How to create custom images

MAAS supports deploying custom OS images. Canonical provides both lp:maas-image-builder and gh:canonical/packer-maas to support creating custom images. These custom images can include static Ubuntu images, created with whatever tool you choose, as well as other OS images. Note that MAAS Image Builder requires the purchase of Ubuntu Advantage support.

While it may be possible to deploy a certain image with MAAS, the particular use case may not be supported by that image's vendor due to licensing or technical reasons. Canonical recommends that, whenever possible, you should customise machines using cloud-init user_data or Curtin preseed data, instead of creating a custom image.

MAAS supports deploying custom DD or TGZ images. Canonical provides both lp:maas-image-builder and gh:canonical/packer-maas to support creating custom images; however, these tools do not currently support Ubuntu. Instead Canonical suggests customising Ubuntu using cloud-init user_data or Curtin preseed data.

This section will help you learn:

- Why customised Ubuntu deployments aren't supported
- Warnings on creating a custom Ubuntu image
- How to create a custom Ubuntu image for MAAS
- How to build MAAS images

Why customised Ubuntu deployments aren't supported

When the MAAS stream is generated by lp:maas-images it starts by downloading the base SquashFS rootfs from cloud-images.ubuntu.com that is used for all clouds. The SquashFS does not contain a kernel so lp:maas-images mounts the SquashFS with an overlay and chroots in. It then installs a kernel and extra initramfs scripts from the cloud-initramfs-rooturl and cloud-initramfs-copymods packages to allow network booting. Once everything is installed the kernel and newly generated initramfs are pulled out of the overlay and everything is unmounted. lp:maas-images provides the unmodified SquashFS, installed kernel, and generated initramfs as separate files on images.maas.io.

MAAS uses the kernel, initramfs, and SquashFS to perform network booting which allows commissioning, testing, and deployments. When deploying Ubuntu MAAS uses the same version of Ubuntu to network boot into and perform the deployment. This ensures there are no compatibility issues. Other operating systems use the Ubuntu version selected for the ephemeral environment for deployment.

Currently MAAS only allows custom images to be loaded as a single TGZ or DD.GZ, there is no way to upload a kernel, `initrd`, and `rootfs` as separate files. Even if MAAS was modified to allow the kernel, `initrd`, and `rootfs` as separate files MAAS requires `cloud-initramfs-copymods` and `cloud-initramfs-rooturl` to be included in the initrd. It is difficult for MAAS to detect if these scripts are missing and even harder for users to debug if they are missing.

Warnings on creating a custom Ubuntu image

Note that:

- Custom images are always deployed with the ephemeral operating system. This can cause hard to debug errors. For example CentOS 6 can only be deployed by Ubuntu Xenial due to advances in the ext4 filesystem.
- MAAS will still install and configure the GA kernel. If your custom image contains a kernel it most likely won't be used. MAAS will not allow you to select which kernel(GA, HWE, lowlatency, etc) when deploying a custom Ubuntu image.
- All GNU/Linux custom images should be created as a TGZ to enable storage customisation. When a DD.GZ is used MAAS/Curtin simply writes the file to the filesystem.

How to create a custom Ubuntu image for MAAS

Note: LXD may prevent device files from being created when extracting the rootfs, it is suggested to do this on metal or on a VM:

- Download the rootfs from images.maas.io
  ```
  wget http://cloud-images.ubuntu.com/focal/current/focal-server-cloudimg-amd64-root.tar.xz
  ```
- Create a work directory
  ```
  mkdir /tmp/work
  ```
- Extract the rootfs
  ```
  cd /tmp/work
  sudo tar xf ../focal-server-cloudimg-amd64-root.tar.xz
  ```

  or

  ```
  unsquashfs ../focal-server-cloudimg-amd64-root.squash
  ```
- Setup chroot
  ```
  sudo mount -o bind /proc /tmp/work/proc
  sudo mount -o bind /dev /tmp/work/dev
  sudo mount -o bind /sys /tmp/work/sys
  sudo mv /tmp/work/etc/resolv.conf /tmp/work/etc/resolv.conf.bak
  sudo cp /etc/resolv.conf /tmp/work/etc/
  ```
- Chroot in
  ```
  sudo chroot /tmp/work /bin/bash
  ```
- Customise image
  ```
  apt update
  apt dist-upgrade
  apt install emacs-nox jq tree
  apt autoremove
  ```
- Exit chroot and unmount binds
  ```
  exit
  sudo umount /tmp/work/proc
  sudo umount /tmp/work/dev
  sudo umount /tmp/work/sys
  sudo mv /tmp/work/etc/resolv.conf.bak /tmp/work/etc/resolv.conf
  ```
- Create TGZ
  ```
  sudo tar -czf ~/focal-custom.tgz -C /tmp/work .
  ```
- Upload it to MAAS
  Note: Ubuntu release names and versions are reserved
  ```
  maas $PROFILE boot-resources create name='custom/focal-custom' title='Ubuntu 20.04 Custom Image' architecture='amd64/generic' filetype='tgz' content@=~/focal-custom.tgz
  ```
- Configure and deploy as normal

How to build MAAS images

There are two methods for building custom images to be deployed to MAAS machines: MAAS Image Builder, and packer. This section will help you learn:

- How to verify packer prequisites

How to verify packer prequisites

The following are required to to create a packer MAAS image:

- A machine running Ubuntu 18.04+ or higher with the ability to run KVM virtual machines.
- Various dependencies required by the chosen template (see the template directory for details)
- Packer

As an example for this article, we will be building a custom Ubuntu image from the default template, which has the following prerequisites:

- qemu-utils
- qemu-system
- ovmf
- cloud-image-utils

Note that these requirements may vary by template and target image.

How to verify packer deployment requirements

The following are required to deploy a packer MAAS image:

- MAAS 3.0+
- Curtin 21.0+

How to customise images

It is possible to customize the image either during the Ubuntu installation, or afterwards (before packing the final image). The former is done by providing autoinstall config, editing the *user-data-flat* and *user-data-lvm* files. The latter is performed by the *install-custom-packages* script.

How to build images via a proxy

The Packer template downloads the Ubuntu net installer from the Internet. To tell Packer to use a proxy, set the HTTP_PROXY environment variable to your proxy server. Alternatively, you may redefine iso_url to a local file, set iso_checksum_type to none to disable the checksums, and remove iso_checksum_url.

How to use packer to build MAAS images

Packer can be used to build images to be deployed by MAAS. In order to use packer, you must have a packer template for the OS version you intend to build.

Currently, templates are available for:

- Ubuntu custom images
- CentOS6
- CentOS7
- CentOS8
- RHEL7
- RHEL8
- VMWare EXSi

This section will help you learn:

- How to install packer
- How to obtain templates
- How to create and use packer images

Note that additional templates will be made available from time to time.

How to install packer

Packer is easily installed from its Debian package:

```
sudo apt install packer
```

For this example Ubuntu template, the following dependencies should also be installed – but note that these dependencies may vary by template and/or target image:

```
sudo apt install qemu-utils
sudo apt install qemu-system
sudo apt install ovmf
sudo apt install cloud-image-utils
```

All of these should install without additional prompts.

How to obtain templates

You can obtain the packer templates by cloning the packer-maas github repository, like this:

```
git clone https://github.com/canonical/packer-maas.git
```

Make sure to pay attention to where the repository is cloned. The Packer template in this cloned repository creates a Ubuntu AMD64 image for use with MAAS.

How to create and use packer images

This subsection will help you learn:

- How to build a packer image
- How to upload packer images to MAAS
- About the default image username

How to build a packer raw image

To build a packer image, you must change to the template repository directory, then to the subdirectory for the image you want to build. From that subdirectory, you can easily build a raw image with LVM, using the Makefile:

```
$ make custom-ubuntu-lvm.dd.gz
```

This makefile will run for a couple of minutes before attempting to boot the image. While waiting for the image to boot, packer will attempt to SSH into the machine repeatedly until it succeeds. You will see terminal messages similar to this one for upwards of three to five minutes:

```
2022/05/09 15:50:46 packer-builder-qemu plugin: [DEBUG] handshaking with SSH
2022/05/09 15:50:50 packer-builder-qemu plugin: [DEBUG] SSH handshake err: ssh: handshake failed: ssh: unable to authenticate, attempted methods [none password], no supported methods remain
2022/05/09 15:50:50 packer-builder-qemu plugin: [DEBUG] Detected authentication error. Increasing handshake attempts.
```

Eventually, you should see a successful SSH connection:

```
2022/05/09 15:50:57 packer-builder-qemu plugin: [INFO] Attempting SSH connection to 127.0.0.1:2351...
2022/05/09 15:50:57 packer-builder-qemu plugin: [DEBUG] reconnecting to TCP connection for SSH
2022/05/09 15:50:57 packer-builder-qemu plugin: [DEBUG] handshaking with SSH
2022/05/09 15:51:17 packer-builder-qemu plugin: [DEBUG] handshake complete!
```

If the process seems to run for a long time, you can predict whether it's going to work by doing a series of `netstat -a` on the `IP:port` given in the connection attempt. Attempts may fail repeatedly, but if you repeat the `netstat -a` command frequently, you will see some tentative connections, like this one:

```
stormrider@neuromancer:~$ netstat -a | grep 2281
```

```
tcp        0      0 0.0.0.0:2281            0.0.0.0:*               LISTEN
tcp        0      0 localhost:46142         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46120         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46138         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46134         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46130         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46124         localhost:2281          TIME_WAIT
stormrider@neuromancer:~$ netstat -a | grep 2281
tcp        0      0 0.0.0.0:2281            0.0.0.0:*               LISTEN
tcp        0      0 localhost:46142         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46146         localhost:2281          ESTABLISHED
tcp        0      0 localhost:2281          localhost:46146         ESTABLISHED
tcp        0      0 localhost:46138         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46134         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46130         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46124         localhost:2281          TIME_WAIT
```

This `ESTABLISHED` connection may not hold the first few times, but eventually, the SSH connection will be made, and the provisioning process will finish. If you want to walk away and come back, be advised that the Makefile clears the terminal buffer at the end, but echoes one final instruction:

```
rm OVMF_VARS.fd
```

You can check the validity of the operation with `ls`, like this:

```
stormrider@neuromancer:~/mnt/Dropbox/src/git/packer-maas/ubuntu$ ls
custom-ubuntu-lvm.dd.gz  packages      seeds-lvm.iso     user-data-lvm
http                     packer_cache  ubuntu-flat.json
Makefile                 README.md     ubuntu-lvm.json
meta-data                scripts       user-data-flat
```

You can also manually run packer. Your current working directory must be in the subdirectory where the packer template is located. In the case of this example, that's `packer-maas/ubuntu`. Once in `packer-maas/ubuntu`, you can generate an image with the following command sequence:

```
$ sudo PACKER_LOG=1 packer build ubuntu-lvm.json
```

ubuntu-lvm.json is configured to run Packer in headless mode. Only Packer output will be seen. If you wish to see the installation output connect to the VNC port given in the Packer output, or change the value of headless to false in the JSON file.

This process is non-interactive.

How to upload packer images to MAAS

You can upload a packer image with the following command:

```
$ maas admin boot-resources create \
    name='custom/ubuntu-raw' \
    title='Ubuntu Custom RAW' \
    architecture='amd64/generic' \
    base_image="ubuntu/focal" \
    filetype='ddgz' \
    content@=custom-ubuntu-lvm.dd.gz
```

Before relying on it in production, you should test your custom image by deploying it to a test (virtual) machine. It's the machine named `open-gannet` in this listing:

```
maas admin machines read | jq -r '(["HOSTNAME","SYSID","POWER","STATUS",
"OWNER", "OS", "DISTRO"] | (., map(length*"-"))),
(.[] | [.hostname, .system_id, .power_state, .status_name, .owner // "-",
.osystem, .distro_series]) | @tsv' | column -t

HOSTNAME     SYSID   POWER  STATUS    OWNER  OS      DISTRO
--------     -----   -----  ------    -----  --      ------
valued-moth  e86c7h  on     Deployed  admin  ubuntu  focal
open-gannet  nk7x8y  on     Deployed  admin  custom  ubuntu-raw
```

About the default image username

The default username for packer-created images is `ubuntu`, the same as the default username for other MAAS images.

How to pack an Ubuntu image for MAAS deployment

Here's how to build a deployable Ubuntu image with packer:

Install packer

Packer is easily installed from its Debian package:

```
sudo apt install packer
```

This should install with no additional prompts.

Install Ubuntu template dependencies

```
sudo apt install qemu-utils
sudo apt install qemu-system
sudo apt install ovmf
sudo apt install cloud-image-utils
```

All of these should install with no additional prompts.

Get the available packer templates

You can obtain the packer templates by cloning the packer-maas github repository, like this:

```
git clone https://github.com/canonical/packer-maas.git
```

Make sure to pay attention to where the repository is cloned. The Packer template in this cloned repository creates a Ubuntu AMD64 image for use with MAAS.

Build an Ubuntu raw image with packer

To build a packer image, you must change to the template repository directory, then to the subdirectory for the image you want to build. From that subdirectory, you can easily build a raw image with LVM, using the Makefile:

```
$ make custom-ubuntu-lvm.dd.gz
```

This makefile will run for a couple of minutes before attempting to boot the image. While waiting for the image to boot, packer will attempt to SSH into the machine repeatedly until it succeeds. You will see terminal messages similar to this one for upwards of three to five minutes:

```
2022/05/09 15:50:46 packer-builder-qemu plugin: [DEBUG] handshaking with SSH
2022/05/09 15:50:50 packer-builder-qemu plugin: [DEBUG] SSH handshake err: ssh: handshake failed: ssh: unable to authenticate, attempted methods [none password], no supported methods remain
2022/05/09 15:50:50 packer-builder-qemu plugin: [DEBUG] Detected authentication error. Increasing handshake attempts.
```

Eventually, you should see a successful SSH connection:

```
2022/05/09 15:50:57 packer-builder-qemu plugin: [INFO] Attempting SSH connection to 127.0.0.1:2351...
2022/05/09 15:50:57 packer-builder-qemu plugin: [DEBUG] reconnecting to TCP connection for SSH
2022/05/09 15:50:57 packer-builder-qemu plugin: [DEBUG] handshaking with SSH
2022/05/09 15:51:17 packer-builder-qemu plugin: [DEBUG] handshake complete!
```

If the process seems to run for a long time, you can predict whether it's going to work by doing a series of `netstat -a` on the `IP:port` given in the connection attempt. Attempts may fail repeatedly, but if you repeat the `netstat -a` command frequently, you will see some tentative connections, like this one:

```
stormrider@neuromancer:~$ netstat -a | grep 2281
tcp        0      0 0.0.0.0:2281            0.0.0.0:*               LISTEN
tcp        0      0 localhost:46142         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46120         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46138         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46134         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46130         localhost:2281          TIME_WAIT
```

```
tcp        0      0 localhost:46124         localhost:2281          TIME_WAIT
stormrider@neuromancer:~$ netstat -a | grep 2281
tcp        0      0 0.0.0.0:2281            0.0.0.0:*               LISTEN
tcp        0      0 localhost:46142         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46146         localhost:2281          ESTABLISHED
tcp        0      0 localhost:2281          localhost:46146         ESTABLISHED
tcp        0      0 localhost:46138         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46134         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46130         localhost:2281          TIME_WAIT
tcp        0      0 localhost:46124         localhost:2281          TIME_WAIT
```

This `ESTABLISHED` connection may not hold the first few times, but eventually, the SSH connection will be made, and the provisioning process will finish. If you want to walk away and come back, be advised that the Makefile clears the terminal buffer at the end, but echoes one final instruction:

```
rm OVMF_VARS.fd
```

Validate the build

You can check the validity of the operation with `ls`, like this:

```
stormrider@neuromancer:~/mnt/Dropbox/src/git/packer-maas/ubuntu$ ls
custom-ubuntu-lvm.dd.gz  packages      seeds-lvm.iso     user-data-lvm
http                     packer_cache  ubuntu-flat.json
Makefile                 README.md     ubuntu-lvm.json
meta-data                scripts       user-data-flat
```

Alternative: Run packer manually

You can also manually run packer. Your current working directory must be in the subdirectory where the packer template is located. In the case of this example, that's `packer-maas/ubuntu`. Once in `packer-maas/ubuntu`, you can generate an image with the following command sequence:

```
$ sudo PACKER_LOG=1 packer build ubuntu-lvm.json
```

ubuntu-lvm.json is configured to run Packer in headless mode. Only Packer output will be seen. If you wish to see the installation output connect to the VNC port given in the Packer output, or change the value of headless to false in the JSON file.

This process is non-interactive.

Upload the Ubuntu image to MAAS

You can upload an Ubuntu raw packer image with the following command:

```
$ maas admin boot-resources create \
    name='custom/ubuntu-raw' \
    title='Ubuntu Custom RAW' \
    architecture='amd64/generic' \
    filetype='ddgz' \
    content@=custom-ubuntu-lvm.dd.gz
```

Verify your custom image

Before relying on it in production, you should test your custom image by deploying it to a test (virtual) machine. It's the machine named `open-gannet` in this listing:

```
maas admin machines read | jq -r '(["HOSTNAME","SYSID","POWER","STATUS",
"OWNER", "OS", "DISTRO"] | (., map(length*"-"))),
(.[] | [.hostname, .system_id, .power_state, .status_name, .owner // "-",
.osystem, .distro_series]) | @tsv' | column -t

HOSTNAME     SYSID   POWER  STATUS    OWNER  OS      DISTRO
--------     -----   -----  ------    -----  --      ------
valued-moth  e86c7h  on     Deployed  admin  ubuntu  focal
open-gannet  nk7x8y  on     Deployed  admin  custom  ubuntu-raw
```

Log into your deployed image and verify that it's right

You should log into your newly-deployed image and verify that it has all the customisations you added to the build process. The default username for packer-created images is `ubuntu`, the same as the default username for other MAAS images.

How to pack a RHEL7 image for MAAS deployment

You can create a RHEL7 image for MAAS deployment via the following procedure.

Verify the requirements

To create a RHEL7 image to upload to MAAS, you must have a machine running Ubuntu 18.04+ or higher with the ability to run KVM virtual machines. You must also aquire the following items, as described in later instructions:

- qemu-utils
- packer
- The RHEL 7 DVD ISO

To deploy the image to MAAS, you must also have:

- MAAS 2.3+
- Curtin 18.1-59+

Install packer

Packer is easily installed from its Debian package:

```
sudo apt install packer
```

This should install with no additional prompts.

Install RHEL7 template dependencies

```
sudo apt install qemu-utils
```

Get the available packer templates

You can obtain the packer templates by cloning the packer-maas github repository, like this:

```
git clone https://github.com/canonical/packer-maas.git
```

Make sure to pay attention to where the repository is cloned. The Packer template in this cloned repository creates a RHEL7 AMD64 image for use with MAAS.

This package should install with no additional prompts.

Locate the RHEL7 template

The packer template in the directory `rhel7` subdirectory creates a RHEL 7 AMD64 image for use with MAAS.

Obtain the RHEL7 DVD ISO

Download the RHEL7 DVD ISO to your `rhel7` subdirectory.

You may have to scroll down the page at the above link to find the RHEL7 version. **Be sure you are getting the ISO image** (a much larger file) and not the bootstrap file.

Customizing the image, if desired

The deployment image may be customized by modifying http/rhel7.ks. See the CentOS kickstart documentation for more information.

Set up a proxy for building the image, if desired

The Packer template pulls all packages from the DVD except for Canonical's cloud-init repository. To use a proxy during the installation add the --proxy=$HTTP_PROXY flag to every line starting with url or repo in http/rhel7.ks. Alternatively you may set the --mirrorlist values to a local mirror.

Build the RHEL7 image

You can easily build the image using the Makefile:

```
$ make ISO=/PATH/TO/rhel-server-7.9-x86_64-dvd.iso
```

Almost immediately, the terminal will prompt you for your user password (to access `sudo` rights). Then, for a few minutes, you will encounter a stream `qemu` text similar to this:

```
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu path: /usr/bin/qemu-system-x86_64, Qemu Image page: /usr/bin/qemu-img
==> qemu: Retrieving ISO
2022/06/07 13:28:00 packer-builder-qemu plugin: Leaving retrieve loop for ISO
2022/06/07 13:28:00 packer-builder-qemu plugin: No floppy files specified. Floppy disk will not be made.
2022/06/07 13:28:00 packer-builder-qemu plugin: No CD files specified. CD disk will not be made.
2022/06/07 13:28:00 packer-builder-qemu plugin: [INFO] Creating disk with Path: output-qemu/packer-qemu and Size: 4G
2022/06/07 13:28:00 packer-builder-qemu plugin: Executing qemu-img: []string{"create", "-f", "qcow2", "output-qemu/packer-qemu", "4G"}
2022/06/07 13:28:00 packer-builder-qemu plugin: stdout: Formatting 'output-qemu/packer-qemu', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=4294967296 lazy_refcounts=off refcount
2022/06/07 13:28:00 packer-builder-qemu plugin: stderr:
2022/06/07 13:28:00 packer-builder-qemu plugin: Found available port: 8144 on IP: 0.0.0.0
==> qemu: Starting HTTP server on port 8144
    qemu: No communicator is set; skipping port forwarding setup.
==> qemu: Looking for available port between 5900 and 6000 on 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Looking for available port between 5900 and 6000 on 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Found available port: 5970 on IP: 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Found available VNC port: 5970 on IP: 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu --version output: QEMU emulator version 6.2.0 (Debian 1:6.2+dfsg-2ubuntu6)
2022/06/07 13:28:00 packer-builder-qemu plugin: Copyright (c) 2003-2021 Fabrice Bellard and the QEMU Project developers
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu version: 6.2.0
==> qemu: Starting VM, booting from CD-ROM
    qemu: view the screen of the VM, connect via VNC without a password to
    qemu: vnc://127.0.0.1:70
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu Builder has no floppy files, not attaching a floppy.
    qemu: The VM will be run headless, without a GUI. If you want to
    qemu: view the screen of the VM, connect via VNC without a password to
    qemu: vnc://127.0.0.1:70
2022/06/07 13:28:00 packer-builder-qemu plugin: Executing /usr/bin/qemu-system-x86_64: []string{"-netdev", "user,id=user.0", "-vnc", "127.0.0.1:70", "-serial", "stdio", "-device", "virtio-net,netdev=user
==> qemu: Overriding default Qemu arguments with qemuargs template option...
2022/06/07 13:28:00 packer-builder-qemu plugin: Started Qemu. Pid: 7970
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu stderr: qemu-system-x86_64: warning: host doesn't support requested feature: CPUID.80000001H:ECX.svm [bit 2]
==> qemu: Waiting 3s for boot...
==> qemu: Connecting to VM via VNC (127.0.0.1:5970)
2022/06/07 13:28:05 packer-builder-qemu plugin: Connected to VNC desktop: QEMU (packer-qemu)
==> qemu: Typing the boot command over VNC...
2022/06/07 13:28:05 packer-builder-qemu plugin: Special code '<up>' found, replacing with: 0xFF52
2022/06/07 13:28:06 packer-builder-qemu plugin: Special code '<tab>' found, replacing with: 0xFF09
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char ' ', code 0x20, shift false
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char 'i', code 0x69, shift false
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char 'n', code 0x6E, shift false
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char 's', code 0x73, shift false
2022/06/07 13:28:07 packer-builder-qemu plugin: Sending char 't', code 0x74, shift false
2022/06/07 13:28:07 packer-builder-qemu plugin: Sending char '.', code 0x2E, shift false
```

Eventually, the screen will clear and you will see an `anaconda` window. Anaconda is the RedHat installer tool, which is preparing your custom image:

```
7) [x] Network configuration            8) [ ] User creation
       (Wired (ens3) connected)                (No user will be created)
2022/06/07 13:29:19 packer-builder-qemu plugin: Qemu stdout:
================================================================================
================================================================================
Progress07 13:29:19 packer-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:19 packer-builder-qemu plugin: Qemu stdout:
.022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: Setting up the installation environment
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: Setting up com_redhat_kdump addon
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: Setting up org_fedora_oscap addon
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: ..
2022/06/07 13:29:23 packer-builder-qemu plugin: Qemu stdout: Configuring storage
...2/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout:
Running pre-installation scriptser-qemu plugin: Qemu stdout:
.022/06/07 13:29:23 packer-builder-qemu plugin: Qemu stdout:
Running pre-installation taskslder-qemu plugin: Qemu stdout:
...2/06/07 13:29:24 packer-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:24 packer-builder-qemu plugin: Qemu stdout: Installing.
2022/06/07 13:29:25 packer-builder-qemu plugin: Qemu stdout: Starting package installation process
Downloading packagespacker-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:29 packer-builder-qemu plugin: Qemu stdout:

[anaconda]1:main* 2:shell  3:log  4:storage-log >Switch tab: Alt+Tab | Help: F1 :shell  3:log  4:sto><'echo -n "Switch tab: Alt+Tab | Help: F
```

Anaconda will run for three to five minutes. When it finishes, it will clear the screen and return you to the shell prompt.

Alternative: Run packer manually

Alternatively, you can manually run packer. Your current working directory must be in `packer-maas/rhel7`, where this file is located. Once in `packer-maas/rhel7`, you can generate an image with:

```
$ sudo PACKER_LOG=1 packer build -var 'rhel7_iso_path=/PATH/TO/rhel-server-7.9-x86_64-dvd.iso' rhel7.json
```

`rhel7.json` is configured to run Packer in headless mode. Only Packer output will be seen. If you wish to see the installation output, connect to the VNC port given in the Packer output or change the value of headless to false in `rhel7.json`.

Upload the RHEL7 image to MAAS

You can upload the RHEL7 raw packer image with the following command:

```
$ maas $PROFILE boot-resources create \
    name='rhel/7-custom' \
    title='RHEL 7 Custom' \
    architecture='amd64/generic' \
    base_image='rhel/rhel7' \
    filetype='tgz' \
    content@=rhel7.tar.gz
```

Verify your custom image

Before relying on it in production, you should test your custom image by deploying it to a test (virtual) machine. It's the machine named `open-gannet` in this listing:

```
maas admin machines read | jq -r '(["HOSTNAME","SYSID","POWER","STATUS",
"OWNER", "OS", "DISTRO"] | (., map(length*"-"))),
(.[] | [.hostname, .system_id, .power_state, .status_name, .owner // "-",
.osystem, .distro_series]) | @tsv' | column -t

HOSTNAME     SYSID   POWER  STATUS    OWNER  OS      DISTRO
--------     -----   -----  ------    -----  --      ------
valued-moth  e86c7h  on     Deployed  admin  ubuntu  focal
open-gannet  nk7x8y  on     Deployed  admin  custom  rhel7-raw
```

Log into your deployed image and verify that it's right

You should log into your newly-deployed image and verify that it has all the customisations you added to the build process. The default username for packer-created RHEL images is `cloud-user`.

How to pack a RHEL8 image for MAAS deployment

You can create a RHEL8 image for MAAS deployment via the following procedure.

Verify the requirements

To create a RHEL8 image to upload to MAAS, you must have a machine running Ubuntu 18.04+ or higher with the ability to run KVM virtual machines. You must also aquire the following items, as described in later instructions:

- qemu-utils
- packer
- The RHEL 8 DVD ISO

To deploy the image to MAAS, you must also have:

- MAAS 2.3+
- Curtin 18.1-59+

Install packer

Packer is easily installed from its Debian package:

```
sudo apt install packer
```

This should install with no additional prompts.

Install RHEL8 template dependencies

```
sudo apt install qemu-utils
```

Get the available packer templates

You can obtain the packer templates by cloning the packer-maas github repository, like this:

```
git clone https://github.com/canonical/packer-maas.git
```

Make sure to pay attention to where the repository is cloned. The Packer template in this cloned repository creates a RHEL8 AMD64 image for use with MAAS.

This package should install with no additional prompts.

Locate the RHEL8 template

The packer template in the directory `rhel8` subdirectory creates a RHEL 8 AMD64 image for use with MAAS.

Obtain the RHEL8 DVD ISO

Download the RHEL8 DVD ISO to your `rhel8` subdirectory.

You may have to scroll down the page at the above link to find the RHEL8 version. **Be sure you are getting the ISO image** (a much larger file) and not the bootstrap file.

Customizing the image, if desired

The deployment image may be customized by modifying http/rhel8.ks. See the CentOS kickstart documentation for more information.

Set up a proxy for building the image, if desired

The Packer template pulls all packages from the DVD except for Canonical's cloud-init repository. To use a proxy during the installation add the --proxy=$HTTP_PROXY flag to every line starting with url or repo in http/rhel8.ks. Alternatively you may set the --mirrorlist values to a local mirror.

Build the RHEL8 image

You can easily build the image using the Makefile:

```
$ make ISO=/PATH/TO/rhel-server-8.6-x86_64-dvd.iso
```

Almost immediately, the terminal will prompt you for your user password (to access `sudo` rights). Then, for a few minutes, you will encounter a stream `qemu` text similar to this:

```
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu path: /usr/bin/qemu-system-x86_64, Qemu Image page: /usr/bin/qemu-img
==> qemu: Retrieving ISO
==> qemu: Trying ./rhel-8.6-x86_64-dvd.iso
2022/06/07 13:28:00 packer-builder-qemu plugin: Acquiring lock for: ./rhel-8.6-x86_64-dvd.iso (/home/stormrider/mnt/Dropbox/src/git/packer-maas/rhel8/packer_cache/4142f50427ed611570687aee099b796c00359ae6
==> qemu: Trying ./rhel-8.6-x86_64-dvd.iso
2022/06/07 13:28:00 packer-builder-qemu plugin: Leaving retrieve loop for ISO
==> qemu: ./rhel-8.6-x86_64-dvd.iso => /home/stormrider/mnt/Dropbox/src/git/packer-maas/rhel8/rhel-8.6-x86_64-dvd.iso
2022/06/07 13:28:00 packer-builder-qemu plugin: No floppy files specified. Floppy disk will not be made.
2022/06/07 13:28:00 packer-builder-qemu plugin: No CD files specified. CD disk will not be made.
2022/06/07 13:28:00 packer-builder-qemu plugin: [INFO] Creating disk with Path: output-qemu/packer-qemu and Size: 4G
2022/06/07 13:28:00 packer-builder-qemu plugin: Executing qemu-img: []string{"create", "-f", "qcow2", "output-qemu/packer-qemu", "4G"}
2022/06/07 13:28:00 packer-builder-qemu plugin: stdout: Formatting 'output-qemu/packer-qemu', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=4294967296 lazy_refcounts=off refcoun
2022/06/07 13:28:00 packer-builder-qemu plugin: stderr:
2022/06/07 13:28:00 packer-builder-qemu plugin: Found available port: 8144 on IP: 0.0.0.0
==> qemu: Starting HTTP server on port 8144
    qemu: No communicator is set; skipping port forwarding setup.
==> qemu: Looking for available port between 5900 and 6000 on 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Looking for available port between 5900 and 6000 on 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Found available port: 5970 on IP: 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Found available VNC port: 5970 on IP: 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu --version output: QEMU emulator version 6.2.0 (Debian 1:6.2+dfsg-2ubuntu6)
2022/06/07 13:28:00 packer-builder-qemu plugin: Copyright (c) 2003-2021 Fabrice Bellard and the QEMU Project developers
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu version: 6.2.0
==> qemu: Starting VM, booting from CD-ROM
    qemu: view the screen of the VM, connect via VNC without a password to
    qemu: vnc://127.0.0.1:70
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu Builder has no floppy files, not attaching a floppy.
    qemu: The VM will be run headless, without a GUI. If you want to
    qemu: view the screen of the VM, connect via VNC without a password to
    qemu: vnc://127.0.0.1:70
2022/06/07 13:28:00 packer-builder-qemu plugin: Executing /usr/bin/qemu-system-x86_64: []string{"-netdev", "user,id=user.0", "-vnc", "127.0.0.1:70", "-serial", "stdio", "-device", "virtio-net,netdev=user
==> qemu: Overriding default Qemu arguments with qemuargs template option...
2022/06/07 13:28:00 packer-builder-qemu plugin: Started Qemu. Pid: 7970
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu stderr: qemu-system-x86_64: warning: host doesn't support requested feature: CPUID.80000001H:ECX.svm [bit 2]
==> qemu: Waiting 3s for boot...
==> qemu: Connecting to VM via VNC (127.0.0.1:5970)
2022/06/07 13:28:05 packer-builder-qemu plugin: Connected to VNC desktop: QEMU (packer-qemu)
==> qemu: Typing the boot command over VNC...
2022/06/07 13:28:05 packer-builder-qemu plugin: Special code '<up>' found, replacing with: 0xFF52
2022/06/07 13:28:06 packer-builder-qemu plugin: Special code '<tab>' found, replacing with: 0xFF09
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char ' ', code 0x20, shift false
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char 'i', code 0x69, shift false
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char 'n', code 0x6E, shift false
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char 's', code 0x73, shift false
2022/06/07 13:28:07 packer-builder-qemu plugin: Sending char 't', code 0x74, shift false
2022/06/07 13:28:07 packer-builder-qemu plugin: Sending char '.', code 0x2E, shift false
```

Eventually, the screen will clear and you will see an `anaconda` window. Anaconda is the RedHat installer tool, which is preparing your custom image:

```
7) [x] Network configuration            8) [ ] User creation
       (Wired (ens3) connected)                (No user will be created)
2022/06/07 13:29:19 packer-builder-qemu plugin: Qemu stdout:
================================================================================
================================================================================
Progress07 13:29:19 packer-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:19 packer-builder-qemu plugin: Qemu stdout:
.022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: Setting up the installation environment
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: Setting up com_redhat_kdump addon
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: Setting up org_fedora_oscap addon
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: ..
2022/06/07 13:29:23 packer-builder-qemu plugin: Qemu stdout: Configuring storage
...2/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout:
Running pre-installation scriptser-qemu plugin: Qemu stdout:
.022/06/07 13:29:23 packer-builder-qemu plugin: Qemu stdout:
Running pre-installation taskslder-qemu plugin: Qemu stdout:
```

```
...2/06/07 13:29:24 packer-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:24 packer-builder-qemu plugin: Qemu stdout: Installing.
2022/06/07 13:29:25 packer-builder-qemu plugin: Qemu stdout: Starting package installation process
Downloading packagespacker-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:29 packer-builder-qemu plugin: Qemu stdout:

[anaconda]1:main* 2:shell  3:log  4:storage-log >Switch tab: Alt+Tab | Help: F1 :shell  3:log  4:sto><'echo -n "Switch tab: Alt+Tab | Help: F
```

Anaconda will run for three to five minutes. When it finishes, it will clear the screen and return you to the shell prompt.

Alternative: Run packer manually

Alternatively, you can manually run packer. Your current working directory must be in `packer-maas/rhel8`, where this file is located. Once in `packer-maas/rhel8`, you can generate an image with:

```
$ sudo PACKER_LOG=1 packer build -var 'rhel8_iso_path=/PATH/TO/rhel-server-8.6-x86_64-dvd.iso' rhel8.json
```

`rhel8.json` is configured to run Packer in headless mode. Only Packer output will be seen. If you wish to see the installation output, connect to the VNC port given in the Packer output or change the value of headless to false in `rhel8.json`.

Upload the RHEL8 image to MAAS

You can upload the RHEL8 raw packer image with the following command:

```
$ maas $PROFILE boot-resources create \
    name='rhel/8-custom' \
    title='RHEL 8 Custom' \
    base_image='rhel/rhel8' \
    architecture='amd64/generic' \
    filetype='tgz' \
    content@=rhel8.tar.gz
```

Verify your custom image

Before relying on it in production, you should test your custom image by deploying it to a test (virtual) machine. It's the machine named `open-gannet` in this listing:

```
maas admin machines read | jq -r '(["HOSTNAME","SYSID","POWER","STATUS",
"OWNER", "OS", "DISTRO"] | (., map(length*"-"))),
(.[] | [.hostname, .system_id, .power_state, .status_name, .owner // "-",
.osystem, .distro_series]) | @tsv' | column -t

HOSTNAME     SYSID   POWER  STATUS    OWNER  OS      DISTRO
--------     -----   -----  ------    -----  --      ------
valued-moth  e86c7h  on     Deployed  admin  ubuntu  focal
open-gannet  nk7x8y  on     Deployed  admin  custom  rhel8-raw
```

Log into your deployed image and verify that it's right

You should log into your newly-deployed image and verify that it has all the customisations you added to the build process. The default username for packer-created RHEL images is `cloud-user`.

How to pack a CentOS 7 image for MAAS deployment

You can create a CentOS 7 image for MAAS deployment via the following procedure.

Verify the requirements

To create a CentOS 7 image to upload to MAAS, you must have a machine running Ubuntu 18.04+ or higher with the ability to run KVM virtual machines. You must also aquire the following items, as described in later instructions:

- qemu-utils
- packer

Note that a suitable CentOS 7 image is downloaded as part of the template.

To deploy the image to MAAS, you must also have:

- MAAS 2.3+
- Curtin 18.1-59+

Install packer

Packer is easily installed from its Debian package:

```
sudo apt install packer
```

This should install with no additional prompts.

Install CentOS 7 template dependencies

```
sudo apt install qemu-utils
```

Get the available packer templates

You can obtain the packer templates by cloning the packer-maas github repository, like this:

```
git clone https://github.com/canonical/packer-maas.git
```

Make sure to pay attention to where the repository is cloned. The Packer template in this cloned repository creates a CentOS 7 AMD64 image for use with MAAS.

This package should install with no additional prompts.

Locate the CentOS 7 template

The packer template in the directory `centos7` subdirectory creates a CentOS 7 AMD64 image for use with MAAS.

Customise the image, if desired

The deployment image may be customized by modifying http/centos7.ks. See the CentOS kickstart documentation for more information.

Set up a proxy for building the image, if desired

The Packer template downloads the CentOS net installer from the Internet. To tell Packer to use a proxy set the HTTP_PROXY environment variable to your proxy server. Alternatively you may redefine iso_url to a local file, set iso_checksum_type to none to disable checksuming, and remove iso_checksum_url.

To use a proxy during the installation add the --proxy=$HTTP_PROXY flag to every line starting with url or repo in http/centos7.ks. Alternatively you may set the --mirrorlist values to a local mirror.

Build the CentOS 7 image

You can easily build the image using the Makefile:

```
$ make
```

For a few minutes, you will encounter a stream `qemu` text similar to this:

```
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu path: /usr/bin/qemu-system-x86_64, Qemu Image page: /usr/bin/qemu-img
==> qemu: Retrieving ISO
2022/06/07 13:28:00 packer-builder-qemu plugin: Leaving retrieve loop for ISO
2022/06/07 13:28:00 packer-builder-qemu plugin: No floppy files specified. Floppy disk will not be made.
2022/06/07 13:28:00 packer-builder-qemu plugin: No CD files specified. CD disk will not be made.
2022/06/07 13:28:00 packer-builder-qemu plugin: [INFO] Creating disk with Path: output-qemu/packer-qemu and Size: 4G
2022/06/07 13:28:00 packer-builder-qemu plugin: Executing qemu-img: []string{"create", "-f", "qcow2", "output-qemu/packer-qemu", "4G"}
2022/06/07 13:28:00 packer-builder-qemu plugin: stdout: Formatting 'output-qemu/packer-qemu', fmt=qcow2 cluster_size=65536 extended_l2=off compression_type=zlib size=4294967296 lazy_refcounts=off refcoun
2022/06/07 13:28:00 packer-builder-qemu plugin: stderr:
2022/06/07 13:28:00 packer-builder-qemu plugin: Found available port: 8144 on IP: 0.0.0.0
```

```
==> qemu: Starting HTTP server on port 8144
    qemu: No communicator is set; skipping port forwarding setup.
==> qemu: Looking for available port between 5900 and 6000 on 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Looking for available port between 5900 and 6000 on 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Found available port: 5970 on IP: 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Found available VNC port: 5970 on IP: 127.0.0.1
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu --version output: QEMU emulator version 6.2.0 (Debian 1:6.2+dfsg-2ubuntu6)
2022/06/07 13:28:00 packer-builder-qemu plugin: Copyright (c) 2003-2021 Fabrice Bellard and the QEMU Project developers
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu version: 6.2.0
==> qemu: Starting VM, booting from CD-ROM
    qemu: view the screen of the VM, connect via VNC without a password to
    qemu: vnc://127.0.0.1:70
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu Builder has no floppy files, not attaching a floppy.
    qemu: The VM will be run headless, without a GUI. If you want to
    qemu: view the screen of the VM, connect via VNC without a password to
    qemu: vnc://127.0.0.1:70
2022/06/07 13:28:00 packer-builder-qemu plugin: Executing /usr/bin/qemu-system-x86_64: []string{"-netdev", "user,id=user.0", "-vnc", "127.0.0.1:70", "-serial", "stdio", "-device", "virtio-net,netdev=user
==> qemu: Overriding default Qemu arguments with qemuargs template option...
2022/06/07 13:28:00 packer-builder-qemu plugin: Started Qemu. Pid: 7970
2022/06/07 13:28:00 packer-builder-qemu plugin: Qemu stderr: qemu-system-x86_64: warning: host doesn't support requested feature: CPUID.80000001H:ECX.svm [bit 2]
==> qemu: Waiting 3s for boot...
==> qemu: Connecting to VM via VNC (127.0.0.1:5970)
2022/06/07 13:28:05 packer-builder-qemu plugin: Connected to VNC desktop: QEMU (packer-qemu)
==> qemu: Typing the boot command over VNC...
2022/06/07 13:28:05 packer-builder-qemu plugin: Special code '<up>' found, replacing with: 0xFF52
2022/06/07 13:28:06 packer-builder-qemu plugin: Special code '<tab>' found, replacing with: 0xFF09
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char ' ', code 0x20, shift false
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char 'i', code 0x69, shift false
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char 'n', code 0x6E, shift false
2022/06/07 13:28:06 packer-builder-qemu plugin: Sending char 's', code 0x73, shift false
2022/06/07 13:28:07 packer-builder-qemu plugin: Sending char 't', code 0x74, shift false
2022/06/07 13:28:07 packer-builder-qemu plugin: Sending char '.', code 0x2E, shift false
```

Eventually, the screen will clear and you will see an `anaconda` window. Anaconda is the RedHat installer tool, which is preparing your custom image:

```
7) [x] Network configuration            8) [ ] User creation
       (Wired (ens3) connected)                (No user will be created)
2022/06/07 13:29:19 packer-builder-qemu plugin: Qemu stdout:
================================================================================
================================================================================
Progress07 13:29:19 packer-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:19 packer-builder-qemu plugin: Qemu stdout:
.022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: Setting up the installation environment
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: Setting up com_redhat_kdump addon
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: Setting up org_fedora_oscap addon
2022/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout: ..
2022/06/07 13:29:23 packer-builder-qemu plugin: Qemu stdout: Configuring storage
...2/06/07 13:29:20 packer-builder-qemu plugin: Qemu stdout:
Running pre-installation scriptser-qemu plugin: Qemu stdout:
.022/06/07 13:29:23 packer-builder-qemu plugin: Qemu stdout:
Running pre-installation taskslder-qemu plugin: Qemu stdout:
...2/06/07 13:29:24 packer-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:24 packer-builder-qemu plugin: Qemu stdout: Installing.
2022/06/07 13:29:25 packer-builder-qemu plugin: Qemu stdout: Starting package installation process
Downloading packagespacker-builder-qemu plugin: Qemu stdout:
2022/06/07 13:29:29 packer-builder-qemu plugin: Qemu stdout:

[anaconda]1:main* 2:shell  3:log  4:storage-log >Switch tab: Alt+Tab | Help: F1 :shell  3:log  4:sto><'echo -n "Switch tab: Alt+Tab | Help: F
```

Anaconda will run for three to five minutes. When it finishes, it will clear the screen and return you to the shell prompt.

Alternative: Run packer manually

Alternatively you can manually run packer. Your current working directory must be in packer-maas/centos7, where this file is located. Once in packer-maas/centos7 you can generate an image with:

```
$ sudo PACKER_LOG=1 packer build centos7.json
```

centos7.json is configured to run Packer in headless mode. Only Packer output will be seen. If you wish to see the installation output connect to the VNC port given in the Packer output or change the value of headless to false in centos7.json.

Installation is non-interactive.

Upload the CentOS 7 image to MAAS

You can upload the CentOS 7 raw packer image with the following command:

```
$ maas $PROFILE boot-resources create
name='centos/7-custom' title='CentOS 7 Custom' architecture='amd64/generic' base_image='centos/centos7' filetype='tgz' content@=centos7.tar.gz
```

Verify your custom image

Before relying on it in production, you should test your custom image by deploying it to a test (virtual) machine. It's the machine named `open-gannet` in this listing:

```
maas admin machines read | jq -r '(["HOSTNAME","SYSID","POWER","STATUS",
"OWNER", "OS", "DISTRO"] | (., map(length*"-"))),
(.[] | [.hostname, .system_id, .power_state, .status_name, .owner // "-",
.osystem, .distro_series]) | @tsv' | column -t

HOSTNAME     SYSID   POWER  STATUS    OWNER  OS      DISTRO
--------     -----   -----  ------    -----  --      ------
valued-moth  e86c7h  on     Deployed  admin  ubuntu  focal
open-gannet  nk7x8y  on     Deployed  admin  custom  centos7-raw
```

Log into your deployed image and verify that it's right

You should log into your newly-deployed image and verify that it has all the customisations you added to the build process. The default username for packer-created CentOS 7 images is `centos`.

How to pack an ESXi image for MAAS deployment

You can create an ESXi image for MAAS deployment via the following procedure. MAAS cannot directly deploy the VMware ESXi ISO; a specialized image must be created from the ISO. Canonical has created a Packer template to automatically do this for you.

Verify the requirements and accept the limitations

VMware ESXi has a specific set of requirements and limitations which are more stringent than MAAS.

Basic requirements

The machine building the deployment image must be a GNU/Linux host with a dual core x86_64 processor supporting hardware virtualization with at least 4GB of RAM and 10GB of disk space available. Additionally the qemu-kvm and qemu-utils packages must be installed on the build system.

libvirt testing

While VMware ESXi does not support running in any virtual machine it is possible to deploy to one. The libvirt machine must be a KVM instance with at least CPU 2 cores and 4GB of RAM. To give VMware ESXi access to hardware virtualization go into machine settings, CPUs, and select 'copy host CPU configuration.'

VMware ESXi has no support for libvirt drivers. Instead an emulated IDE disk and an emulated e1000 NIC must be used.

Known storage limitations

Only datastores may be configured using the devices available on the system. The first 9 partitions of the disk are reserved for VMware ESXi operating system usage.

WARNING: VMware does not support cloning boot devices - you may run into issues triggered by non-unique UUID. This may lead to data corruption on VMFS datastores when using cloned boot devices.

Known networking limitations

Bridges are not supported in VMware ESXi. In addition, certain MAAS bond modes are mapped to VMware ESXi NIC team sharing with load balancing, as follows:

- balance-rr - portid
- active-backup - explicit
- 802.3ad - iphash, LACP rate and XMIT hash policy settings are ignored.

No other bond modes are currently supported.

WARNING: VMware ESXi does not allow VMs to use a PortGroup that has a VMK attached to it. All configured devices will have a VMK attached. To use a vSwitch with VMs you must leave a device or alias unconfigured in MAAS.

Image fails to build due to qemu-nbd error

If the image fails to build due to a `qemu-nbd` error, try disconnecting the device with:

```
$ sudo qemu-nbd -d /dev/nbd4
```

Prerequisites to create and deploy images

- A machine running Ubuntu 18.04+ with the ability to run KVM virtual machines.
- qemu-utils
- Python Pip
- Packer
- The VMware ESXi installation ISO must be downloaded manually.
- MAAS 2.5 or above, MAAS 2.6 required for storage configuration

Install packer

Packer is easily installed from its Debian package:

```
sudo apt install packer
```

This should install with no additional prompts.

Install ESXi template dependencies

```
sudo apt install qemu-utils
```

Install Python `pip`, if not installed

```
sudo apt install pip
```

Get the available packer templates

You can obtain the packer templates by cloning the packer-maas github repository, like this:

```
git clone https://github.com/canonical/packer-maas.git
```

Make sure to pay attention to where the repository is cloned. This package should install with no additional prompts.

Locate the ESXi template

The appropriate packer template can be found in the subdirectory `vmware-esxi` in the packer repository.

Customise the image, if desired

The deployment image may be customized by modifying `packer-maas/vmware-esxi/KS.CFG` see Installation and Upgrade Scripts in the VMware ESXi installation and Setup manual for more information.

Build the ESXi image

You can easily build the image using the Makefile:

```
$ make ISO=/path/to/VMware-VMvisor-Installer-6.7.0.update03-14320388.x86_64.iso
```

Alternative: Run packer manually

Alternatively, you can manually run packer. Your current working directory must be in `packer-maas/vmware-esxi`, where this file is located. Once in `packer-maas/vmware-esxi`, you can generate an image with:

```
$ sudo PACKER_LOG=1 packer build -var 'vmware_esxi_iso_path=/path/to/VMware-VMvisor-Installer-6.7.0.update03-14320388.x86_64.iso' vmware-esxi.json
```

`vmware-esxi.json` is configured to runpPacker in headless mode. Only packer output will be seen. If you wish to see the installation outputj, connect to the VNC port given in the packer output, or remove the line containing “headless” in `vmware-esxi.json`.

Installation is non-interactive.

Upload the ESXi image to MAAS

You can upload the ESXi image to MAAS with the following command:

```
$ maas $PROFILE boot-resources create name='esxi/6.7' title='VMware ESXi 6.7' architecture='amd64/generic' filetype='ddgz' content@=vmware-esxi.dd.gz
```

How to use MAAS Image Builder to build MAAS images

MAAS Image Builder is an older tool, still required to build some images (e.g., Windows images). Wherever possible, we recommend you use `packer`, as described above.

In order to use MAAS Image Builder, you must purchase Ubuntu Advantage for Infrastructure.

This article will help you learn:

- How to install MAAS Image Builder via a private Canonical PPA (which you can request)
- How to create custom CentOS images
- How to create custom RHEL images
- How to create Windows images
- How to create other kinds of custom images

You can customise most images as much or as little as you wish, then use them to commission machines with MAAS.

How to install MAAS Image Builder

To get MAAS Image Builder, you must be subscribed to a private PPA provided by Canonical Support to those customers who have purchased Ubuntu Advantage for Infrastructure. Note that the steps below will fail if you have not purchased Ubuntu Advantage and been subscribed to the private PPA by your Canonical support rep.

Once subscribed, you need to obtain your credentials at this external link:

launchpad.net

OpenID transaction in progress

Also, you must add the repository with the `add-apt-repository` command. Note: Be sure to substitute your unique URL in the command below:

```
$ sudo add-apt-repository \
“https://LaunchpadID:Password@private-ppa.launchpad.net/maas-image-builder-partners/stable/ubuntu"
```

Once you have added the private PPA, you can install the Image Builder like this:

```
$ sudo apt-get install maas-image-builder
```

All done? Great! Now you can build and customise images for MAAS machines, as shown in the sections below.

How to create custom CentOS images

MAAS already provides the latest available CentOS 7 and CentOS 8 for automatic download. If you need something else, though, MAAS Image Builder supports the ability to create various CentOS images.

Network Requirements

Access to the Internet is required, since you will need to start with one of these sites:

- http://mirror.centos.org - OS, updates, and extra repositories
- https://download.fedoraproject.org - EPEL
- https://copr-be.cloud.fedoraproject.org - Canonical maintained cloud-init repository

Creating images behind a proxy

MAAS Image Builder can create CentOS images behind a proxy – just set the 'http_proxy' environment variable to *your* particular proxy. Once deployed, `yum` will use this MAAS-configured proxy.

Creating the images

`maas-image-builder` is designed to automate the process of generating the images for MAAS and `curtin`. Here are some specific examples:

```
$ sudo maas-image-builder -o centos6-amd64-root-tgz --arch amd64 centos --edition 6
$ sudo maas-image-builder -o centos6-i386-root-tgz --arch i386 centos --edition 6
$ sudo maas-image-builder -o centos7-amd64-root-tgz --arch amd64 centos --edition 7
```

Customising CentOS images

Starting from MAAS Image Builder 1.0.4, customisation of CentOS images is now supported. You can provide a custom kickstart, in *addition* to the kickstart that MAAS Image Builder uses to create the images. You can customise your image like this:

```
$ sudo maas-image-builder -o centos7-amd64-root-tgz --arch amd64 centos --edition 7 --custom-kickstart ./custom.ks
```

Uploading the image into MAAS

Custom CentOS images can be uploaded to MAAS as shown in the command below. *Do note* that the name **must** start with 'centos' and **must** be one line:

```
maas admin boot-resources create name=centos/centos6-custom architecture=amd64/generic content@=./build-output/centos-amd64-root-tgz
```

You can use the MAAS WebUI to check that your custom CentOS image is valid and selectable for deployment.

How to create custom RHEL images

Currently, MAAS *only* supports RHEL as a custom image. In future versions of MAAS, RHEL will be natively supported.

Some Requirements

In order to create RHEL images, you will need access to these sites:

- A RHEL DVD ISO - Contains all RHEL archives which are not available without a license
- https://download.fedoraproject.org - Access to the EPEL repository to install required deps
- https://copr-be.cloud.fedoraproject.org - Access to the Canonical maintained cloud-init copr repository

Creating images behind a proxy

MAAS image builder supports creating RHEL images behind a proxy. To use a proxy when building a RHEL image, just set the 'http_proxy' environment variable to *your* local proxy. Once deployed, `yum` will use the MAAS-configured proxy.

Creating the images

To generate a usable RHEL image, `maas-image-builder` automates image generation; these images can be used by MAAS and `curtin`.

```
 $ sudo maas-image-builder -a amd64 -o rhel8-amd64-root-tgz rhel --rhel-iso blah.iso
```

Install into MAAS

The custom RHEL image can be uploaded to MAAS, but note that the name **must** start with 'rhel' and **must** be expressed as a single line, like this:

```
maas admin boot-resources create name=rhel/8 title="RedHat Enterprise Linux 8" architecture=amd64/generic content@=rhel8-amd64-root-tgz
```

How to create Windows images

Since Windows is a proprietary operating system, MAAS can't download these images. You need to manually generate images to use with MAAS. On the upside, the end result will be much simpler, since there are CLI and WebUI tools to upload a Windows ISO – which *helps* automate the process.

Hyper-V 2012 R2
In this example, Windows Hyper-V 2012 R2 is used, but rest assured that multiple versions are supported. Download the Hyper-V 2012 R2 ISO from Microsoft, so it can be used in the image generation process. You can obtain the download at:

http://technet.microsoft.com/en-US/evalcenter/dn205299.aspx

There are several other supported versions (for --windows-edition):

- win2012
- win2012r2
- win2012hv
- win2012hvr2
- win2016
- win2016hv

Image Builder

MAAS Image builder can automate the process of generating images for MAAS and `curtin`. In this instance, though, you need Windows drivers for your specific hardware. You can obtain these windows drivers with the following command:

```
sudo maas-image-builder -o windows-win2012hvr2-amd64-root-dd  windows \
--windows-iso ~/Downloads/2012hvr2.ISO \
--windows-edition win2012hvr2 [--windows-updates] \
[--windows-drivers <folder/>]  [--windows-language en-US]
```

As an example, consider:

```
sudo maas-image-builder -o windows-win2012hvr2-amd64-root-dd windows \
--windows-iso win2012hvr2.iso  --windows-edition win2012hvr2 \
--windows-updates --windows-drivers ~/Win2012hvr2_x64/DRIVERS/  --windows-language en-US
```

Please note that this will not *install* any Windows updates. In order to obtain an up-to-date image of Windows, be sure provide the `–windows-updates` flag. This requires access to a bridged connection with a DHCP server, an interface that can be specified with `-i`.

Also note that you may be required to have specific Windows drivers for this image to work in your hardware. Be sure you inject those drivers when installing them. Those drivers are the default `*.inf` files.

Debug

You can debug the Windows installation process by connecting to `localhost:5901` using a VNC client.

Install into MAAS

The generated images need to be placed into the correct directories so MAAS can deploy them onto a node:

```
maas admin boot-resources create name=windows/win2012hvr2 \
architecture=amd64/generic filetype=ddtgz \
content@=./build-output/windows-win2012hvr2-amd64-root-dd
```

Now, using the MAAS WebUI, a node can be selected to use Windows Hyper-V 2012 R2. This selection gets reset when a node is stopped, so make sure to set it *before* starting nodes. You can also set the default operating system (and release) in the settings menu, which removes the need to set it per-node.

How to create other kinds of custom images

To install other custom images, use the following command sequence:

```
maas <user> boot-resources create name=<custom-image-codename> title="Ubuntu Custom Image" \
architecture=amd64/generic content@=/location/of/custom/image/ubuntu-custom-root-tgz
```

As an example:

```
maas admin boot-resources create name=custom1 \
title="Ubuntu Custom Image" architecture=amd64/generic \
content@=/home/ubuntu/ubuntu-custom-root-tgz
```

Last updated 26 days ago.

Help improve this document in the forum.