xx.xxx.xxx",
    "u_net_mask": "255.255.255.0",
    "u_net_gateway": "yy.yy.yyy.y",
    "u_net_dns": "zz.zzz.zz.zzz",
    "u_net_proxy": "http://proxy.infra.dgfip:3128"
}
EOF
```

Serveur de fichiers

Afin de pouvoir télécharger le fichier **kickstart** on peut utiliser un serveur http tel que **SimpleHTTPServer**. Pour cela il suffit de se placer dans le répertoire à servir puis exécuter la commende `$python -m SimpleHTTPServer <port_number>`:

```
cd /data/nfsshare/packer/rockylinux/http
python -m SimpleHTTPServer 8023

Serving HTTP on 0.0.0.0 port 8023 ...
```

Exécution de virt-install

Les arguments du paramètre **–extra-args** spécifient le paramètre de réseau statique, en utilisant la structure suivante :

```
ip=[ip]::[gateway]:[netmask]:[hostname]:[interface]:[autoconf]
```

```
virt-install --name rocky8-dgfip --ram 2048 --disk
path=/data/container/rocky8-dgfip.qcow2,size=16 --vcpus 1 --os-variant
rocky8.4 --network=bridge:br-int,model=virtio,virtualport_type=openvswitch -
-graphics none --console pty,target_type=serial --location
/data/nfsshare/http/iso/el/8/Rocky-8.4-x86_64-boot.iso --check
path_in_use=off --extra-
args="ip=xx.xx.xxx.xxx::yy.yy.yyy.y:255.255.255.0:test.example.com::none
dns=zz.zzz.zz.zzz inst.ks=http://xx.xx.xxx.xxx:8023/8/ks.cfg console=tty0
console=ttyS0,115200n8"
```

Packer

Packer est un outil développé et maintenu par Hashicorp qui permet de créer des images système automatisées.

Quand on veut déployer une application sur un serveur, il faut souvent installer et configurer beaucoup de services, Packer, peut automatiser tout ça.

Il permet également de créer des images pour différents systèmes de virtualisation.

l'installation depuis packer utilise **qemu-build** il faut donc utiliser l'interface **br-tun** compatible **tun/tap**, les adresses disponibles sont dans la plage entre **xx.xx.xxx.155** et **xx.xx.xxx.158** et la passerelle par défaut est **yy.yy.yyy.y** (adresse de l'interface **br-tun**).

Le fichier contenant les paramètres réseau, se présente ainsi:

```
cat > http/8/network.json <<'EOF'
{
    "u_net_ip": "xx.xx.xxx.xxx",
    "u_net_mask": "255.255.255.0",
    "u_net_gateway": "yy.yy.yyy.y",
    "u_net_dns": "zz.zzz.zz.zzz",
    "u_net_proxy": "http://xx.xx.xxx.xxx:3128"
}
EOF
```

Les scriptes json

La commande `qemu-system-x86_64 -machine help` permet de lister les machines prises en charge, .

```
Supported machines are:
pc                   RHEL 7.6.0 PC (i440FX + PIIX, 1996) (alias of pc-
i440fx-rhel7.6.0)
pc-i440fx-rhel7.6.0  RHEL 7.6.0 PC (i440FX + PIIX, 1996) (default)
pc-i440fx-rhel7.5.0  RHEL 7.5.0 PC (i440FX + PIIX, 1996)
pc-i440fx-rhel7.4.0  RHEL 7.4.0 PC (i440FX + PIIX, 1996)
pc-i440fx-rhel7.3.0  RHEL 7.3.0 PC (i440FX + PIIX, 1996)
pc-i440fx-rhel7.2.0  RHEL 7.2.0 PC (i440FX + PIIX, 1996)
pc-i440fx-rhel7.1.0  RHEL 7.1.0 PC (i440FX + PIIX, 1996)
pc-i440fx-rhel7.0.0  RHEL 7.0.0 PC (i440FX + PIIX, 1996)
q35                  RHEL-8.6.0 PC (Q35 + ICH9, 2009) (alias of pc-q35-
rhel8.6.0)
pc-q35-rhel8.6.0     RHEL-8.6.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel8.5.0     RHEL-8.5.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel8.4.0     RHEL-8.4.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel8.3.0     RHEL-8.3.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel8.2.0     RHEL-8.2.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel8.1.0     RHEL-8.1.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel8.0.0     RHEL-8.0.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel7.6.0     RHEL-7.6.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel7.5.0     RHEL-7.5.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel7.4.0     RHEL-7.4.0 PC (Q35 + ICH9, 2009)
pc-q35-rhel7.3.0     RHEL-7.3.0 PC (Q35 + ICH9, 2009)
none                 empty machine
```

Cela permet de déterminer le type de machine à définir dans **qemuargs**:

```
cat > /data/nfsshare/packer/rockylinux/rockylinux-8.4-Dgfip.json <<'EOF'
{
  "builders": [
    {
      "type": "qemu",
      "iso_url": "/data/nfsshare/http/iso/el/8/Rocky-8.4-x86_64-boot.iso",
      "iso_checksum": "none",
      "output_directory": "output_centos_tdhtest",
      "shutdown_command": "echo 'packer' | sudo -S shutdown -P now",
      "disk_size": "5120M",
      "format": "qcow2",
      "accelerator": "kvm",
      "http_directory": "/data/nfsshare/packer/rockylinux/http",
      "ssh_password": "vagrant",
      "ssh_port": 22,
      "ssh_timeout": "10000s",
      "ssh_username": "vagrant",
      "http_bind_address": "0.0.0.0",
      "vnc_bind_address": "0.0.0.0",
      "http_port_min": "8023",
      "http_port_max": "8023",
      "vm_name": "rocky8",
      "net_device": "virtio-net",
```

```
      "net_bridge": "br-tun",
      "disk_interface": "virtio",
      "boot_wait": "10s",
  "qemuargs": [
    [ "-m", "1024M" ],
    [ "-machine", "q35,accel=kvm"],
    [ "-display", "none" ]
  ],
      "boot_command": [
        "<tab> text ip={{user `u_net_ip`}}::{{user `u_net_gateway`}}:{{user
`u_net_mask`}}:test.example.com::none nameserver={{user `u_net_dns`}}
inst.ks=http://{{ .HTTPIP }}:{{ .HTTPPort }}/8/ks.cfg<enter><wait>"
      ]
    }
  ]
}
EOF
```

Un serveur http est démarré au début de processus, afin que la VM puisse télécharger le fichier **kickstart**. Le port de ce serveur est forcé sur **8023** car il est également utilisé à l'intérieur de **kickstart** dans la section **%pre** afin de récupérer les paramètres d'installation (ip, gateway, dns, proxy)

Il est préférable de mettre à l'écoute sur tous les interfaces les serveurs http `"http_bind_address": "0.0.0.0"`, et vnc `"vnc_bind_address": "0.0.0.0"`, afin de pouvoir les contacter depuis l'extérieur.

Exécution du scripte

Pour exécuter ce scripte il faut se placer dans le répertoire contenant le scripte et le dossier `http` qui contient les fichiers **kickstart** téléchargés lors de l'installation.

```
cd /data/container/packer/rockylinux
packer build -var-file=http/8/network.json rockylinux-8.4-Dgfip.json
```

A la fin du processus une VM est disponible dans le dossier retourné par le message:

```
==> Builds finished. The artifacts of successful builds are:
--> qemu: VM files in directory: output_centos_tdhtest
```

```
ls -alh output_centos_tdhtest/
```

```
total 2,3G
drwxr-xr-x. 2 root root   20  4 juil. 03:27 .
drwxr-xr-x. 5 root root 4,0K  4 juil. 03:16 ..
-rw-r--r--. 1 root root 2,3G  4 juil. 03:27 rocky8
```