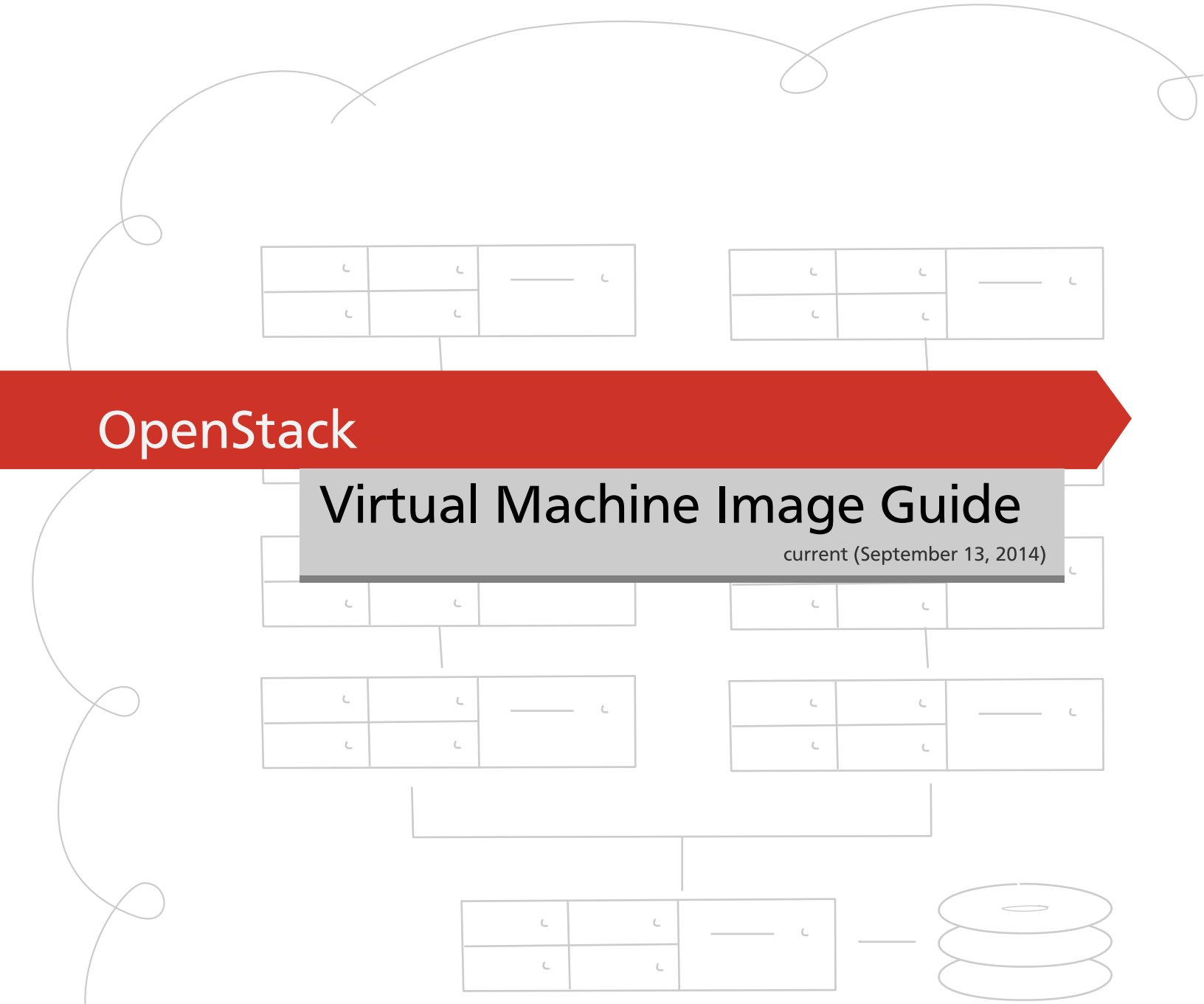


OpenStack

Virtual Machine Image Guide

current (September 13, 2014)

docs.openstack.org

OpenStack Virtual Machine Image Guide

current (2014-09-13)

This guide describes how to obtain, create, and modify virtual machine images that are compatible with OpenStack.

Table of Contents

List of Tables

Preface

Conventions

The OpenStack documentation uses several typesetting conventions.

Notices

Notices take these forms:

Note

A handy tip or reminder.

Important

Something you must be aware of before proceeding.

Warning

Critical information about the risk of data loss or security issues.

Command prompts

$ prompt Any user, including the `root` user, can run commands that are prefixed with the $ prompt.

 # prompt The `root` user must run commands that are prefixed with the # prompt. You can also prefix these commands with the **sudo** command, if available, to run them.

Document change history

This version of the guide replaces and obsoletes all earlier versions.

The following table describes the most recent changes:

Revision Date	Summary of Changes
April 17, 2014	• Minor revisions for Icehouse - moved property listing into *Command-Line Interface Reference* and added a note about Windows time zones.
October 25, 2013	• Adds information about image formats, properties.
October 17, 2013	• Havana release.
June 4, 2013	• Updated title for consistency.
May 28, 2013	• Initial release of this guide.

1. Introduction

An OpenStack Compute cloud is not very useful unless you have virtual machine images (which some people call "virtual appliances"). This guide describes how to obtain, create, and modify virtual machine images that are compatible with OpenStack.

To keep things brief, we'll sometimes use the term "image" instead of "virtual machine image".

What is a virtual machine image?

A virtual machine image is a single file which contains a virtual disk that has a bootable operating system installed on it.

Virtual machine images come in different formats, some of which are described below. In a later chapter, we'll describe how to convert between formats.

Raw The "raw" image format is the simplest one, and is natively supported by both KVM and Xen hypervisors. You can think of a raw image as being the bit-equivalent of a block device file, created as if somebody had copied, say, `/dev/sda` to a file using the **dd** command.

Note

We don't recommend creating raw images by dd'ing block device files, we discuss how to create raw images later.

qcow2 The qcow2 (QEMU copy-on-write version 2) format is commonly used with the KVM hypervisor. It has some additional features over the raw format, such as:

- Using sparse representation, so the image size is smaller
- Support for snapshots

Because qcow2 is sparse, it's often faster to convert a raw image to qcow2 and upload it then to upload the raw file.

Note

Because raw images don't support snapshots, OpenStack Compute will automatically convert raw image files to qcow2 as needed.

AMI/AKI/ARI The AMI/AKI/ARI format was the initial image format supported by Amazon EC2. The image consists of three files:

- AMI (Amazon Machine Image):

  This is a virtual machine image in raw format, as described above.

- AKI (Amazon Kernel Image)

  A kernel file that the hypervisor will load initially to boot the image. For a Linux machine, this would be a *vmlinuz* file.

- ARI (Amazon Ramdisk Image)

  An optional ramdisk file mounted at boot time. For a Linux machine, this would be an *initrd* file.

UEC tarball A UEC (Ubuntu Enterprise Cloud) tarball is a gzipped tarfile that contains an AMI file, AKI file, and ARI file.

Note

Ubuntu Enterprise Cloud refers to a discontinued Eucalyptus-based Ubuntu cloud solution that has been replaced by the OpenStack-based Ubuntu Cloud Infrastructure.

VMDK VMware's ESXi hypervisor uses the VMDK (Virtual Machine Disk) format for images.

VDI VirtualBox uses the VDI (Virtual Disk Image) format for image files. None of the OpenStack Compute hypervisors support VDI directly, so you will need to convert these files to a different format to use them with OpenStack.

VHD Microsoft Hyper-V uses the VHD (Virtual Hard Disk) format for images.

VHDX The version of Hyper-V that ships with Microsoft Server 2012 uses the newer VHDX format, which has some additional features over VHD such as support for larger disk sizes and protection against data corruption during power failures.

OVF OVF (Open Virtualization Format) is a packaging format for virtual machines, defined by the Distributed Management Task Force (DMTF) standards group. An OVF package contains one or more image files, a .ovf XML metadata file that contains information about the virtual machine, and possibly other files as well.

An OVF package can be distributed in different ways. For example, it could be distributed as a set of discrete files, or as a tar archive file with an .ova (open virtual appliance/application) extension.

OpenStack Compute does not currently have support for OVF packages, so you will need to extract the image file(s) from an OVF package if you wish to use it with OpenStack.

ISO The ISO format is a disk image formatted with the read-only ISO 9660 (also known as ECMA-119) filesystem commonly used for CDs and DVDs. While we don't normally think of ISO as a virtual machine image format, since ISOs contain bootable filesystems with an installed operating sys-

tem, you can treat them the same as you treat other virtual machine image files.

Disk and container formats for images

When you add an image to the Image Service, you can specify its disk and container formats.

Disk formats

The disk format of a virtual machine image is the format of the underlying disk image. Virtual appliance vendors have different formats for laying out the information contained in a virtual machine disk image.

Set the disk format for your image to one of the following values:

- `raw`. An unstructured disk image format; if you have a file without an extension it is possibly a raw format
- `vhd`. The VHD disk format, a common disk format used by virtual machine monitors from VMware, Xen, Microsoft, VirtualBox, and others
- `vmdk`. Common disk format supported by many common virtual machine monitors
- `vdi`. Supported by VirtualBox virtual machine monitor and the QEMU emulator
- `iso`. An archive format for the data contents of an optical disc, such as CD-ROM.
- `qcow2`. Supported by the QEMU emulator that can expand dynamically and supports Copy on Write
- `aki`. An Amazon kernel image.
- `ari`. An Amazon ramdisk image.
- `ami`. An Amazon machine image.

Container formats

The container format indicates whether the virtual machine image is in a file format that also contains metadata about the actual virtual machine.

Note

The Image Service and other OpenStack projects do not currently support the container format. It is safe to specify `bare` as the container format if you are unsure.

You can set the container format for your image to one of the following values:

- `bare`. The image does not have a container or metadata envelope.
- `ovf`. The OVF container format.

- `aki`. An Amazon kernel image.
- `ari`. An Amazon ramdisk image.
- `ami`. An Amazon machine image.

Image metadata

Image metadata can help end users determine the nature of an image, and is used by associated OpenStack components and drivers which interface with the Image Service.

Metadata can also determine the scheduling of hosts. If the `property` option is set on an image, and Compute is configured so that the `ImagePropertiesFilter` scheduler filter is enabled (default), then the scheduler only considers compute hosts that satisfy that property.

Note

Compute's `ImagePropertiesFilter` value is specified in the `scheduler_default_filter` value in the `/etc/nova/nova.conf` file.

You can add metadata to Image Service images by using the `--property key=value` option with the **glance image-create** or **glance image-update** command. More than one property can be specified. For example:

```
$ glance image-update img-uuid --property architecture=arm --property
 hypervisor_type=qemu
```

Common image properties are also specified in the `/etc/glance/schema-image.json` file. For a complete list of valid property keys and values, refer to the *OpenStack Command-Line Reference*.

All associated properties for an image can be displayed using the **glance image-show** command. For example:

```
$ glance image-show myCirrosImage
+------------------------------------
+-------------------------------------+
| Property                            | Value
 |
+------------------------------------
+-------------------------------------+
| Property 'base_image_ref'           | 397e713c-b95b-4186-ad46-6126863ea0a9
 |
| Property 'image_location'           | snapshot
 |
| Property 'image_state'              | available
 |
| Property 'image_type'               | snapshot
 |
| Property 'instance_type_ephemeral_gb' | 0
 |
| Property 'instance_type_flavorid'   | 2
 |
| Property 'instance_type_id'         | 5
 |
```

```
| Property 'instance_type_memory_mb'    | 2048
 |
| Property 'instance_type_name'         | m1.small
 |
| Property 'instance_type_root_gb'      | 20
 |
| Property 'instance_type_rxtx_factor'  | 1
 |
| Property 'instance_type_swap'         | 0
 |
| Property 'instance_type_vcpu_weight'  | None
 |
| Property 'instance_type_vcpus'        | 1
 |
| Property 'instance_uuid'              | 84c6e57d-a6b1-44b6-81eb-fcb36afd31b5
 |
| Property 'kernel_id'                  | df430cc2-3406-4061-b635-a51c16e488ac
 |
| Property 'owner_id'                   | 66265572db174a7aa66eba661f58eb9e
 |
| Property 'ramdisk_id'                 | 3cf852bd-2332-48f4-9ae4-7d926d50945e
 |
| Property 'user_id'                    | 376744b5910b4b4da7d8e6cb483b06a8
 |
| checksum                              | 8e4838effa1969ad591655d6485c7ba8
 |
| container_format                      | ami
 |
| created_at                            | 2013-07-22T19:45:58
 |
| deleted                               | False
 |
| disk_format                           | ami
 |
| id                                    | 7e5142af-1253-4634-bcc6-89482c5f2e8a
 |
| is_public                             | False
 |
| min_disk                              | 0
 |
| min_ram                               | 0
 |
| name                                  | myCirrosImage
 |
| owner                                 | 66265572db174a7aa66eba661f58eb9e
 |
| protected                             | False
 |
| size                                  | 14221312
 |
| status                                | active
 |
| updated_at                            | 2013-07-22T19:46:42
 |
+---------------------------------------
+---------------------------------------+
```

2. Get images

The simplest way to obtain a virtual machine image that works with OpenStack is to download one that someone else has already created.

CirrOS (test) images

CirrOS is a minimal Linux distribution that was designed for use as a test image on clouds such as OpenStack Compute. You can download a CirrOS image in various formats from the CirrOS Launchpad download page.

If your deployment uses QEMU or KVM, we recommend using the images in qcow2 format. The most recent 64-bit qcow2 image as of this writing is cirros-0.3.2-x86_64-disk.img

Note

In a CirrOS image, the login account is `cirros`. The password is `cubswin:)`

Official Ubuntu images

Canonical maintains an official set of Ubuntu-based images.

Images are arranged by Ubuntu release, and by image release date, with "current" being the most recent. For example, the page that contains the most recently built image for Ubuntu 14.04 "Trusty Tahr" is http://cloud-images.ubuntu.com/trusty/current/. Scroll to the bottom of the page for links to images that can be downloaded directly.

If your deployment uses QEMU or KVM, we recommend using the images in qcow2 format. The most recent version of the 64-bit QCOW2 image for Ubuntu 14.04 is trusty-server-cloudimg-amd64-disk1.img.

Note

In an Ubuntu cloud image, the login account is `ubuntu`.

Official Red Hat Enterprise Linux images

Red Hat maintains official Red Hat Enterprise Linux cloud images. A valid Red Hat Enterprise Linux subscription is required to download these images:

- Red Hat Enterprise Linux 7 KVM Guest Image
- Red Hat Enterprise Linux 6 KVM Guest Image

Note

In a RHEL image, the login account is `cloud-user`.

Official Fedora images

The Fedora project maintains a list of official cloud images at http://cloud.fedoraproject.org/. The images include the `cloud-init` utility to support key and user data injection. The default user name is `fedora`.

Note

In a Fedora image, the login account is `fedora`.

Official openSUSE and SLES images

SUSE does not provide openSUSE or SUSE Linux Enterprise Server (SLES) images for direct download. Instead, they provide a web-based tool called SUSE Studio that you can use to build openSUSE and SLES images.

For example, Christian Berendt used openSUSE to create a test openSUSE 12.3 image.

Official images from other Linux distributions

As of this writing, we are not aware of other distributions that provide images for download.

Rackspace Cloud Builders (multiple distros) images

Rackspace Cloud Builders maintains a list of pre-built images from various distributions (Red Hat, CentOS, Fedora, Ubuntu). Links to these images can be found at rackerjoe/oz-image-build on GitHub.

Microsoft Windows images

Cloudbase Solutions hosts an OpenStack Windows Server 2012 Standard Evaluation image that runs on Hyper-V, KVM, and XenServer/XCP.

3. OpenStack Linux image requirements

For a Linux-based image to have full functionality in an OpenStack Compute cloud, there are a few requirements. For some of these, you can fulfill the requirement by installing the cloud-init package. Read this section before you create your own image to be sure that the image supports the OpenStack features that you plan to use.

- Disk partitions and resize root partition on boot (cloud-init)
- No hard-coded MAC address information
- SSH server running
- Disable firewall
- Access instance using ssh public key (cloud-init)
- Process user data and other metadata (cloud-init)
- Paravirtualized Xen support in Linux kernel (Xen hypervisor only with Linux kernel version < 3.0)

Disk partitions and resize root partition on boot (cloud-init)

When you create a Linux image, you must decide how to partition the disks. The choice of partition method can affect the resizing functionality, as described in the following sections.

The size of the disk in a virtual machine image is determined when you initially create the image. However, OpenStack lets you launch instances with different size drives by specifying different flavors. For example, if your image was created with a 5 GB disk, and you launch an instance with a flavor of `m1.small`. The resulting virtual machine instance has, by default, a primary disk size of 10 GB. When the disk for an instance is resized up, zeros are just added to the end.

Your image must be able to resize its partitions on boot to match the size requested by the user. Otherwise, after the instance boots, you must manually resize the partitions to access

the additional storage to which you have access when the disk size associated with the flavor exceeds the disk size with which your image was created.

Xen: 1 ext3/ext4 partition (no LVM, no /boot, no swap)

If you use the OpenStack XenAPI driver, the Compute service automatically adjusts the partition and file system for your instance on boot. Automatic resize occurs if the following conditions are all true:

- `auto_disk_config=True` is set as a property on the image in the image registry.
- The disk on the image has only one partition.
- The file system on the one partition is ext3 or ext4.

Therefore, if you use Xen, we recommend that when you create your images, you create a single ext3 or ext4 partition (not managed by LVM). Otherwise, read on.

Non-Xen with cloud-init/cloud-tools: One ext3/ext4 partition (no LVM, no /boot, no swap)

You must configure these items for your image:

- The partition table for the image describes the original size of the image
- The file system for the image fills the original size of the image

Then, during the boot process, you must:

- Modify the partition table to make it aware of the additional space:
  - If you do not use LVM, you must modify the table to extend the existing root partition to encompass this additional space.
  - If you use LVM, you can add a new LVM entry to the partition table, create a new LVM physical volume, add it to the volume group, and extend the logical partition with the root volume.
- Resize the root volume file system.

The simplest way to support this in your image is to install the cloud-utils package (contains the **growpart** tool for extending partitions), the cloud-initramfs-growroot package (which supports resizing root partition on the first boot), and the cloud-init package into your image. With these installed, the image performs the root partition resize on boot. For example, in the `/etc/rc.local` file. These packages are in the Ubuntu and Debian package repository, as well as the EPEL repository (for Fedora/RHEL/CentOS/Scientific Linux guests).

If you cannot install `cloud-initramfs-tools`, Robert Plestenjak has a GitHub project called linux-rootfs-resize that contains scripts that update a ramdisk by using **growpart** so that the image resizes properly on boot.

If you can install the cloud-utils and cloud-init packages, we recommend that when you create your images, you create a single ext3 or ext4 partition (not managed by LVM).

Non-Xen without cloud-init/cloud-tools: LVM

If you cannot install cloud-init and cloud-tools inside of your guest, and you want to support resize, you must write a script that your image runs on boot to modify the partition table. In this case, we recommend using LVM to manage your partitions. Due to a limitation in the Linux kernel (as of this writing), you cannot modify a partition table of a raw disk that has partitions currently mounted, but you can do this for LVM.

Your script must do something like the following:

1. Detect if any additional space is available on the disk. For example, parse the output of **parted /dev/sda –script "print free"**.
2. Create a new LVM partition with the additional space. For example, **parted /dev/sda –script "mkpart lvm ..."**.
3. Create a new physical volume. For example, **pvcreate /dev/*sda6***.
4. Extend the volume group with this physical partition. For example, **vgextend *vg00* /dev/*sda6***.
5. Extend the logical volume contained the root partition by the amount of space. For example, **lvextend /dev/mapper/*node-root* /dev/*sda6***.
6. Resize the root file system. For example, **resize2fs /dev/mapper/*node-root***.

You do not need a `/boot` partition unless your image is an older Linux distribution that requires that `/boot` is not managed by LVM.

No hard-coded MAC address information

You must remove the network persistence rules in the image because they cause the network interface in the instance to come up as an interface other than eth0. This is because your image has a record of the MAC address of the network interface card when it was first installed, and this MAC address is different each time that the instance boots. You should alter the following files:

- Replace `/etc/udev/rules.d/70-persistent-net.rules` with an empty file (contains network persistence rules, including MAC address)
- Replace `/lib/udev/rules.d/75-persistent-net-generator.rules` with an empty file (this generates the file above)
- Remove the HWADDR line from `/etc/sysconfig/network-scripts/ifcfg-eth0` on Fedora-based images

Note

If you delete the network persistent rules files, you may get a udev kernel warning at boot time, which is why we recommend replacing them with empty files instead.

Ensure ssh server runs

You must install an ssh server into the image and ensure that it starts up on boot, or you cannot connect to your instance by using ssh when it boots inside of OpenStack. This package is typically called `openssh-server`.

Disable firewall

In general, we recommend that you disable any firewalls inside of your image and use OpenStack security groups to restrict access to instances. The reason is that having a firewall installed on your instance can make it more difficult to troubleshoot networking issues if you cannot connect to your instance.

Access instance by using ssh public key (cloud-init)

The typical way that users access virtual machines running on OpenStack is to ssh using public key authentication. For this to work, your virtual machine image must be configured to download the ssh public key from the OpenStack metadata service or config drive, at boot time.

Use cloud-init to fetch the public key

The cloud-init package automatically fetches the public key from the metadata server and places the key in an account. The account varies by distribution. On Ubuntu-based virtual machines, the account is called `ubuntu`. On Fedora-based virtual machines, the account is called `ec2-user`.

You can change the name of the account used by cloud-init by editing the `/etc/cloud/cloud.cfg` file and adding a line with a different user. For example, to configure cloud-init to put the key in an account named `admin`, edit the configuration file so it has the line:

```
user: admin
```

Write a custom script to fetch the public key

If you are unable or unwilling to install cloud-init inside the guest, you can write a custom script to fetch the public key and add it to a user account.

To fetch the ssh public key and add it to the root account, edit the `/etc/rc.local` file and add the following lines before the line “touch /var/lock/subsys/local”. This code fragment is taken from the rackerjoe oz-image-build CentOS 6 template.

```
if [ ! -d /root/.ssh ]; then
  mkdir -p /root/.ssh
  chmod 700 /root/.ssh
fi

# Fetch public key using HTTP
```

```
ATTEMPTS=30
FAILED=0
while [ ! -f /root/.ssh/authorized_keys ]; do
  curl -f http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
 > /tmp/metadata-key 2>/dev/null
  if [ $? -eq 0 ]; then
    cat /tmp/metadata-key >> /root/.ssh/authorized_keys
    chmod 0600 /root/.ssh/authorized_keys
    restorecon /root/.ssh/authorized_keys
    rm -f /tmp/metadata-key
    echo "Successfully retrieved public key from instance metadata"
    echo "*****************"
    echo "AUTHORIZED KEYS"
    echo "*****************"
    cat /root/.ssh/authorized_keys
    echo "*****************"
  else
    FAILED=`expr $FAILED + 1`
    if [ $FAILED -ge $ATTEMPTS ]; then
      echo "Failed to retrieve public key from instance metadata after $FAILED
 attempts, quitting"
      break
    fi
      echo "Could not retrieve public key from instance metadata (attempt #
$FAILED/$ATTEMPTS), retrying in 5 seconds..."
      sleep 5
  fi
done
```

Note

Some VNC clients replace : (colon) with ; (semicolon) and _ (underscore) with - (hyphen). If editing a file over a VNC session, make sure it's http: not http; and authorized_keys not authorized-keys.

Process user data and other metadata (cloud-init)

In addition to the ssh public key, an image might need additional information from OpenStack, such as user data that the user submitted when requesting the image. For example, you might want to set the host name of the instance when it is booted. Or, you might wish to configure your image so that it executes user data content as a script on boot.

This information is accessible through the metadata service or the config drive. As the OpenStack metadata service is compatible with version 2009-04-04 of the Amazon EC2 metadata service, consult the Amazon EC2 documentation on Using Instance Metadata for details on how to retrieve user data.

The easiest way to support this type of functionality is to install the cloud-init package into your image, which is configured by default to treat user data as an executable script, and sets the host name.

Ensure image writes boot log to console

You must configure the image so that the kernel writes the boot log to the `ttyS0` device. In particular, the `console=ttyS0` argument must be passed to the kernel on boot.

If your image uses grub2 as the boot loader, there should be a line in the grub configuration file. For example, `/boot/grub/grub.cfg`, which looks something like this:

```
linux /boot/vmlinuz-3.2.0-49-virtual root=UUID=6d2231e4-0975-4f35-
a94f-56738c1a8150 ro console=ttyS0
```

If `console=ttyS0` does not appear, you must modify your grub configuration. In general, you should not update the `grub.cfg` directly, since it is automatically generated. Instead, you should edit `/etc/default/grub` and modify the value of the `GRUB_CMDLINE_LINUX_DEFAULT` variable:

```
GRUB_CMDLINE_LINUX_DEFAULT="console=ttyS0"
```

Next, update the grub configuration. On Debian-based operating-systems such as Ubuntu, run this command:

```
# update-grub
```

On Fedora-based systems, such as RHEL and CentOS, and on openSUSE, run this command:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

Paravirtualized Xen support in the kernel (Xen hypervisor only)

Prior to Linux kernel version 3.0, the mainline branch of the Linux kernel did not have support paravirtualized Xen virtual machine instances (what Xen calls DomU guests). If you are running the Xen hypervisor with paravirtualization, and you want to create an image for an older Linux distribution that has a pre 3.0 kernel, you must ensure that the image boots a kernel that has been compiled with Xen support.

Manage the image cache

Use options in `nova.conf` to control whether, and for how long, unused base images are stored in `/var/lib/nova/instances/_base/`. If you have configured live migration of instances, all your compute nodes share one common `/var/lib/nova/instances/` directory.

For information about libvirt images in OpenStack, see The life of an OpenStack libvirt image from Pádraig Brady.

Table 3.1. Image cache management configuration options

Configuration option=Default value	(Type) Description
preallocate_images=none	(StrOpt) VM image preallocation mode: • `none`. No storage provisioning occurs up front. • `space`. Storage is fully allocated at instance start. The `$instance_dir/` images are fallocated to immediately determine if enough space is available, and to possibly improve VM I/O performance due to ongoing allocation avoidance, and better locality of block allocations.
remove_unused_base_images=True	(BoolOpt) Should unused base images be removed? When set to True, the interval at which base images are removed

Configuration option=Default value	(Type) Description
	are set with the following two settings. If set to False base images are never removed by Compute.
remove_unused_original_minimum_age_seconds=86400	(IntOpt) Unused unresized base images younger than this are not removed. Default is 86400 seconds, or 24 hours.
remove_unused_resized_minimum_age_seconds=3600	(IntOpt) Unused resized base images younger than this are not removed. Default is 3600 seconds, or one hour.

To see how the settings affect the deletion of a running instance, check the directory where the images are stored:

```
# ls -lash /var/lib/nova/instances/_base/
```

In the `/var/log/compute/compute.log` file, look for the identifier:

```
2012-02-18 04:24:17 41389 WARNING nova.virt.libvirt.imagecache [-] Unknown
 base file: /var/lib/nova/instances/_base/06a057b9c7b0b27e3b496f53d1e88810
a0d1d5d3_20
2012-02-18 04:24:17 41389 INFO nova.virt.libvirt.imagecache [-] Removable base
 files: /var/lib/nova/instances/_base/06a057b9c7b0b27e3b496f53d1e88810
a0d1d5d3 /var/lib/nova/instances/_base/
06a057b9c7b0b27e3b496f53d1e88810a0d1d5d3_20
2012-02-18 04:24:17 41389 INFO nova.virt.libvirt.imagecache [-] Removing base
 file: /var/lib/nova/instances/_base/06a057b9c7b0b27e3b496f53d1e88810a0d1d5d3
```

Because 86400 seconds (24 hours) is the default time for `remove_unused_original_minimum_age_seconds`, you can either wait for that time interval to see the base image removed, or set the value to a shorter time period in `nova.conf`. Restart all nova services after changing a setting in `nova.conf`.

4. Modify images

Once you have obtained a virtual machine image, you may want to make some changes to it before uploading it to the OpenStack Image Service. Here we describe several tools available that allow you to modify images.

Warning

Do not attempt to use these tools to modify an image that is attached to a running virtual machine. These tools are designed to only modify images that are not currently running.

guestfish

The **guestfish** program is a tool from the libguestfs project that allows you to modify the files inside of a virtual machine image.

Note

guestfish does not mount the image directly into the local file system. Instead, it provides you with a shell interface that enables you to view, edit, and delete files. Many of **guestfish** commands, such as **touch**, **chmod**, and **rm**, resemble traditional bash commands.

Example guestfish session

Sometimes, you must modify a virtual machine image to remove any traces of the MAC address that was assigned to the virtual network interface card when the image was first created, because the MAC address will be different when it boots the next time. This example shows how to use guestfish to remove references to the old MAC address by deleting the `/etc/udev/rules.d/70-persistent-net.rules` file and removing the `HWADDR` line from the `/etc/sysconfig/network-scripts/ifcfg-eth0` file.

Assume that you have a CentOS qcow2 image called `centos63_desktop.img`. Mount the image in read-write mode as root, as follows:

```
# guestfish --rw -a centos63_desktop.img

Welcome to guestfish, the libguestfs filesystem interactive shell for
editing virtual machine filesystems.

Type: 'help' for help on commands
      'man' to read the manual
      'quit' to quit the shell
```

```
><fs>
```

This starts a guestfish session. Note that the guestfish prompt looks like a fish: `> <fs>`.

We must first use the **run** command at the guestfish prompt before we can do anything else. This will launch a virtual machine, which will be used to perform all of the file manipulations.

```
><fs> run
```

We can now view the file systems in the image using the **list-filesystems** command:

```
><fs> list-filesystems
/dev/vda1: ext4
/dev/vg_centosbase/lv_root: ext4
/dev/vg_centosbase/lv_swap: swap
```

We need to mount the logical volume that contains the root partition:

```
><fs> mount /dev/vg_centosbase/lv_root /
```

Next, we want to delete a file. We can use the **rm** guestfish command, which works the same way it does in a traditional shell.

```
><fs> rm /etc/udev/rules.d/70-persistent-net.rules
```

We want to edit the `ifcfg-eth0` file to remove the HWADDR line. The **edit** command will copy the file to the host, invoke your editor, and then copy the file back.

```
><fs> edit /etc/sysconfig/network-scripts/ifcfg-eth0
```

If you want to modify this image to load the 8021q kernel at boot time, you must create an executable script in the `/etc/sysconfig/modules/` directory. You can use the **touch** guestfish command to create an empty file, the **edit** command to edit it, and the **chmod** command to make it executable.

```
><fs> touch /etc/sysconfig/modules/8021q.modules
><fs> edit /etc/sysconfig/modules/8021q.modules
```

We add the following line to the file and save it:

```
modprobe 8021q
```

Then we set to executable:

```
><fs> chmod 0755 /etc/sysconfig/modules/8021q.modules
```

We're done, so we can exit using the **exit** command:

```
><fs> exit
```

Go further with guestfish

There is an enormous amount of functionality in guestfish and a full treatment is beyond the scope of this document. Instead, we recommend that you read the guestfs-recipes documentation page for a sense of what is possible with these tools.

guestmount

For some types of changes, you may find it easier to mount the image's file system directly in the guest. The **guestmount** program, also from the libguestfs project, allows you to do so.

For example, to mount the root partition from our `centos63_desktop.qcow2` image to `/mnt`, we can do:

```
# guestmount -a centos63_desktop.qcow2 -m /dev/vg_centosbase/lv_root --rw /mnt
```

If we didn't know in advance what the mount point is in the guest, we could use the `-i`(inspect) flag to tell guestmount to automatically determine what mount point to use:

```
# guestmount -a centos63_desktop.qcow2 -i --rw /mnt
```

Once mounted, we could do things like list the installed packages using rpm:

```
# rpm -qa --dbpath /mnt/var/lib/rpm
```

Once done, we unmount:

```
# umount /mnt
```

virt-* tools

The libguestfs project has a number of other useful tools, including:

- virt-edit for editing a file inside of an image.
- virt-df for displaying free space inside of an image.
- virt-resize for resizing an image.
- virt-sysprep for preparing an image for distribution (for example, delete SSH host keys, remove MAC address info, or remove user accounts).
- virt-sparsify for making an image sparse
- virt-p2v for converting a physical machine to an image that runs on KVM
- virt-v2v for converting Xen and VMware images to KVM images

Modify a single file inside of an image

This example shows how to use **virt-edit** to modify a file. The command can take either a filename as an argument with the `-a` flag, or a domain name as an argument with the `-d` flag. The following examples shows how to use this to modify the `/etc/shadow` file in instance with libvirt domain name `instance-000000e1` that is currently running:

```
# virsh shutdown instance-000000e1
# virt-edit -d instance-000000e1 /etc/shadow
```

```
# virsh start instance-000000e1
```

Resize an image

Here's a simple of example of how to use **virt-resize** to resize an image. Assume we have a 16 GB Windows image in qcow2 format that we want to resize to 50 GB. First, we use **virt-filesystems** to identify the partitions:

```
# virt-filesystems --long --parts --blkdevs -h -a /data/images/win2012.qcow2
Name       Type       MBR  Size  Parent
/dev/sda1  partition  07   350M  /dev/sda
/dev/sda2  partition  07   16G   /dev/sda
/dev/sda   device     -    16G   -
```

In this case, it's the `/dev/sda2` partition that we want to resize. We create a new qcow2 image and use the **virt-resize** command to write a resized copy of the original into the new image:

```
# qemu-img create -f qcow2 /data/images/win2012-50gb.qcow2 50G
# virt-resize --expand /dev/sda2 /data/images/win2012.qcow2 \
  /data/images/win2012-50gb.qcow2
Examining /data/images/win2012.qcow2 ...
**********

Summary of changes:

/dev/sda1: This partition will be left alone.

/dev/sda2: This partition will be resized from 15.7G to 49.7G.  The
    filesystem ntfs on /dev/sda2 will be expanded using the
    'ntfsresize' method.

**********
Setting up initial partition table on /data/images/win2012-50gb.qcow2 ...
Copying /dev/sda1 ...
 100% ############################################################
 00:00
Copying /dev/sda2 ...
 100% ############################################################
 00:00
Expanding /dev/sda2 using the 'ntfsresize' method ...

Resize operation completed with no errors.  Before deleting the old
disk, carefully check that the resized disk boots and works correctly.
```

Loop devices, kpartx, network block devices

If you don't have access to libguestfs, you can mount image file systems directly in the host using loop devices, kpartx, and network block devices.

Warning

Mounting untrusted guest images using the tools described in this section is a security risk, always use libguestfs tools such as guestfish and guestmount if you have access to them. See A reminder why you should never mount guest disk images on the host OS by Daniel Berrangé for more details.

Mount a raw image (without LVM)

If you have a raw virtual machine image that is not using LVM to manage its partitions. First, use the **losetup** command to find an unused loop device.

```
# losetup -f
/dev/loop0
```

In this example, `/dev/loop0` is free. Associate a loop device with the raw image:

```
# losetup /dev/loop0 fedora17.img
```

If the image only has a single partition, you can mount the loop device directly:

```
# mount /dev/loop0 /mnt
```

If the image has multiple partitions, use **kpartx** to expose the partitions as separate devices (for example, `/dev/mapper/loop0p1`), then mount the partition that corresponds to the root file system:

```
# kpartx -av /dev/loop0
```

If the image has, say three partitions (/boot, /, swap), there should be one new device created per partition:

```
$ ls -l /dev/mapper/loop0p*
brw-rw---- 1 root disk 43, 49 2012-03-05 15:32 /dev/mapper/loop0p1
brw-rw---- 1 root disk 43, 50 2012-03-05 15:32 /dev/mapper/loop0p2
brw-rw---- 1 root disk 43, 51 2012-03-05 15:32 /dev/mapper/loop0p3
```

To mount the second partition, as root:

```
# mkdir /mnt/image
# mount /dev/mapper/loop0p2 /mnt
```

Once you're done, to clean up:

```
# umount /mnt
# kpartx -d /dev/loop0
# losetup -d /dev/loop0
```

Mount a raw image (with LVM)

If your partitions are managed with LVM, use losetup and kpartx as in the previous example to expose the partitions to the host:

```
# losetup -f
/dev/loop0
# losetup /dev/loop0 rhel62.img
# kpartx -av /dev/loop0
```

Next, you need to use the **vgscan** command to identify the LVM volume groups and then **vgchange** to expose the volumes as devices:

```
# vgscan
Reading all physical volumes.  This may take a while...
  Found volume group "vg_rhel62x8664" using metadata type lvm2
# vgchange -ay
```

```
   2 logical volume(s) in volume group "vg_rhel62x8664" now active
# mount /dev/vg_rhel62x8664/lv_root /mnt
```

Clean up when you're done:

```
# umount /mnt
# vgchange -an vg_rhel62x8664
# kpartx -d /dev/loop0
# losetup -d /dev/loop0
```

Mount a qcow2 image (without LVM)

You need the `nbd` (network block device) kernel module loaded to mount qcow2 images. This will load it with support for 16 block devices, which is fine for our purposes. As root:

```
# modprobe nbd max_part=16
```

Assuming the first block device (`/dev/nbd0`) is not currently in use, we can expose the disk partitions using the **qemu-nbd** and **partprobe** commands. As root:

```
# qemu-nbd -c /dev/nbd0 image.qcow2
# partprobe /dev/nbd0
```

If the image has, say three partitions (/boot, /, swap), there should be one new device created for each partition:

```
$ ls -l /dev/nbd3*
brw-rw---- 1 root disk 43, 48 2012-03-05 15:32 /dev/nbd0
brw-rw---- 1 root disk 43, 49 2012-03-05 15:32 /dev/nbd0p1
brw-rw---- 1 root disk 43, 50 2012-03-05 15:32 /dev/nbd0p2
brw-rw---- 1 root disk 43, 51 2012-03-05 15:32 /dev/nbd0p3
```

Note

If the network block device you selected was already in use, the initial **qemu-nbd** command will fail silently, and the `/dev/nbd3p{1,2,3}` device files will not be created.

If the image partitions are not managed with LVM, they can be mounted directly:

```
# mkdir /mnt/image
# mount /dev/nbd3p2 /mnt
```

When you're done, clean up:

```
# umount /mnt
# qemu-nbd -d /dev/nbd0
```

Mount a qcow2 image (with LVM)

If the image partitions are managed with LVM, after you use **qemu-nbd** and **partprobe**, you must use **vgscan** and **vgchange -ay** in order to expose the LVM partitions as devices that can be mounted:

```
# modprobe nbd max_part=16
# qemu-nbd -c /dev/nbd0 image.qcow2
# partprobe /dev/nbd0# vgscan
```

```
  Reading all physical volumes.  This may take a while...
  Found volume group "vg_rhel62x8664" using metadata type lvm2
# vgchange -ay
   2 logical volume(s) in volume group "vg_rhel62x8664" now active
# mount /dev/vg_rhel62x8664/lv_root /mnt
```

When you're done, clean up:

```
# umount /mnt
# vgchange -an vg_rhel62x8664
# qemu-nbd -d /dev/nbd0
```

5. Create images manually

Creating a new image is a step done outside of your OpenStack installation. You create the new image manually on your own system and then upload the image to your cloud.

To create a new image, you will need the installation CD or DVD ISO file for the guest operating system. You'll also need access to a virtualization tool. You can use KVM for this. Or, if you have a GUI desktop virtualization tool (such as, VMware Fusion and VirtualBox), you can use that instead and just convert the file to raw once you're done.

When you create a new virtual machine image, you will need to connect to the graphical console of the hypervisor, which acts as the virtual machine's display and allows you to interact with the guest operating system's installer using your keyboard and mouse. KVM can expose the graphical console using the VNC (Virtual Network Computing) protocol or the newer SPICE protocol. We'll use the VNC protocol here, since you're more likely to be able to find a VNC client that works on your local desktop.

Verify the libvirt default network is running

Before starting a virtual machine with libvirt, verify that the libvirt "default" network has been started. This network must be active for your virtual machine to be able to connect out to the network. Starting this network will create a Linux bridge (usually called `virbr0`), iptables rules, and a dnsmasq process that will serve as a DHCP server.

To verify that the libvirt "default" network is enabled, use the **virsh net-list** command and verify that the "default" network is active:

```
# virsh net-list
Name                 State      Autostart
-----------------------------------------
default              active     yes
```

If the network is not active, start it by doing:

```
# virsh net-start default
```

Use the virt-manager X11 GUI

If you plan to create a virtual machine image on a machine that can run X11 applications, the simplest way to do so is to use the **virt-manager** GUI, which is installable as the `virt-manager` package on both Fedora-based and Debian-based systems. This GUI has an embedded VNC client in it that will let you view and interact with the guest's graphical console.

If you are building the image on a headless server, and you have an X server on your local machine, you can launch **virt-manager** using ssh X11 forwarding to access the GUI. Since virt-manager interacts directly with libvirt, you typically need to be root to access it. If you can ssh directly in as root (or with a user that has permissions to interact with libvirt), do:

```
$ ssh -X root@server virt-manager
```

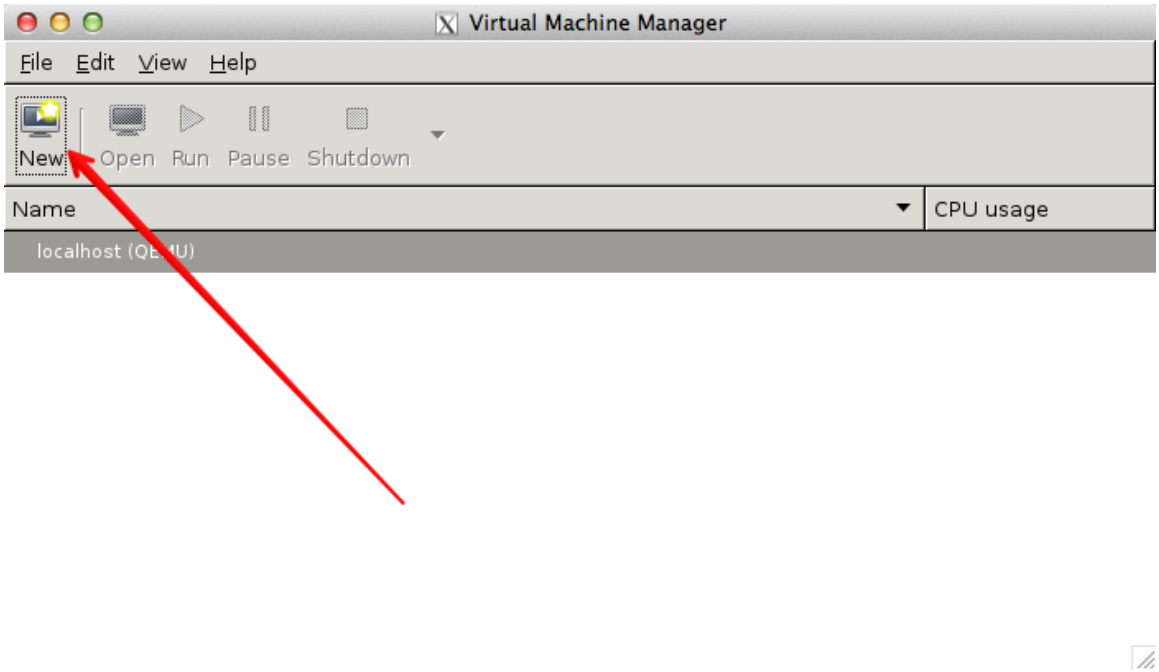
If the account you use to ssh into your server does not have permissions to run libvirt, but has sudo privileges, do:

```
$ ssh -X root@server
$ sudo virt-manager
```

Note

The `-X` flag passed to ssh will enable X11 forwarding over ssh. If this does not work, try replacing it with the `-Y` flag.

Click the "New" button at the top-left and step through the instructions.



You will be shown a series of dialog boxes that will allow you to specify information about the virtual machine.

Note

When using qcow2 format images you should check the option 'customize before install', go to disk properties and explicitly select the qcow2 format. This ensures the virtual machine disk size will be correct.

Use virt-install and connect by using a local VNC client

If you do not wish to use virt-manager (for example, you do not want to install the dependencies on your server, you don't have an X server running locally, the X11 forwarding over SSH isn't working), you can use the **virt-install** tool to boot the virtual machine through libvirt and connect to the graphical console from a VNC client installed on your local machine.

Because VNC is a standard protocol, there are multiple clients available that implement the VNC spec, including TigerVNC (multiple platforms), TightVNC (multiple platforms), RealVNC (multiple platforms), Chicken (Mac OS X), Krde (KDE), and Vinagre (GNOME).

The following example shows how to use the **qemu-img** command to create an empty image file **virt-install** command to start up a virtual machine using that image file. As root:

```
# qemu-img create -f qcow2 /data/centos-6.4.qcow2 10G
# virt-install --virt-type kvm --name centos-6.4 --ram 1024 \
--cdrom=/data/CentOS-6.4-x86_64-netinstall.iso \
--disk path=/data/centos-6.4.qcow2,size=10,format=qcow2 \
--network network=default\
--graphics vnc,listen=0.0.0.0 --noautoconsole \
--os-type=linux --os-variant=rhel6

Starting install...
Creating domain...                                                    |
  0 B     00:00
Domain installation still in progress. You can reconnect to
the console to complete the installation process.
```

The KVM hypervisor starts the virtual machine with the libvirt name, `centos-6.4`, with 1024 MB of RAM. The virtual machine also has a virtual CD-ROM drive associated with the `/data/CentOS-6.4-x86_64-netinstall.iso` file and a local 10 GB hard disk in qcow2 format that is stored in the host at `/data/centos-6.4.qcow2`. It configures networking to use libvirt's default network. There is a VNC server that is listening on all interfaces, and libvirt will not attempt to launch a VNC client automatically nor try to display the text console (`--no-autoconsole`). Finally, libvirt will attempt to optimize the configuration for a Linux guest running a RHEL 6.x distribution.

Note

When using the libvirt `default` network, libvirt will connect the virtual machine's interface to a bridge called `virbr0`. There is a dnsmasq process managed by libvirt that will hand out an IP address on the 192.168.122.0/24 subnet, and libvirt has iptables rules for doing NAT for IP addresses on this subnet.

Run the **virt-install –os-variant list** command to see a range of allowed `--os-variant` options.

Use the **virsh vncdisplay *vm-name*** command to get the VNC port number.

```
# virsh vncdisplay centos-6.4
:1
```

In the example above, the guest `centos-6.4` uses VNC display `:1`, which corresponds to TCP port `5901`. You should be able to connect a VNC client running on your local machine to display :1 on the remote machine and step through the installation process.

Example: CentOS image

This example shows you how to install a CentOS image and focuses mainly on CentOS 6.4. Because the CentOS installation process might differ across versions, the installation steps might differ if you use a different version of CentOS.

Download a CentOS install ISO

1. Navigate to the CentOS mirrors page.
2. Click one of the HTTP links in the right-hand column next to one of the mirrors.
3. Click the folder link of the CentOS version that you want to use. For example, `6.4/`.
4. Click the `isos/` folder link.
5. Click the `x86_64/` folder link for 64-bit images.
6. Click the netinstall ISO image that you want to download. For example, `CentOS-6.4-x86_64-netinstall.iso` is a good choice because it is a smaller image that downloads missing packages from the Internet during installation.

Start the installation process

Start the installation process using either **virt-manager** or **virt-install** as described in the previous section. If you use **virt-install**, do not forget to connect your VNC client to the virtual machine.

Assume that:

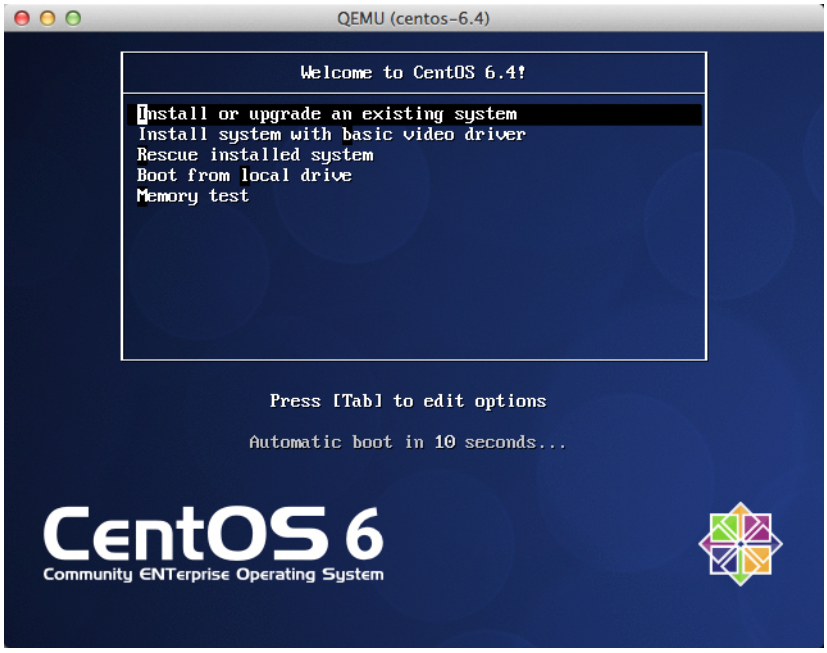
- The name of your virtual machine image is `centos-6.4`; you need this name when you use **virsh** commands to manipulate the state of the image.
- You saved the netinstall ISO image to the `/data/isos` directory.

If you use **virt-install**, the commands should look something like this:

```
# qemu-img create -f qcow2 /tmp/centos-6.4.qcow2 10G
# virt-install --virt-type kvm --name centos-6.4 --ram 1024 \
--disk /tmp/centos-6.4.qcow2,format=qcow2 \
--network network=default \
--graphics vnc,listen=0.0.0.0 --noautoconsole \
--os-type=linux --os-variant=rhel6 \
--extra-args="console=tty0 console=ttyS0,115200n8 serial" \
--location=/data/isos/CentOS-6.4-x86_64-netinstall.iso
```

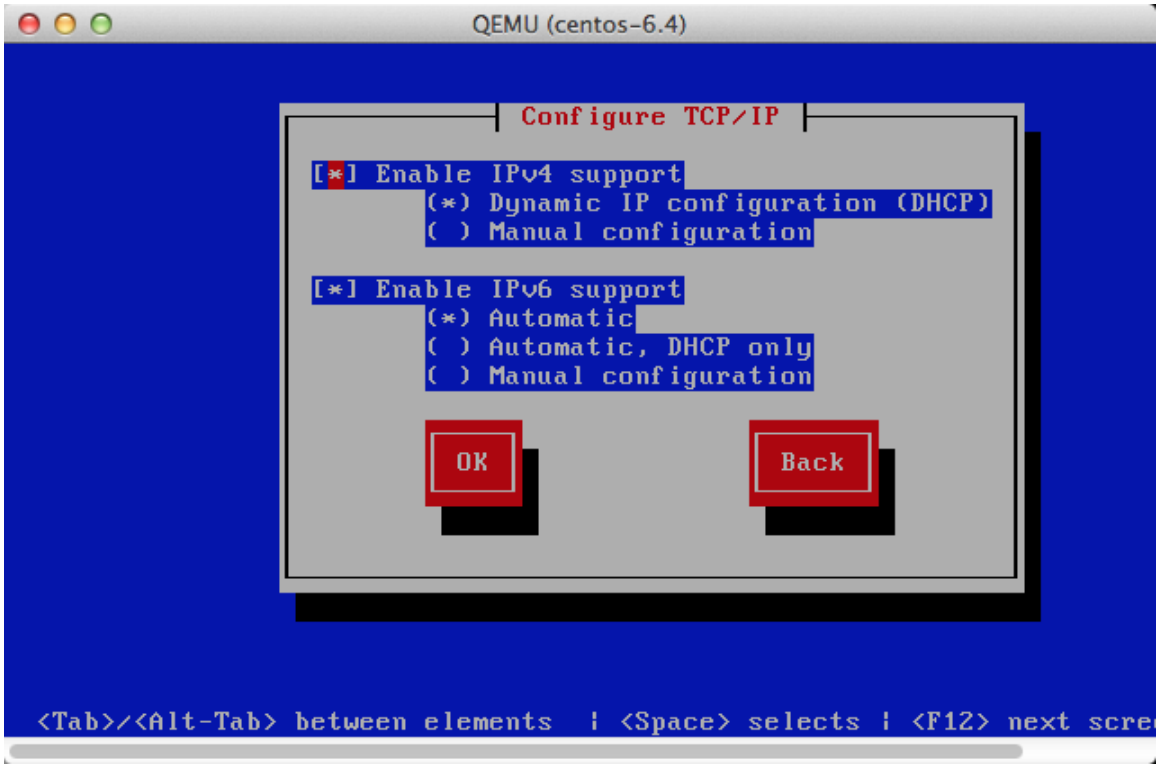
Step through the installation

At the initial Installer boot menu, choose the **Install or upgrade an existing system** option. Step through the installation prompts. Accept the defaults.



Configure TCP/IP

The default TCP/IP settings are fine. In particular, ensure that Enable IPv4 support is enabled with DHCP, which is the default.



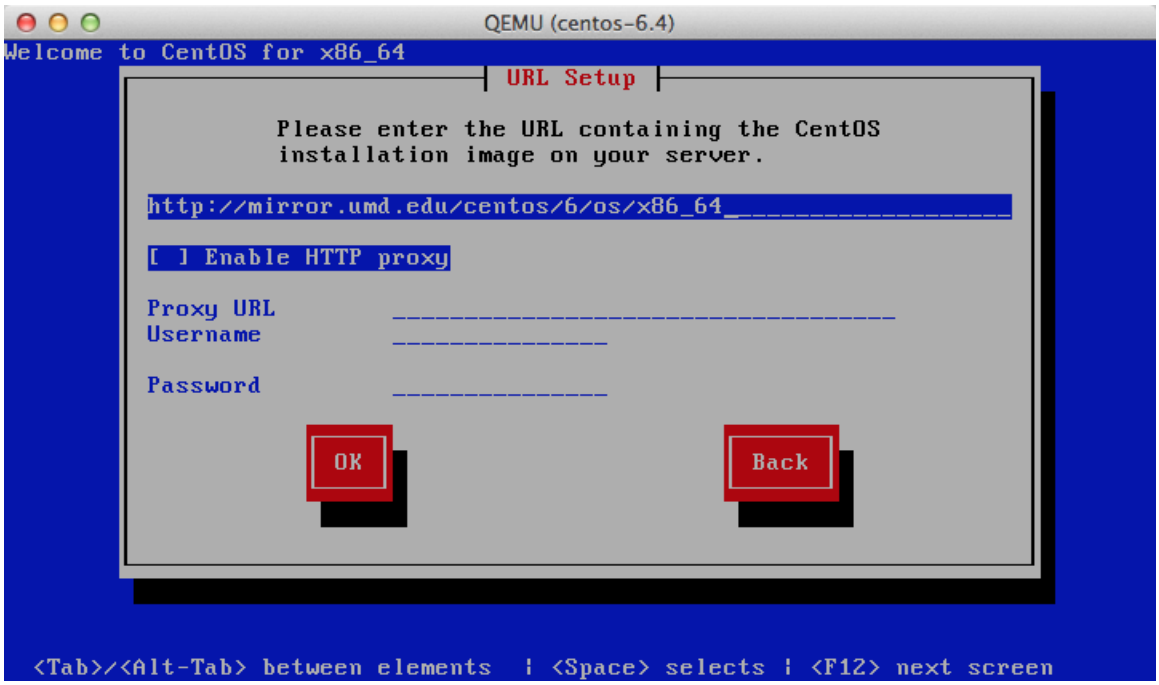
Point the installer to a CentOS web server

Choose URL as the installation method.

Depending on the version of CentOS, the net installer requires that the user specify either a URL or the web site and a CentOS directory that corresponds to one of the CentOS mirrors. If the installer asks for a single URL, a valid URL might be `http://mirror.umd.edu/centos/6/os/x86_64`.

Note

Consider using other mirrors as an alternative to `mirror.umd.edu`.



If the installer asks for web site name and CentOS directory separately, you might enter:

- Web site name: `mirror.umd.edu`
- CentOS directory: `centos/6/os/x86_64`

See CentOS mirror page to get a full list of mirrors, click on the "HTTP" link of a mirror to retrieve the web site name of a mirror.

Storage devices

If prompted about which type of devices your installation uses, choose **Basic Storage Devices**.

Hostname

The installer may ask you to choose a host name. The default (`localhost.localdomain`) is fine. You install the `cloud-init` package later, which sets the host name on boot when a new instance is provisioned using this image.

Partition the disks

There are different options for partitioning the disks. The default installation uses LVM partitions, and creates three partitions (`/boot`, `/`, swap), which works fine. Alternatively, you might want to create a single ext4 partition that is mounted to "`/`", which also works fine.

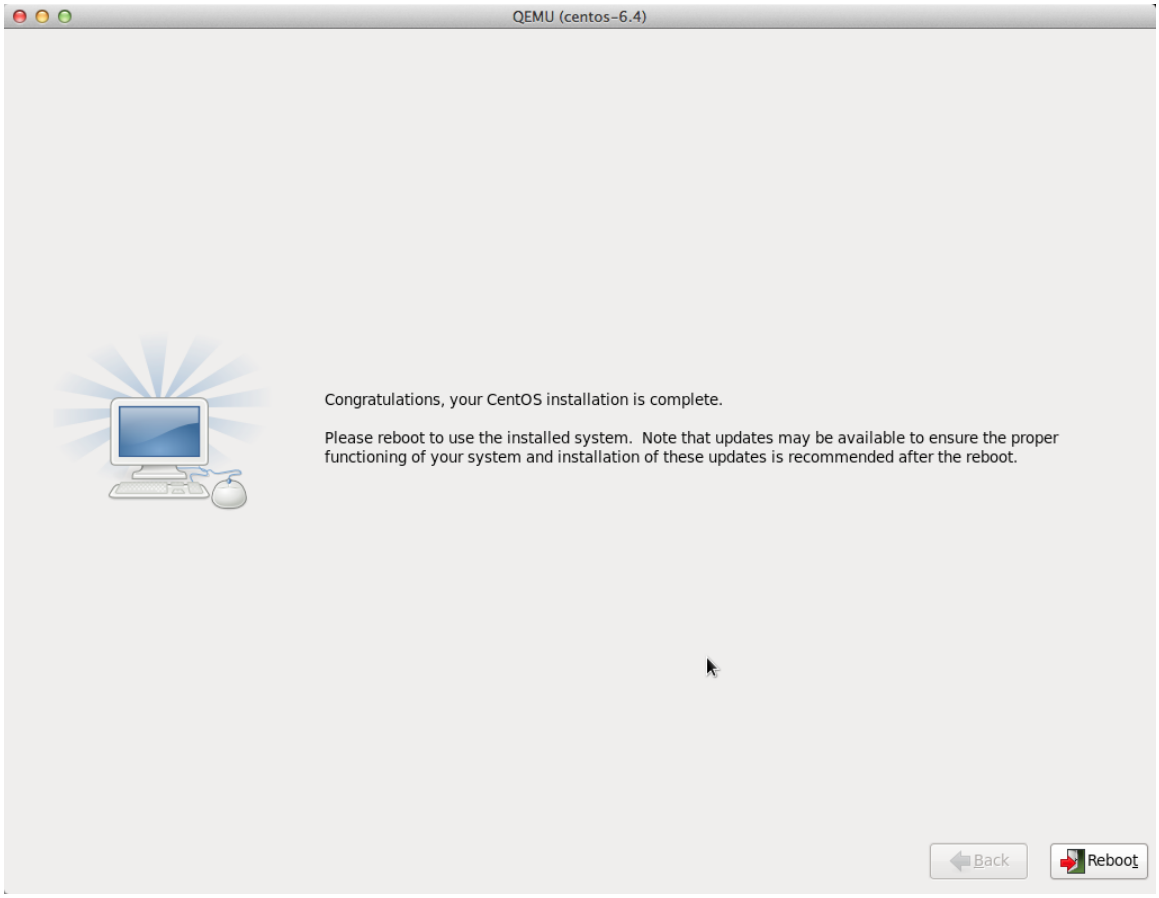
If unsure, use the default partition scheme for the installer because no scheme is better than another.

Step through the installation

Step through the installation, using the default options. The simplest thing to do is to choose the "Basic Server" install (may be called "Server" install on older versions of CentOS), which installs an SSH server.

Detach the CD-ROM and reboot

After the install completes, the **Congratulations, your CentOS installation is complete** screen appears.



To eject a disk by using the **virsh** command, libvirt requires that you attach an empty disk at the same target that the CDROM was previously attached, which should be hdc. You can confirm the appropriate target using the **dom dumpxml *vm-image*** command.

```
# virsh dumpxml centos-6.4
<domain type='kvm'>
  <name>centos-6.4</name>
...
    <disk type='block' device='cdrom'>
      <driver name='qemu' type='raw'/>
      <target dev='hdc' bus='ide'/>
      <readonly/>
      <address type='drive' controller='0' bus='1' target='0' unit='0'/>
    </disk>
...
</domain>
```

Run the following commands from the host to eject the disk and reboot using virsh, as root. If you are using virt-manager, the commands below will work, but you can also use the GUI to detach and reboot it by manually stopping and starting.

```
# virsh attach-disk --type cdrom --mode readonly centos-6.4 "" hdc
# virsh destroy centos-6.4
# virsh start centos-6.4
```

Log in to newly created image

When you boot for the first time after installation, you might be prompted about authentication tools. Select **Exit**. Then, log in as root.

Install the ACPI service

To enable the hypervisor to reboot or shutdown an instance, you must install and run the `acpid` service on the guest system.

Run the following commands inside the CentOS guest to install the ACPI service and configure it to start when the system boots:

```
# yum install acpid
# chkconfig acpid on
```

Configure to fetch metadata

An instance must interact with the metadata service to perform several tasks on start up. For example, the instance must get the ssh public key and run the user data script. To ensure that the instance performs these tasks, use one of these methods:

- Install a `cloud-init` RPM, which is a port of the Ubuntu cloud-init package. This is the recommended approach.
- Modify `/etc/rc.local` to fetch desired information from the metadata service, as described in the next section.

Use cloud-init to fetch the public key

The `cloud-init` package automatically fetches the public key from the metadata server and places the key in an account. You can install `cloud-init` inside the CentOS guest by adding the EPEL repo:

```
# yum install http://download.fedoraproject.org/pub/epel/6/x86_64/epel-
release-6-8.noarch.rpm
# yum install cloud-init
```

The account varies by distribution. On Ubuntu-based virtual machines, the account is called `ubuntu`. On Fedora-based virtual machines, the account is called `ec2-user`.

You can change the name of the account used by `cloud-init` by editing the `/etc/cloud/cloud.cfg` file and adding a line with a different user. For example, to configure `cloud-init` to put the key in an account named `admin`, add this line to the configuration file:

```
user: admin
```

Write a script to fetch the public key (if no cloud-init)

If you are not able to install the `cloud-init` package in your image, to fetch the ssh public key and add it to the root account, edit the `/etc/rc.local` file and add the following lines before the line `touch /var/lock/subsys/local`:

```
if [ ! -d /root/.ssh ]; then
```

```
  mkdir -p /root/.ssh
  chmod 700 /root/.ssh
fi

# Fetch public key using HTTP
ATTEMPTS=30
FAILED=0
while [ ! -f /root/.ssh/authorized_keys ]; do
  curl -f http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key \
    > /tmp/metadata-key 2>/dev/null
  if [ \$? -eq 0 ]; then
    cat /tmp/metadata-key >> /root/.ssh/authorized_keys
    chmod 0600 /root/.ssh/authorized_keys
    restorecon /root/.ssh/authorized_keys
    rm -f /tmp/metadata-key
    echo "Successfully retrieved public key from instance metadata"
    echo "*****************"
    echo "AUTHORIZED KEYS"
    echo "*****************"
    cat /root/.ssh/authorized_keys
    echo "*****************"
done
```

Note

Some VNC clients replace the colon (`:`) with a semicolon (`;`) and the underscore (_) with a hyphen (`-`). Make sure to specify `http:` and not `http;`. Make sure to specify `authorized_keys` and not `authorized-keys`.

Note

The previous script only gets the ssh public key from the metadata server. It does not get user data, which is optional data that can be passed by the user when requesting a new instance. User data is often used to run a custom script when an instance boots.

As the OpenStack metadata service is compatible with version 2009-04-04 of the Amazon EC2 metadata service, consult the Amazon EC2 documentation on Using Instance Metadata for details on how to get user data.

Disable the zeroconf route

For the instance to access the metadata service, you must disable the default zeroconf route:

```
# echo "NOZEROCONF=yes" >> /etc/sysconfig/network
```

Configure console

For the **nova console-log** command to work properly on CentOS 6.*x*, you might need to add the following lines to the `/boot/grub/menu.lst` file:

```
serial --unit=0 --speed=115200
terminal --timeout=10 console serial
# Edit the kernel line to add the console entries
kernel ... console=tty0 console=ttyS0,115200n8
```

Shut down the instance

From inside the instance, as root:

```
# /sbin/shutdown -h now
```

Clean up (remove MAC address details)

The operating system records the MAC address of the virtual Ethernet card in locations such as `/etc/sysconfig/network-scripts/ifcfg-eth0` and `/etc/udev/rules.d/70-persistent-net.rules` during the instance process. However, each time the image boots up, the virtual Ethernet card will have a different MAC address, so this information must be deleted from the configuration file.

There is a utility called **virt-sysprep**, that performs various cleanup tasks such as removing the MAC address references. It will clean up a virtual machine image in place:

```
# virt-sysprep -d centos-6.4
```

Undefine the libvirt domain

Now that you can upload the image to the Image Service, you no longer need to have this virtual machine image managed by libvirt. Use the **virsh undefine *`vm-image`*** command to inform libvirt:

```
# virsh undefine centos-6.4
```

Image is complete

The underlying image file that you created with **qemu-img create** is ready to be uploaded. For example, you can upload the `/tmp/centos-6.4.qcow2` image to the Image Service.

Example: Ubuntu image

This example installs a Ubuntu 14.04 (Trusty Tahr) image. To create an image for a different version of Ubuntu, follow these steps with the noted differences.

Download an Ubuntu install ISO

Because the goal is to make the smallest possible base image, this example uses the network installation ISO. The Ubuntu 64-bit 14.04 network installer ISO is at http://archive.ubuntu.com/ubuntu/dists/trusty/main/installer-amd64/current/images/netboot/mini.iso.

Start the install process

Start the installation process by using either **virt-manager** or **virt-install** as described in the previous section. If you use **virt-install**, do not forget to connect your VNC client to the virtual machine.

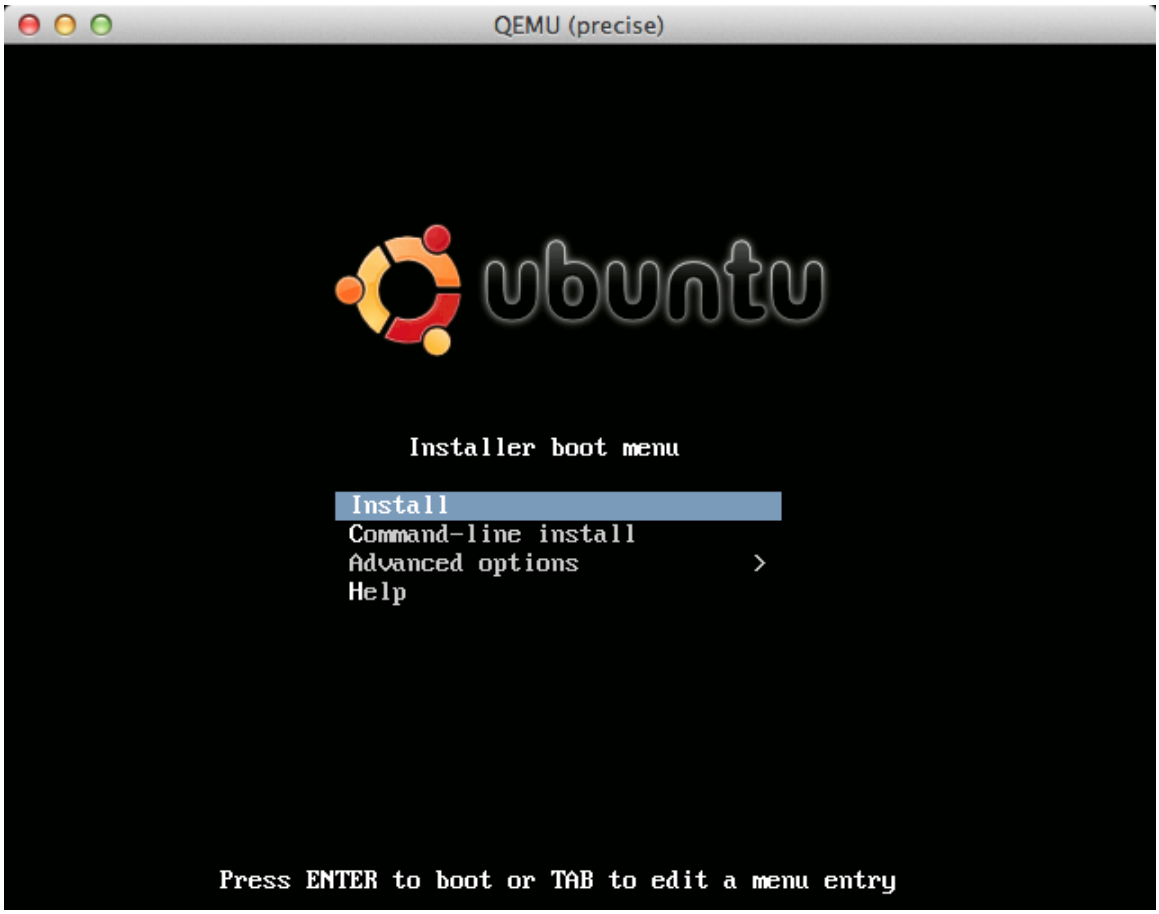
Assume that the name of your virtual machine image is `ubuntu-14.04`, which you need to know when you use **virsh** commands to manipulate the state of the image.

If you are using **virt-manager**, the commands should look something like this:

```
# qemu-img create -f qcow2 /tmp/trusty.qcow2 10G
# virt-install --virt-type kvm --name trusty --ram 1024 \
    --cdrom=/data/isos/trusty-64-mini.iso \
    --disk /tmp/trusty.qcow2,format=qcow2 \
    --network network=default \
    --graphics vnc,listen=0.0.0.0 --noautoconsole \
    --os-type=linux --os-variant=ubuntutrusty
```

Step through the install

At the initial Installer boot menu, choose the **Install** option. Step through the install prompts, the defaults should be fine.



Hostname

The installer may ask you to choose a hostname. The default (`ubuntu`) is fine. We will install the cloud-init package later, which will set the hostname on boot when a new instance is provisioned using this image.

Select a mirror

The default mirror proposed by the installer should be fine.

Step through the install

Step through the install, using the default options. When prompted for a user name, the default (`ubuntu`) is fine.

Partition the disks

There are different options for partitioning the disks. The default installation will use LVM partitions, and will create three partitions (`/boot`, `/`, swap), and this will work fine. Alternatively, you may wish to create a single ext4 partition, mounted to "`/`", should also work fine.

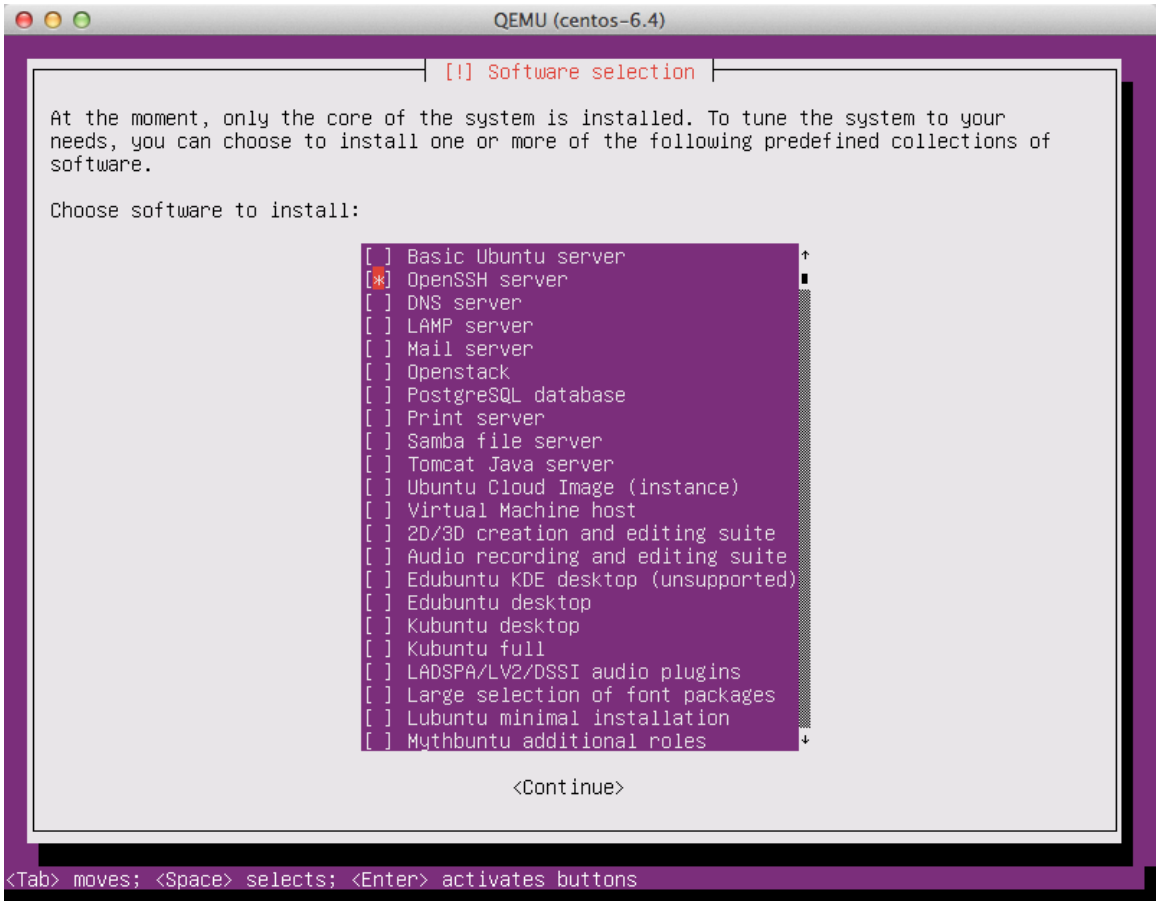
If unsure, we recommend you use the installer's default partition scheme, since there is no clear advantage to one scheme or another.

Automatic updates

The Ubuntu installer will ask how you want to manage upgrades on your system. This option depends on your specific use case. If your virtual machine instances will be connected to the Internet, we recommend "Install security updates automatically".

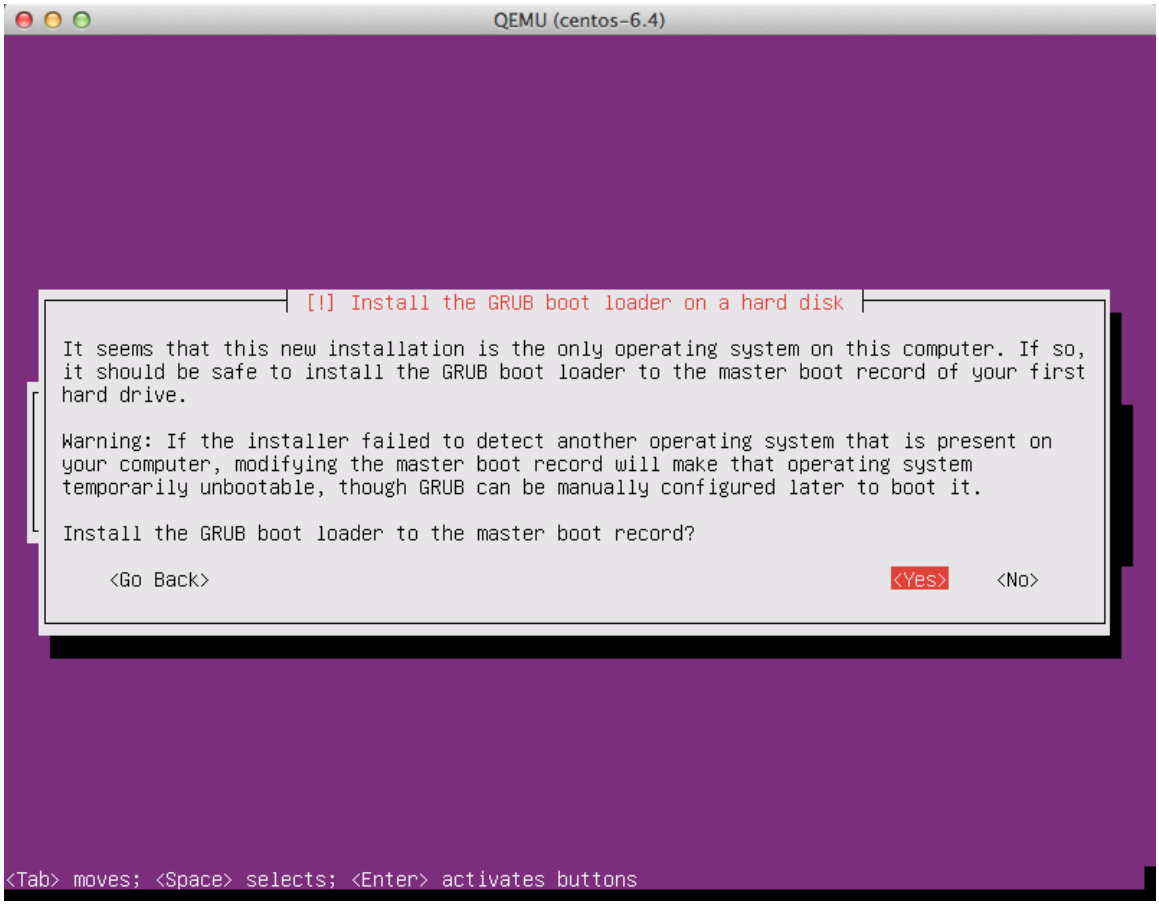
Software selection: OpenSSH server

Choose "OpenSSH server"so that you will be able to SSH into the virtual machine when it launches inside of an OpenStack cloud.



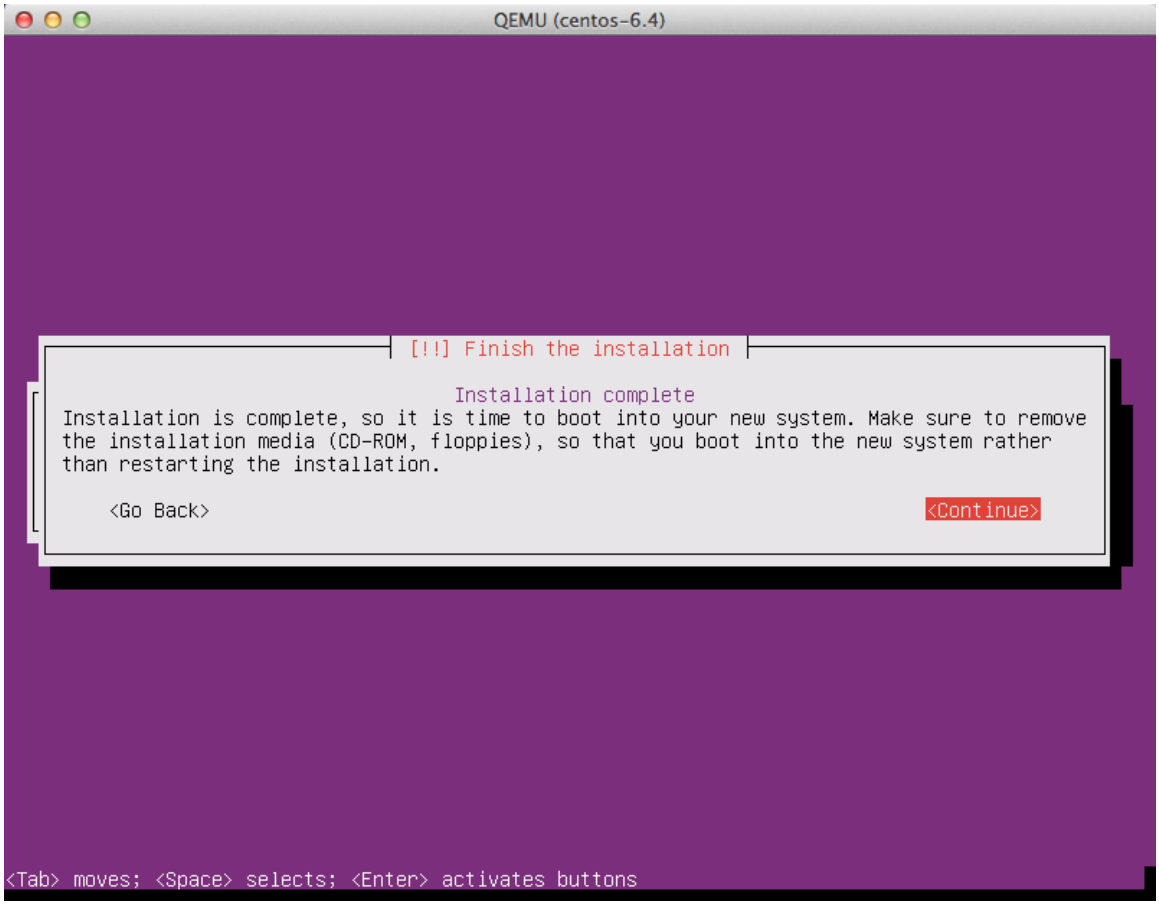
Install GRUB boot loader

Select "Yes" when asked about installing the GRUB boot loader to the master boot record.



Detach the CD-ROM and reboot

Select the defaults for all of the remaining options. When the installation is complete, you will be prompted to remove the CD-ROM.



Note

When you hit "Continue" the virtual machine will shut down, even though it says it will reboot.

To eject a disk using **virsh**, libvirt requires that you attach an empty disk at the same target that the CDROM was previously attached, which should be hdc. You can confirm the appropriate target using the **dom dumpxml *vm-image*** command.

```
# virsh dumpxml trusty
<domain type='kvm'>
  <name>trusty</name>
...
    <disk type='block' device='cdrom'>
      <driver name='qemu' type='raw'/>
      <target dev='hdc' bus='ide'/>
      <readonly/>
      <address type='drive' controller='0' bus='1' target='0' unit='0'/>
    </disk>
...
</domain>
```

Run the following commands in the host as root to start up the machine again as paused, eject the disk and resume. If you are using virt-manager, you may use the GUI instead.

```
# virsh start trusty --paused
```

```
# virsh attach-disk --type cdrom --mode readonly trusty "" hdc
# virsh resume trusty
```

Note

In the previous example, you paused the instance paused, ejected the disk, and unpaused the instance. In theory, you could have ejected the disk at the **Installation complete** screen. However, our testing indicates that the Ubuntu installer locks the drive so that it cannot be ejected at that point.

Log in to newly created image

When you boot for the first time after install, it may ask you about authentication tools, you can just choose 'Exit'. Then, log in as root using the root password you specified.

Install cloud-init

The **cloud-init** script starts on instance boot and will search for a metadata provider to fetch a public key from. The public key will be placed in the default user account for the image.

Install the cloud-init package:

```
# apt-get install cloud-init
```

When building Ubuntu images **cloud-init** must be explicitly configured for the metadata source in use. The OpenStack metadata server emulates the EC2 metadata service used by images in Amazon EC2.

To set the metadata source to be used by the image run the **dpkg-reconfigure** command against the cloud-init package. When prompted select the EC2 data source:

```
# dpkg-reconfigure cloud-init
```

The account varies by distribution. On Ubuntu-based virtual machines, the account is called "ubuntu". On Fedora-based virtual machines, the account is called "ec2-user".

You can change the name of the account used by cloud-init by editing the `/etc/cloud/cloud.cfg` file and adding a line with a different user. For example, to configure cloud-init to put the key in an account named "admin", edit the config file so it has the line:

```
user: admin
```

Shut down the instance

From inside the instance, as root:

```
# /sbin/shutdown -h now
```

Clean up (remove MAC address details)

The operating system records the MAC address of the virtual Ethernet card in locations such as `/etc/udev/rules.d/70-persistent-net.rules` during the instance process. However, each time the image boots up, the virtual Ethernet card will have a different MAC address, so this information must be deleted from the configuration file.

There is a utility called **virt-sysprep**, that performs various cleanup tasks such as removing the MAC address references. It will clean up a virtual machine image in place:

```
# virt-sysprep -d trusty
```

Undefine the libvirt domain

Now that the image is ready to be uploaded to the Image Service, you no longer need to have this virtual machine image managed by libvirt. Use the **virsh undefine *vm-image*** command to inform libvirt:

```
# virsh undefine trusty
```

Image is complete

The underlying image file that you created with **qemu-img create**, such as `/tmp/trusty.qcow2`, is now ready for uploading to the OpenStack Image Service.

Example: Microsoft Windows image

This example creates a Windows Server 2012 qcow2 image, using **virt-install** and the KVM hypervisor.

1. Follow this steps to prepare the installation:

   a. Download a Windows Server 2012 installation ISO. Evaluation images are available on the Microsoft website (registration required).

   b. Download the signed VirtIO drivers ISO from the Fedora website.

   c. Create a 10 GB qcow2 image:

   ```
   $ qemu-img create -f qcow2 ws2012.qcow2 10G
   ```

2. Start the Windows Server 2012 installation with the **virt-install** command:

   ```
   # virt-install --connect qemu:///system \
     --name ws2012 --ram 2048 --vcpus 2 \
     --network network=default,model=virtio \
     --disk path=ws2012.qcow2,device=disk,bus=virtio \
     --cdrom /path/to/en_windows_server_2012_x64_dvd.iso \
     --disk path=/path/to/virtio-win-0.1-XX.iso,device=cdrom \
     --vnc --os-type windows --os-variant win2k8
   ```

   Use **virt-manager** or **virt-viewer** to connect to the VM and start the Windows installation.

3. Enable the VirtIO drivers.

   The disk is not detected by default by the Windows installer. When requested to choose an installation target, click **Load driver** and browse the file system to select the `E:\WIN8\AMD64` folder. The Windows installer displays a list of drivers to install. Select the VirtIO SCSI and network drivers, and continue the installation.

   Once the installation is completed, the VM restarts. Define a password for the administrator when prompted.

4. Log in as administrator and start a command window.

5. Complete the VirtIO drivers installation by running the following command:

```
C:\pnputil -i -a E:\WIN8\AMD64\*.INF
```

6. To allow *Cloudbase-Init* to run scripts during an instance boot, set the PowerShell execution policy to be unrestricted:

```
C:\powershell
C:\Set-ExecutionPolicy Unrestricted
```

7. Download and install Cloudbase-Init:

```
C:\Invoke-WebRequest -UseBasicParsing http://www.cloudbase.it/downloads/
CloudbaseInitSetup_Beta_x64.msi -OutFile cloudbaseinit.msi
C:\.\cloudbaseinit.msi
```

   In the **configuration options** window, change the following settings:

   - Username: `Administrator`
   - Network adapter to configure: `Red Hat VirtIO Ethernet Adapter`
   - Serial port for logging: `COM1`

   When the installation is done, in the **Complete the Cloudbase-Init Setup Wizard** window, select the **Run Sysprep** and **Shutdown** check boxes and click **Finish**.

   Wait for the machine shutdown.

Your image is ready to upload to the Image Service:

```
$ glance image-create --name WS2012 --disk-format qcow2 \
  --container-format bare --is-public \
  --file ws2012.qcow2
```

Example: FreeBSD image

This example creates a minimal FreeBSD image that is compatible with OpenStack and bsd-cloudinit. The bsd-cloudinit program is independently maintained and in active development. The best source of information on the current state of the project is at http://pellaeon.github.io/bsd-cloudinit.

KVM with virtio drivers is used as the virtualization platform because that is the most widely used among OpenStack operators. If you use a different platform for your cloud virtualization, use that same platform in the image creation step.

This example shows how to create a FreeBSD 10 image. To create a FreeBSD 9.2 image, follow these steps with the noted differences.

To create a FreeBSD image

1. Make a virtual drive:

```
$ qemu-img create -f qcow2 freebsd.qcow2 1G
```

The minimum supported disk size for FreeBSD is 1 GB. Because the goal is to make the smallest possible base image, the example uses that minimum size. This size is sufficient to include the optional `doc`, `games`, and `lib32` collections. To include the `ports` collection, add another 1 GB. To include `src`, add 512 MB.

2. Get the installer ISO:

```
$ curl ftp://ftp.freebsd.org/pub/FreeBSD/releases\
/amd64/amd64/ISO-IMAGES/10.0/FreeBSD-10.0-RELEASE-amd64-bootonly.iso >\
FreeBSD-10.0-RELEASE-amd64-bootonly.iso
```

3. Launch a VM on your local workstation. Use the same hypervisor, virtual disk, and virtual network drivers as you use in your production environment.

   The following command uses the minimum amount of RAM, which is 128 MB:

```
$ kvm -smp 1 -m 128 -cdrom FreeBSD-10.0-RELEASE-amd64-bootonly.iso \
-drive if=virtio,file=freebsd.qcow2 \
-net nic,model=virtio -net user
```

   You can specify up to 1 GB additional RAM to make the installation process run faster.

   This VM must also have Internet access to download packages.

Note

By using the same hypervisor, you can ensure that you emulate the same devices that exist in production. However, if you use full hardware virtualization instead of paravirtualization, you do not need to use the same hypervisor; you must use the same type of virtualized hardware because FreeBSD device names are related to their drivers. If the name of your root block device or primary network interface in production differs than the names used during image creation, errors can occur.

   You now have a VM that boots from the downloaded install ISO and is connected to the blank virtual disk that you created previously.

4. To install the operating system, complete the following steps inside the VM:

   a. When prompted, choose to run the ISO in **Install** mode.

   b. Accept the default keymap or select an appropriate mapping for your needs.

   c. Provide a host name for your image. If you use bsd-cloudinit, it overrides this value with the name provided by OpenStack when an instance boots from this image.

   d. When prompted about the optional `doc`, `games`, `lib32`, `ports`, and `src` system components, select only those that you need. It is possible to have a fully functional installation without selecting additional components selected. As noted previously, a minimal system with a 1 GB virtual disk supports `doc`, `games`, and `lib32` inclusive. The `ports` collection requires at least 1 GB additional space and possibly more if you plan to install many ports. The `src` collection requires an additional 512 MB.

e. Configure the primary network interface to use DHCP. In this example, which uses a virtio network device, this interface is named `vtnet0`.

f. Accept the default network mirror.

g. Set up disk partitioning.

Disk partitioning is a critical element of the image creation process and the auto-generated default partitioning scheme does not work with bsd-cloudinit at this time.

Because the default does not work, you must select manual partitioning. The partition editor should list only one block device. If you use virtio for the disk device driver, it is named `vtbd0`. Select this device and run the **create** command three times:

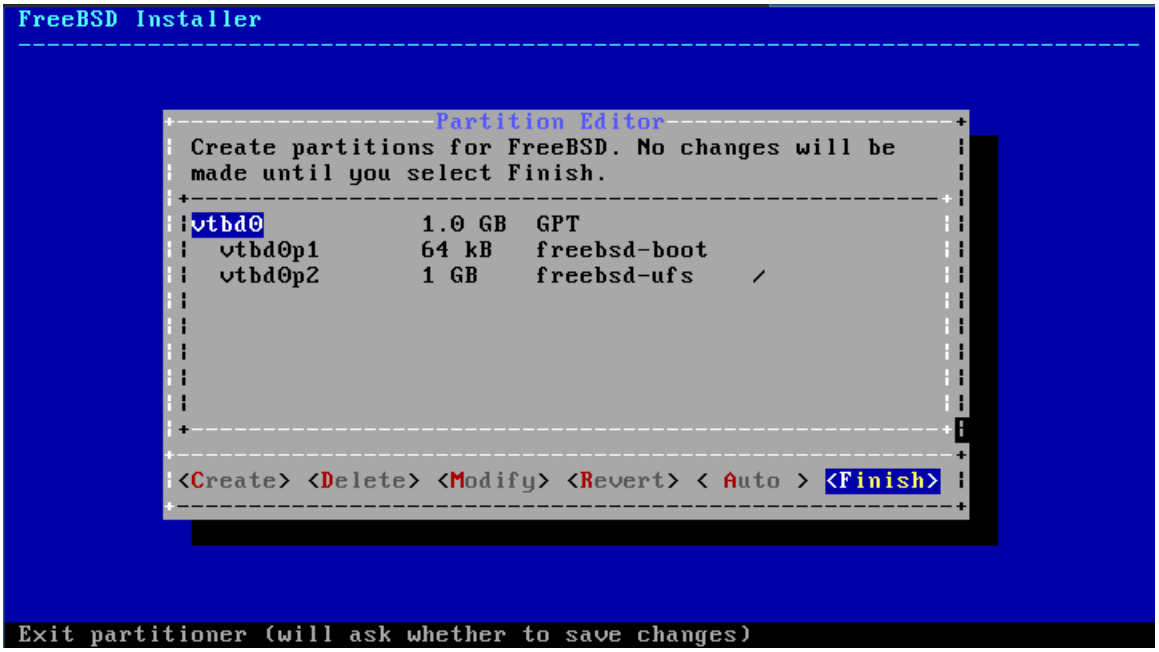
1. Select **Create** to create a partition table. This action is the default when no partition table exists. Then, select **GPT GUID Partition Table** from the list. This choice is the default.

2. Create two partitions:

   - First partition: A 64 kB `freebsd-boot` partition with no mount point.

   - Second partition: A `freebsd-ufs` partition with a mount point of `/` with all remaining free space.

The following figure shows a completed partition table with a 1 GB virtual disk:



Select **Finish** and then **Commit** to commit your changes.

Note

If you modify this example, the root partition, which is mounted on `/`, must be the last partition on the drive so that it can expand at run time to the disk size that your instance type provides. Also note that bsd-cloudinit currently has a hard-coded assumption that this is the second partition.

5. Select a root password.
6. Select the CMOS time zone.

   The virtualized CMOS almost always stores its time in UTC, so unless you know otherwise, select UTC.
7. Select the time zone appropriate to your environment.
8. From the list of services to start on boot, you must select `ssh`. Optionally, select other services.
9. Optionally, add users.

   You do not need to add users at this time. The bsd-cloudinit program adds a `freebsd` user account if one does not exist. The `ssh` keys for this user are associated with OpenStack. To customize this user account, you can create it now. For example, you might want to customize the shell for the user.
10. **Final config**

    This menu enables you to update previous settings. Check that the settings are correct, and click **exit**.
11. After you exit, you can open a shell to complete manual configuration steps. Select **Yes** to make a few OpenStack-specific changes:

    a. Set up the console:

    ```
    # echo 'console="comconsole,vidconsole"' >> /boot/loader.conf
    ```

    This sets console output to go to the serial console, which is displayed by **nova consolelog**, and the video console for sites with VNC or Spice configured.

    b. Minimize boot delay:

    ```
    # echo 'autoboot_delay="1"' >> /boot/loader.conf
    ```

    c. Download the latest bsd-cloudinit-installer. The download commands differ between FreeBSD 10.0 and 9.2 because of differences in how the **fetch** command handles HTTPS URLs.

    In FreeBSD 10.0 the **fetch** command verifies SSL peers by default, so you need to install the ca_root_nss package that contains certificate authority root certificates and tell **fetch** where to find them. For FreeBSD 10.0 run these commands:

    ```
    # pkg install ca_root_nss
    ```

```
# fetch --ca-cert=/usr/local/share/certs/ca-root-nss.crt \
https://raw.github.com/pellaeon/bsd-cloudinit-installer/master/
installer.sh
```

FreeBSD 9.2 **fetch** does not support peer-verification for https. For FreeBSD 9.2, run this command:

```
# fetch https://raw.github.com/pellaeon/bsd-cloudinit-installer/
master/installer.sh
```

d. Run the installer:

```
# sh ./installer.sh
```

The installer installs necessary prerequisites and downloads and installs the latest bsd-cloudinit.

e. Install sudo and configure the `freebsd` user to have passwordless access:

```
# pkg install sudo
# echo 'freebsd ALL=(ALL) NOPASSWD: ALL' > /usr/local/etc/sudoers.d/
10-cloudinit
```

12. Power off the system:

```
# shutdown -s now
```

6. Tool support for image creation

There are several tools that are designed to automate image creation.

Oz

Oz is a command-line tool that automates the process of creating a virtual machine image file. Oz is a Python app that interacts with KVM to step through the process of installing a virtual machine. It uses a predefined set of kickstart (Red Hat-based systems) and preseed files (Debian-based systems) for operating systems that it supports, and it can also be used to create Microsoft Windows images. On Fedora, install Oz with yum:

```
# yum install oz
```

Note

As of this writing, there are no Oz packages for Ubuntu, so you will need to either install from source or build your own .deb file.

A full treatment of Oz is beyond the scope of this document, but we will provide an example. You can find additional examples of Oz template files on GitHub at rackerjoe/oz-image-build/templates. Here's how you would create a CentOS 6.4 image with Oz.

Create a template file (we'll call it `centos64.tdl`) with the following contents. The only entry you will need to change is the `<rootpw>` contents.

```
<template>
  <name>centos64</name>
  <os>
    <name>CentOS-6</name>
    <version>4</version>
    <arch>x86_64</arch>
    <install type='iso'>
      <iso>http://mirror.rackspace.com/CentOS/6/isos/x86_64/CentOS-6.4-x86_64-
bin-DVD1.iso</iso>
    </install>
    <rootpw>CHANGE THIS TO YOUR ROOT PASSWORD</rootpw>
  </os>
  <description>CentOS 6.4 x86_64</description>
  <repositories>
    <repository name='epel-6'>
      <url>http://download.fedoraproject.org/pub/epel/6/$basearch</url>
      <signed>no</signed>
    </repository>
  </repositories>
```

```
  <packages>
    <package name='epel-release'/>
    <package name='cloud-utils'/>
    <package name='cloud-init'/>
  </packages>
  <commands>
    <command name='update'>
yum update
yum clean all
sed -i '/^HWADDR/d' /etc/sysconfig/network-scripts/ifcfg-eth0
echo -n > /etc/udev/rules.d/70-persistent-net.rules
echo -n > /lib/udev/rules.d/75-persistent-net-generator.rules
    </command>
  </commands>
</template>
```

This Oz template specifies where to download the Centos 6.4 install ISO. Oz will use the version information to identify which kickstart file to use. In this case, it will be RHEL6.auto. It adds EPEL as a repository and install the `epel-release`, `cloud-utils`, and `cloud-init` packages, as specified in the `packages` section of the file.

After Oz does the initial OS install using the kickstart file, it customizes the image by doing an update. It also removes any reference to the eth0 device that libvirt creates while Oz does the customizing, as specified in the `command` section of the XML file.

To run this, do, as root:

```
# oz-install -d3 -u centos64.tdl -x centos64-libvirt.xml
```

- The `-d3` flag tells Oz to show status information as it runs.
- The `-u` tells Oz to do the customization (install extra packages, run the commands) once it does the initial install.
- The `-x <filename>` flag tells Oz what filename to use to write out a libvirt XML file (otherwise it will default to something like `centos64Apr_03_2013-12:39:42`).

If you leave out the `-u` flag, or you want to edit the file to do additional customizations, you can use the **oz-customize** command, using the libvirt XML file that **oz-install** creates. For example:

```
# oz-customize -d3 centos64.tdl centos64-libvirt.xml
```

Oz will invoke libvirt to boot the image inside of KVM, then Oz will ssh into the instance and perform the customizations.

VMBuilder

VMBuilder (Virtual Machine Builder) is a command-line tool that creates virtual machine images for different hypervisors. The version of VMBuilder that ships with Ubuntu can only create Ubuntu virtual machine guests. The version of VMBuilder that ships with Debian can create Ubuntu and Debian virtual machine guests.

The *Ubuntu Server Guide* has documentation on how to use VMBuilder to create an Ubuntu image.

BoxGrinder

BoxGrinder is another tool for creating virtual machine images, which it calls appliances. BoxGrinder can create Fedora, Red Hat Enterprise Linux, or CentOS images. BoxGrinder is currently only supported on Fedora.

VeeWee

VeeWee is often used to build Vagrant boxes, but it can also be used to build KVM images.

Packer

Packer is a tool for creating machine images for multiple platforms from a single source configuration.

imagefactory

imagefactory is a newer tool designed to automate the building, converting, and uploading images to different cloud providers. It uses Oz as its back-end and includes support for OpenStack-based clouds.

SUSE Studio

SUSE Studio is a web application for building and testing software applications in a web browser. It supports the creation of physical, virtual or cloud-based applications and includes support for building images for OpenStack based clouds using SUSE Linux Enterprise and openSUSE as distributions.

7. Converting between image formats

Converting images from one format to another is generally straightforward.

qemu-img convert: raw, qcow2, VDI, VMDK

The **qemu-img convert** command can do conversion between multiple formats, including raw, qcow2, VDI (VirtualBox), VMDK (VMware) and VHD (Hyper-V).

Table 7.1. qemu-img format strings

Image format	Argument to qemu-img
raw	`raw`
qcow2	`qcow2`
VDI (VirtualBox)	`vdi`
VMDK (VMware)	`vmdk`
VHD (Hyper-V)	`vpc`

This example will convert a raw image file named centos63.dsk to a qcow2 image file.

```
$ qemu-img convert -f raw -O qcow2 centos64.dsk centos64.qcow2
```

To convert from vmdk to raw, you would do:

```
$ qemu-img convert -f vmdk -O raw centos64.vmdk centos64.img
```

Note

The `-f format` flag is optional. If omitted, qemu-img will try to infer the image format.

VBoxManage: VDI (VirtualBox) to raw

If you've created a VDI image using VirtualBox, you can convert it to raw format using the **VBoxManage** command-line tool that ships with VirtualBox. On Mac OS X, VirtualBox stores images by default in the `~/VirtualBox VMs/` directory. The following example creates a raw image in the current directory from a VirtualBox VDI image.

```
$ VBoxManage clonehd ~/VirtualBox\ VMs/fedora18.vdi fedora18.img --format raw
```

Appendix A. Community support

Table of Contents

The following resources are available to help you run and use OpenStack. The OpenStack community constantly improves and adds to the main features of OpenStack, but if you have any questions, do not hesitate to ask. Use the following resources to get OpenStack support, and troubleshoot your installations.

Documentation

For the available OpenStack documentation, see docs.openstack.org.

To provide feedback on documentation, join and use the <openstack-docs@lists.openstack.org> mailing list at OpenStack Documentation Mailing List, or report a bug.

The following books explain how to install an OpenStack cloud and its associated components:

- *Installation Guide for Debian 7.0*
- *Installation Guide for openSUSE and SUSE Linux Enterprise Server*
- *Installation Guide for Red Hat Enterprise Linux, CentOS, and Fedora*
- *Installation Guide for Ubuntu 14.04 (LTS)*

The following books explain how to configure and run an OpenStack cloud:

- *Architecture Design Guide*
- *Cloud Administrator Guide*
- *Configuration Reference*
- *Operations Guide*
- *High Availability Guide*
- *Security Guide*

- *Virtual Machine Image Guide*

The following books explain how to use the OpenStack dashboard and command-line clients:

- *API Quick Start*
- *End User Guide*
- *Admin User Guide*
- *Command-Line Interface Reference*

The following documentation provides reference and guidance information for the OpenStack APIs:

- OpenStack API Complete Reference (HTML)
- API Complete Reference (PDF)
- *OpenStack Block Storage Service API v2 Reference*
- *OpenStack Compute API v2 and Extensions Reference*
- *OpenStack Identity Service API v2.0 Reference*
- *OpenStack Image Service API v2 Reference*
- *OpenStack Networking API v2.0 Reference*
- *OpenStack Object Storage API v1 Reference*

The *Training Guides* offer software training for cloud administration and management.

ask.openstack.org

During the set up or testing of OpenStack, you might have questions about how a specific task is completed or be in a situation where a feature does not work correctly. Use the ask.openstack.org site to ask questions and get answers. When you visit the http://ask.openstack.org site, scan the recently asked questions to see whether your question has already been answered. If not, ask a new question. Be sure to give a clear, concise summary in the title and provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.

OpenStack mailing lists

A great way to get answers and insights is to post your question or problematic scenario to the OpenStack mailing list. You can learn from and help others who might have similar issues. To subscribe or view the archives, go to http://lists.openstack.org/cgi-bin/mailman/listinfo/openstack. You might be interested in the other mailing lists for specific projects or development, which you can find on the wiki. A description of all mailing lists is available at http://wiki.openstack.org/MailingLists.

The OpenStack wiki

The OpenStack wiki contains a broad range of topics but some of the information can be difficult to find or is a few pages deep. Fortunately, the wiki search feature enables you to search by title or content. If you search for specific information, such as about networking or nova, you can find a large amount of relevant material. More is being added all the time, so be sure to check back often. You can find the search box in the upper-right corner of any OpenStack wiki page.

The Launchpad Bugs area

The OpenStack community values your set up and testing efforts and wants your feedback. To log a bug, you must sign up for a Launchpad account at https://launchpad.net/+login. You can view existing bugs and report bugs in the Launchpad Bugs area. Use the search feature to determine whether the bug has already been reported or already been fixed. If it still seems like your bug is unreported, fill out a bug report.

Some tips:

- Give a clear, concise summary.
- Provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.
- Be sure to include the software and package versions that you are using, especially if you are using a development branch, such as, `"Juno release" vs git commit bc79c3ecc55929bac585d04a03475b72e06a3208`.
- Any deployment-specific information is helpful, such as whether you are using Ubuntu 14.04 or are performing a multi-node installation.

The following Launchpad Bugs areas are available:

- Bugs: OpenStack Block Storage (cinder)
- Bugs: OpenStack Compute (nova)
- Bugs: OpenStack Dashboard (horizon)
- Bugs: OpenStack Identity (keystone)
- Bugs: OpenStack Image Service (glance)
- Bugs: OpenStack Networking (neutron)
- Bugs: OpenStack Object Storage (swift)
- Bugs: Bare Metal (ironic)
- Bugs: Data Processing Service (sahara)
- Bugs: Database Service (trove)

- Bugs: Orchestration (heat)
- Bugs: Telemetry (ceilometer)
- Bugs: Queue Service (marconi)
- Bugs: OpenStack API Documentation (developer.openstack.org)
- Bugs: OpenStack Documentation (docs.openstack.org)

The OpenStack IRC channel

The OpenStack community lives in the #openstack IRC channel on the Freenode network. You can hang out, ask questions, or get immediate feedback for urgent and pressing issues. To install an IRC client or use a browser-based client, go to http://webchat.freenode.net/. You can also use Colloquy (Mac OS X, http://colloquy.info/), mIRC (Windows, http://www.mirc.com/), or XChat (Linux). When you are in the IRC channel and want to share code or command output, the generally accepted method is to use a Paste Bin. The OpenStack project has one at http://paste.openstack.org. Just paste your longer amounts of text or logs in the web form and you get a URL that you can paste into the channel. The OpenStack IRC channel is `#openstack` on `irc.freenode.net`. You can find a list of all OpenStack IRC channels at https://wiki.openstack.org/wiki/IRC.

Documentation feedback

To provide feedback on documentation, join and use the `<openstack-docs@lists.openstack.org>` mailing list at OpenStack Documentation Mailing List, or report a bug.

OpenStack distribution packages

The following Linux distributions provide community-supported packages for OpenStack:

- **Debian:** http://wiki.debian.org/OpenStack
- **CentOS, Fedora, and Red Hat Enterprise Linux:** http://openstack.redhat.com/
- **openSUSE and SUSE Linux Enterprise Server:** http://en.opensuse.org/Portal:OpenStack
- **Ubuntu:** https://wiki.ubuntu.com/ServerTeam/CloudArchive