QVD Disk Image Creation (Ubuntu 12.04)

QVD DOCUMENTATION

<documentation@theqvd.com>

November 5, 2018

Contents

Introduction

The QVD provides demo desktop images for use with the product, but sooner or later the serious user will need to consider building their own desktops, tailored to the needs of their particular users. This guide aims to explain that process in its simplest form, and to give the administrator the basic building blocks for their own desktop images.

The process itself is fairly straightforward and entails the following steps to create KVM and LXC images:

- Installing a clean build system in KVM
- Installing inside that build system LXC userspace utils and creating a container (LXC only)
- Configuring either the build system or the LXC container to run the QVD Virtual Machine Agent which manages the connections between the QVD client and the desktop
- Customizing and branding the desktop to the needs of the user
- Shutting down and zipping up the container or the KVM install itself
- Deploying the zipped image to QVD using either the Web Administration Tool or the qvd admin command line tool

Building the Image

Install a Build System

To create a disk image for either KVM or LXC, you will need to install a base image to work from. This base image will form the basis of the KVM image if that is what you are creating, for LXC it will provide a system from which to build and zip an LXC image.

We will use the the lightweight Ubuntu variant Lubuntu to create the images as it is relatively small and comes with the minimal desktop LXDE as standard. We will use the the alternate download cd as it's optimized for lower RAM installations and stick with the stable LTS release Precise Pangolin for purposes of stability, and for compatibility with LXC.

The iso can be found on the Lubuntu Precise Alternate download page, or using this direct link. You can check the md5sum of the download here. Before we start, you will need to ensure that you have both *qemu-kvm* and *qemu-utils* installed.

```
$ sudo apt-get install qemu-kvm qemu-utils
```

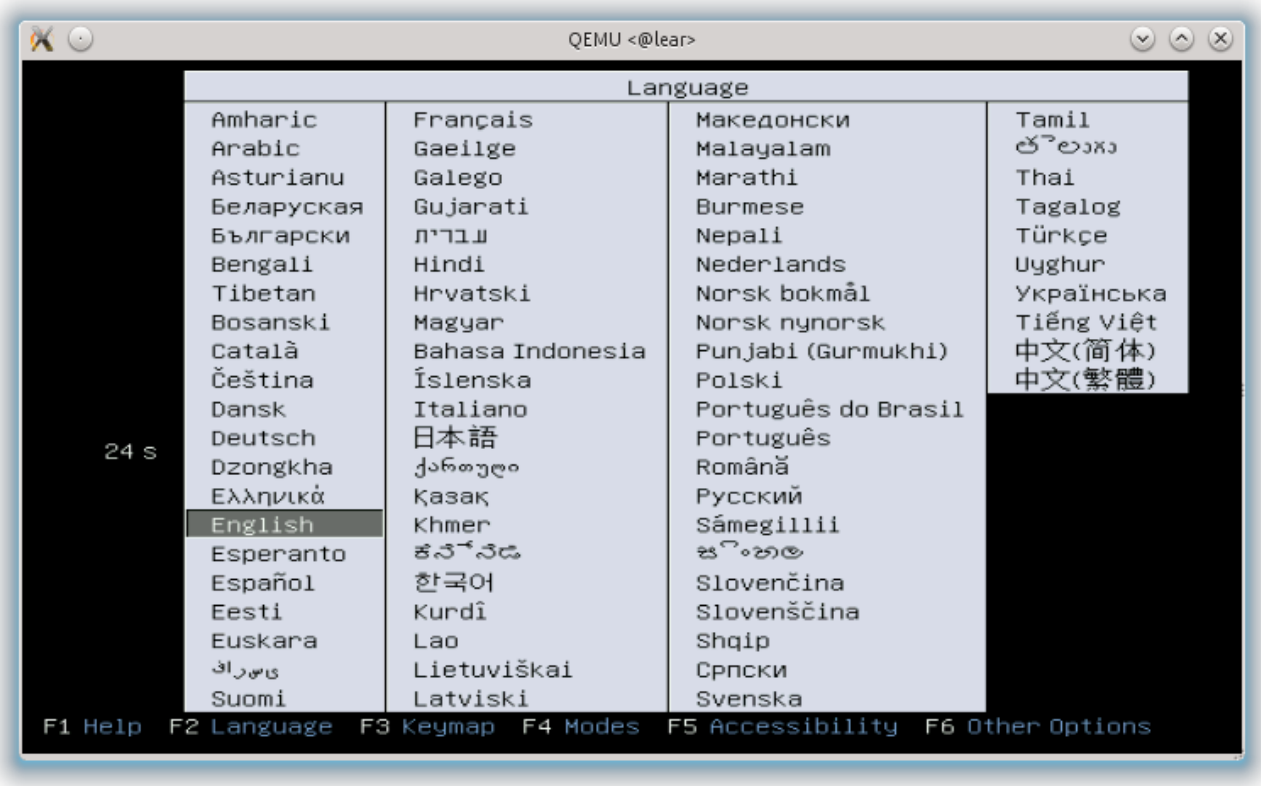
Now, create a qcow2 disk image file, setting the maximum file size at 4.5GB which is the minimum required by Lubuntu. Note, however, that initially the file will be a fraction of this size as it will grow dynamically to accomodate needs. Indeed the final result should be considerably smaller and even less when compressed.

```
$ kvm-img create -f qcow2 qvd.img 4.5G
```

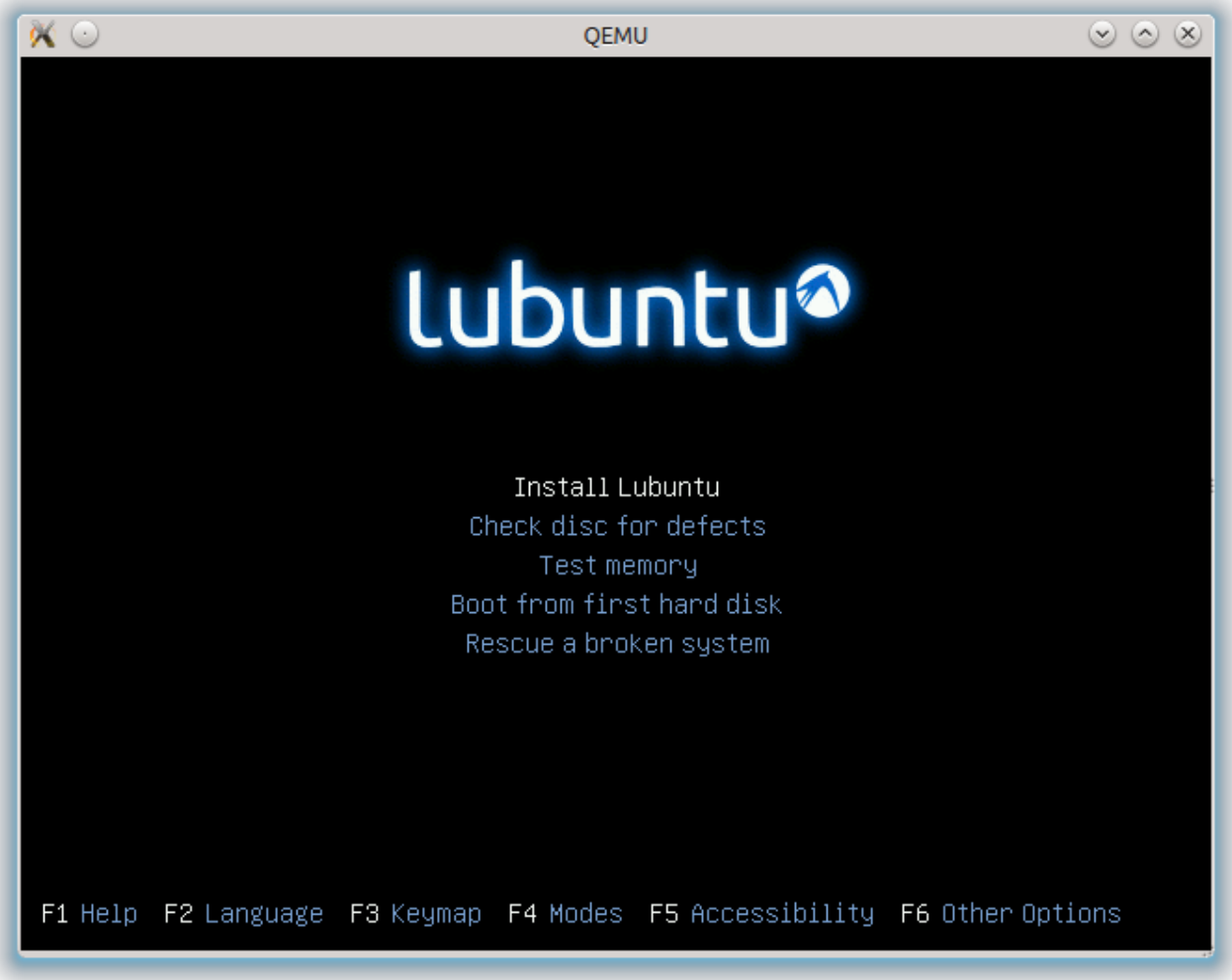
Now start the image with the lubuntu iso as your cdrom device to get the install underway and with a mere 512MB RAM:

```
$ kvm -hda qvd.img -cdrom lubuntu-12.04-alternate-amd64.iso -m 512
```

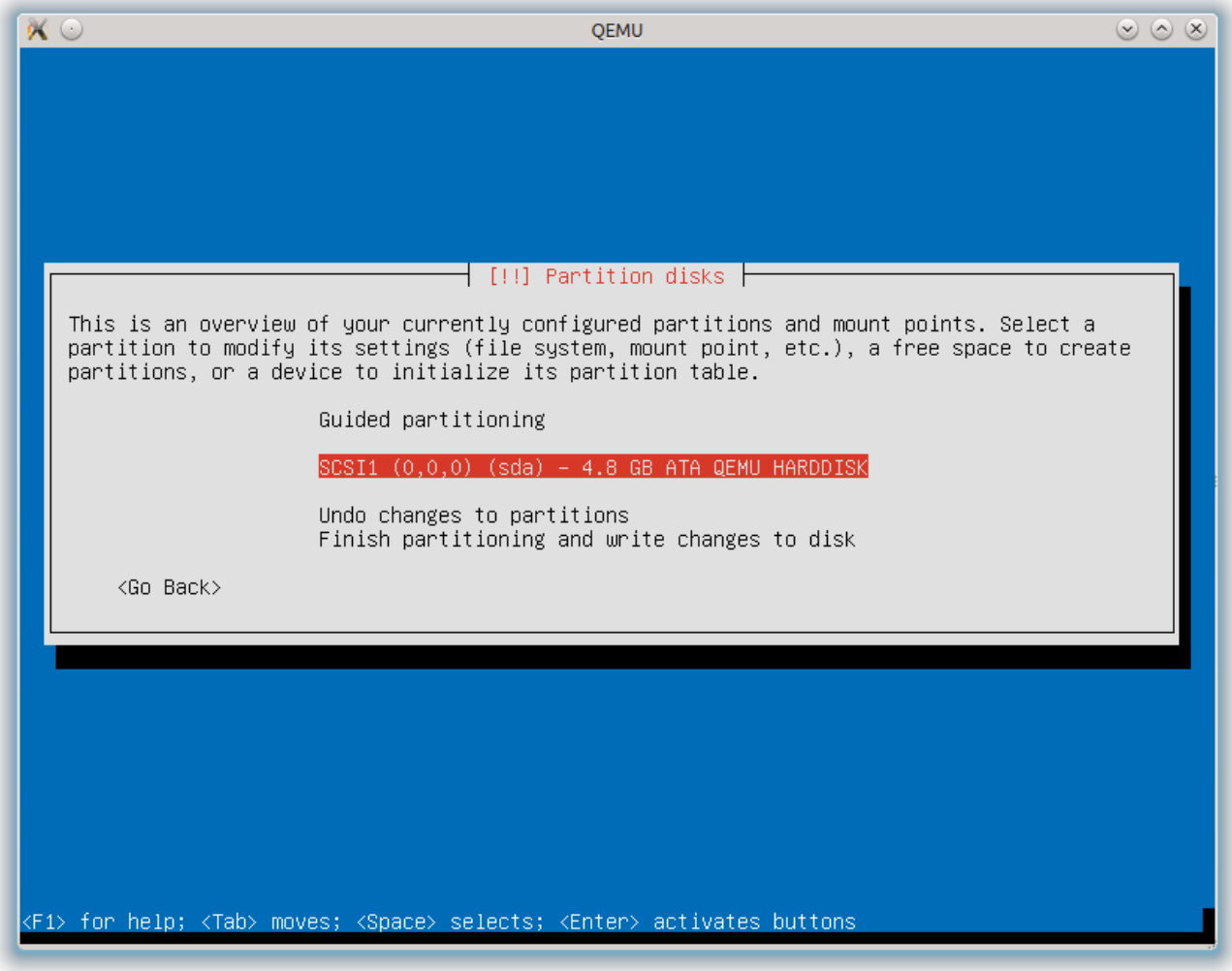
Obviously if your iso is in a different folder to this image, you will need to change your path accordingly. This will bring up the Lubuntu installer.



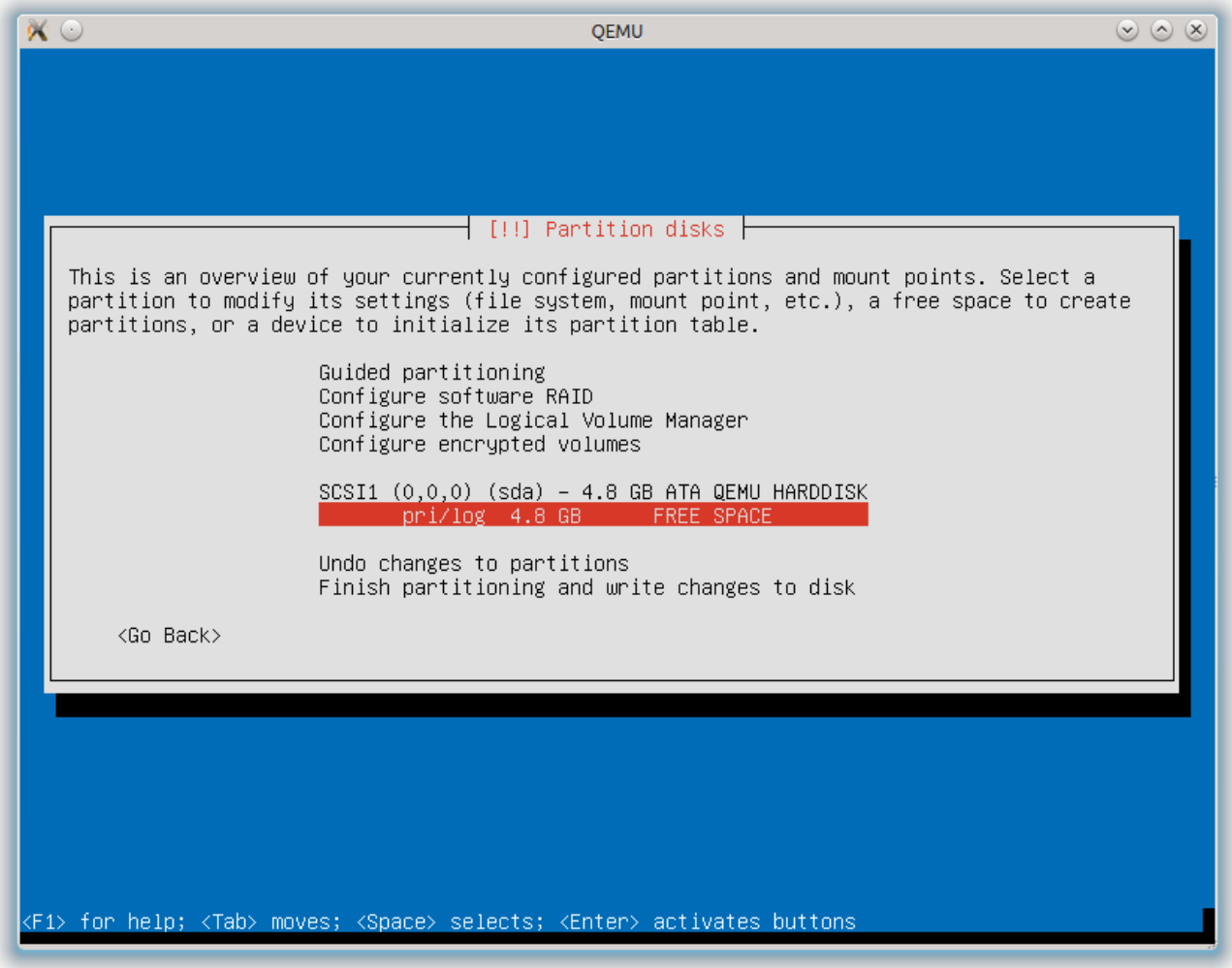
Select your language and in the next screen select *Install Lubuntu*.



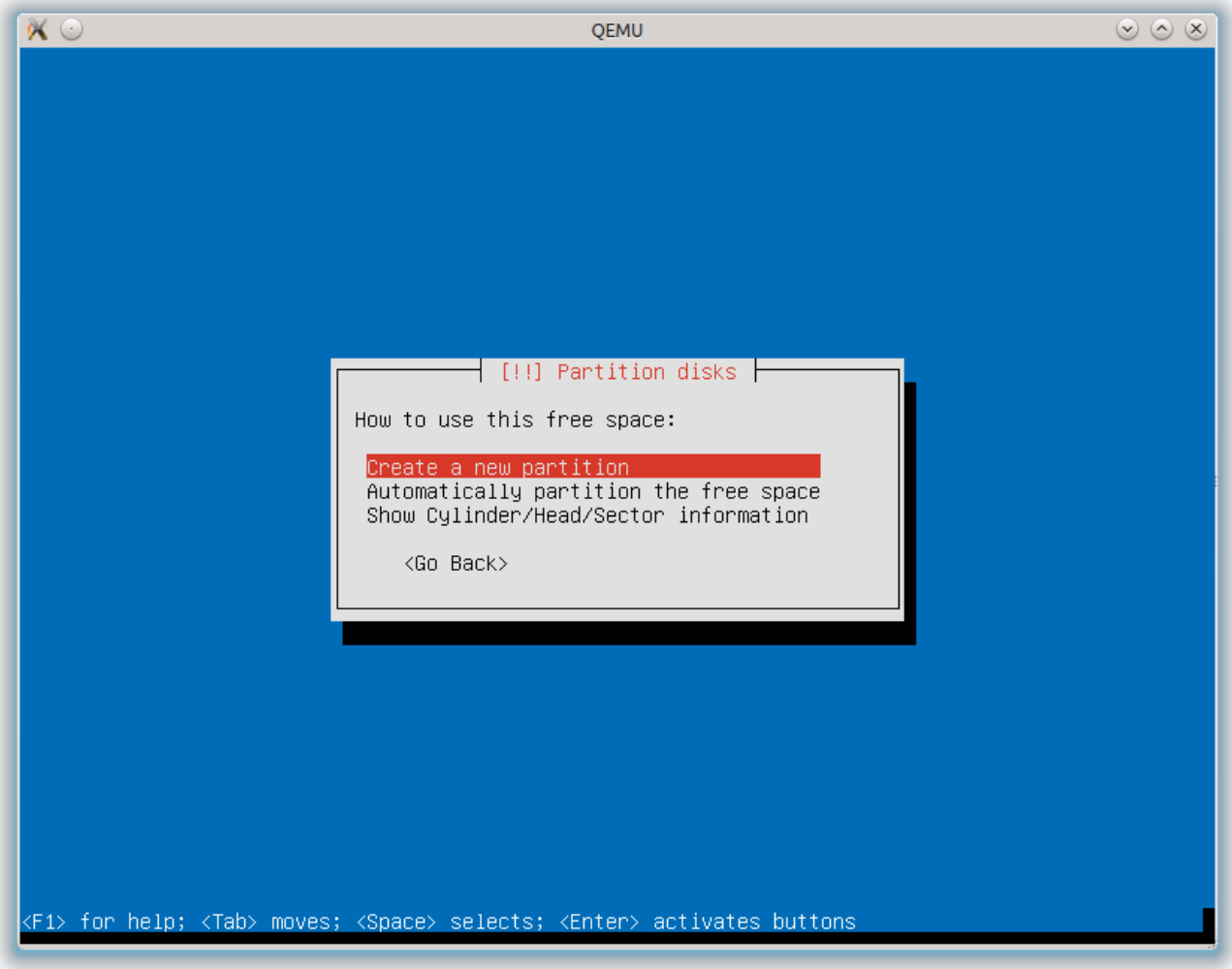
Most of the install is pretty self explanatory, so we won't cover it in too much depth. Work through the steps until you reach disk partitioning. As a rule we don't want or need a swap partition within disk images in QVD, so let's select the *Manual* option. Now scroll down to *sda* in the next screen and hit enter.



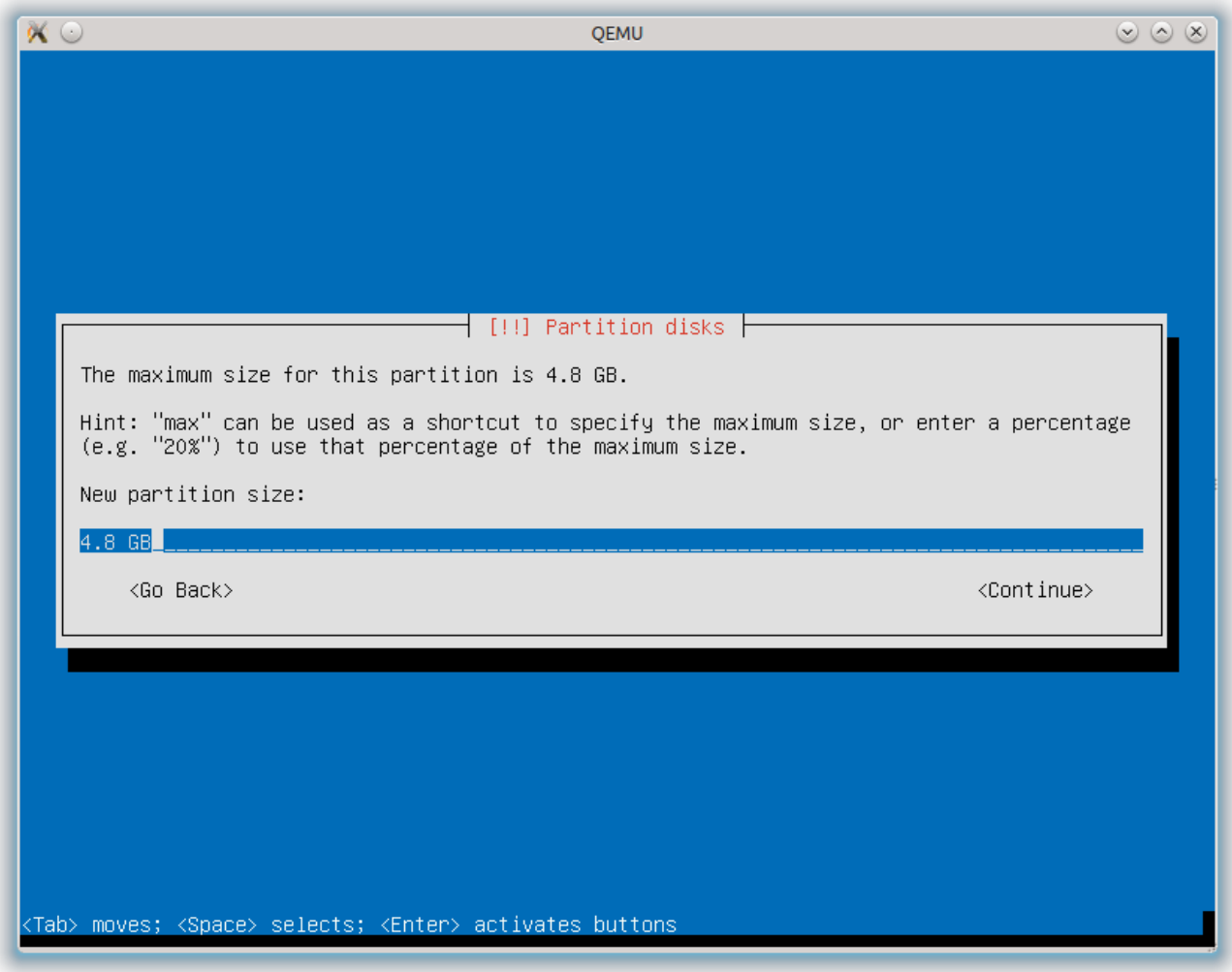
Yes to *Create New Empty Partition Table On This Device?* You'll now be back in the main partition screen with a new, empty, partition table. Select this empty partition.



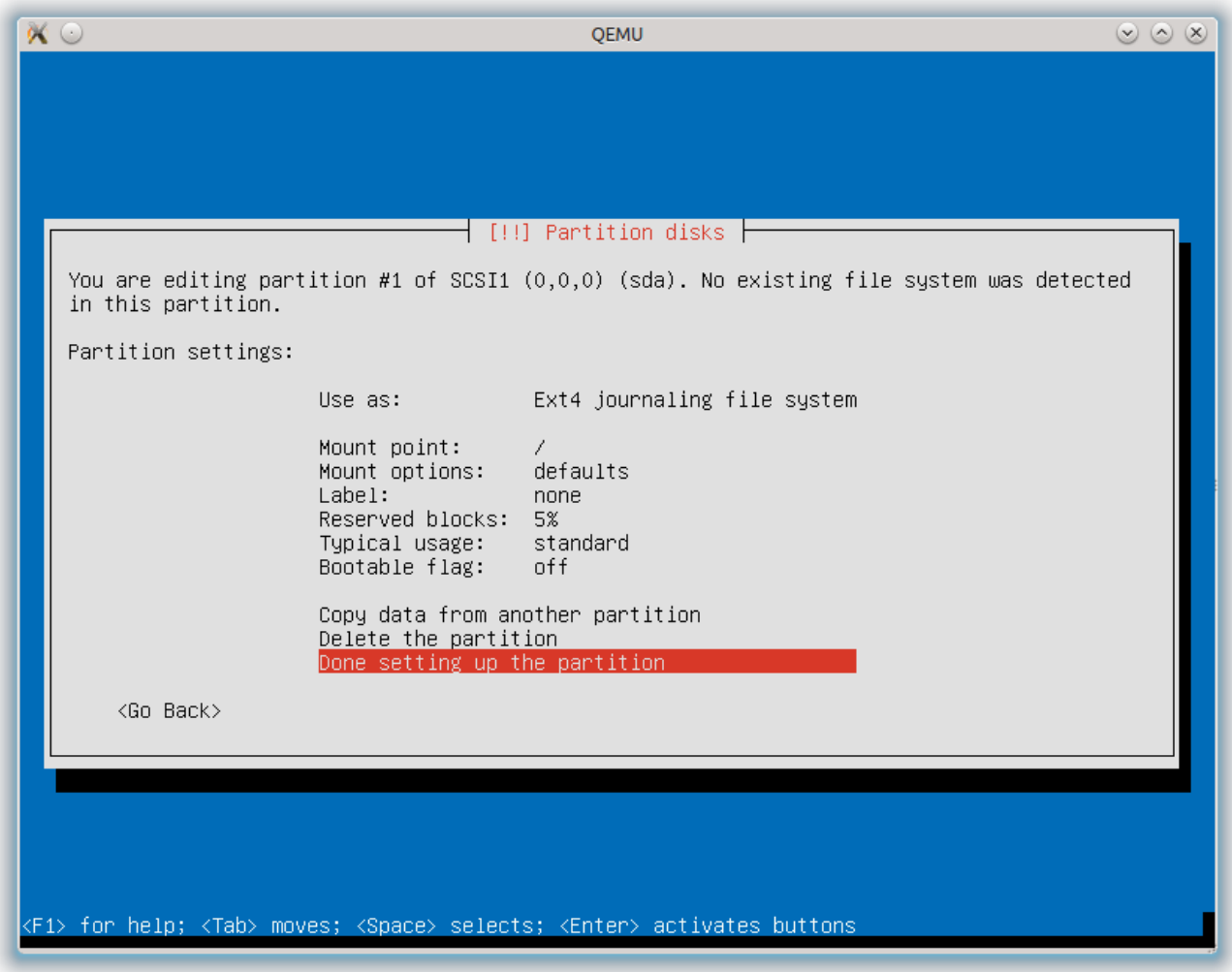
Next opt to create a new partition.



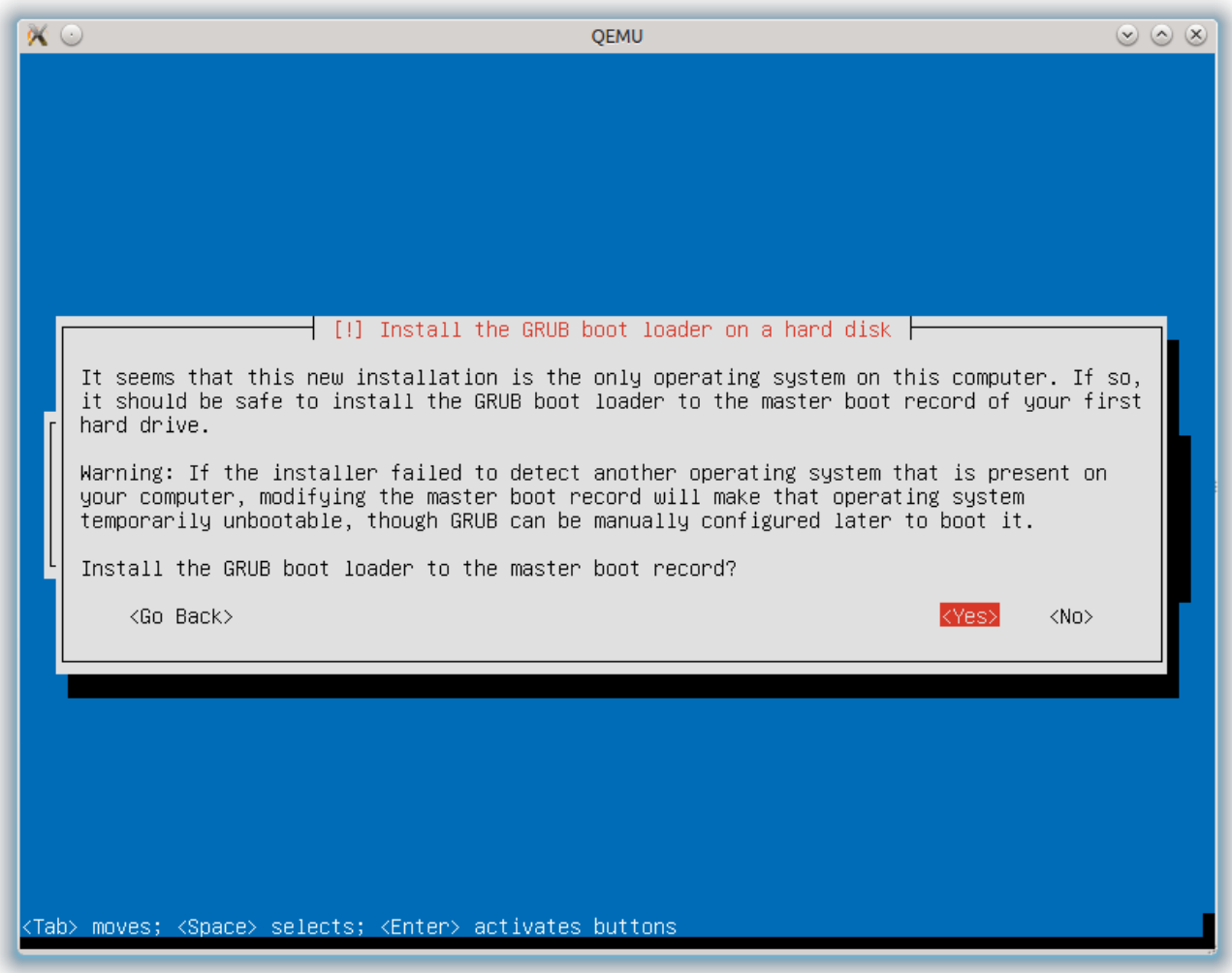
Agree to the partition size (it should be the full size of the disk).



Select *Primary* partition and you will be in the partition disks page with a bunch of sensible defaults. These look fine, so choose *Done Setting Up The Partition*.

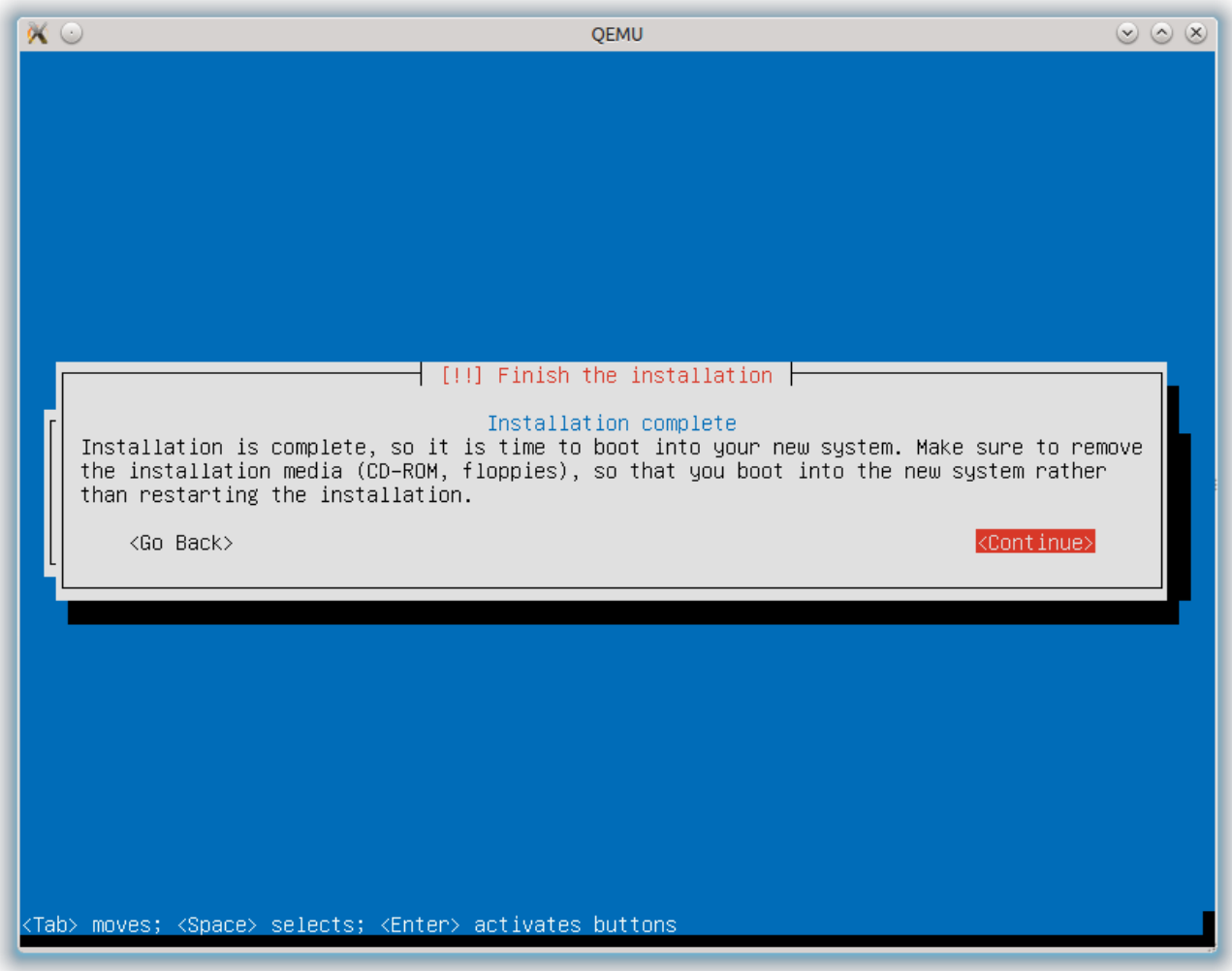


Back in the main screen, select *Finish* and ignore the warning about lack of swap and writing changes to disk. Thus will begin the installing of the core packages of the OS. This will take a few minutes. Once that is finished you will get a question about proxy setup which you will need to answer (or just *Continue*) and this will be followed by apt setup and software install. At the end of the install, you will need to install GRUB to the MBR.



The install will prompt you for the details of an indiviudal user for your system. This user will be your administrative user for KVM and indeed inside the VDI as well as being the account you use to create LXC images if that is your aim.

Finally, set up the time zone to complete the installation.



Select *Continue* to boot into the new OS. and log in using the user credentials that you set during the install. At this point, the install process diverges depending on whether you want to create a KVM image, or an LXC one. If the former, continue on to the next section, for the latter, skip to the Create an LXC Disk Image section.

Create a KVM Disk Image

As stated previously, the KVM install we are working in is going to be the basis of the image that you create. That is to say, everything that is installed and customized inside this VM will be a part of the desktop environment you create for your users. The user you have created will not be a desktop user, rather the administrative user that you can use to maintain and install software on the image to make available for your users.

include:DiskImageCreationUbuntu/install_vma_ssh_serial_ubuntu.adoc[]

Customize and Brand the Desktop

From here, it's up to you. Install system updates and the packages that you would like available for your users. This may involve the packages available for your distro, it may be the custom software that your company uses. Ready the desktop for use as you would like your users to experience it, and once you are happy with your disk image, power it down and it's ready for use with QVD.

Tar and Zip the Image

QVD accepts compressed tar archives and the image should compress substantially to make is quicker to transfer to other hosts, so you should tar and zip the KVM image as follows:

```
$ tar czf qvd.img.tar.gz qvd.img
```

And unzip it on the destination server when you are ready to move it to staging:

```
$ cd /var/lib/qvd/staging && sudo tar zxf /path/to/qvd.img.tar.gz
```

Once you've moved the image into staging, it will be available in the QVD Web Administration Tool. You can create a new set of runtime settings for the image (OSF) and add the disk image to the database. Finally, you will need to create a new Virtual Machine for each user that will use this disk image and it's ready to be used. And that concludes our little introduction to creating KVM images for use with QVD. It's pretty simple really, and the beauty of the KVM image is that any time you want to tweak an image, you can open it up in kvm and administer it as though it were a real desktop, shut down, copy the image to staging and deploy.

Create an LXC Disk Image

LXC is a fairly recent but rapidly stabilising technology that allows for running multiple isolated Linux systems (containers) on a single host. Because these containers utilise the same kernel as the host, it's an extremely efficient method of virtualisation that allows for a greater VM density than KVM. That said, it requires a that the host and the guests are compatible that with the same kernel, and for this reason it is strongly recommended that you use the same distro and version for your clients as for your node. Hence the requirement that we use Lubuntu 12.04 for this document.

As stated earlier, the KVM system you have just installed will not itself be the basis for the QVD image, but will be used to install LXC and create a suitable environment for your desktop users within the KVM image itself.

Install the LXC User Space Tools

Once logged into your kvm build environment, su to root, and install the LXC user space tools:

```
# apt-get install lxc
```

Create a Basic LXC Configuration

That done, create a file called lxc.conf somewhere convenient, and populate it with the following network settings:

```
lxc.network.type=veth
lxc.network.link=lxcbr0
lxc.network.flags=up
```

The first line will tell lxc that we wish to use a virtual ethernet device inside the container. The second line gives lxc the name of the network bridge that we want the container to utilise to connect to the network. Finally, we set the network as up.

Create and Run the LXC Container

Now let's create a new Linux container:

```
# lxc-create -n qvd-vm -t ubuntu -f lxc.conf
```

The first time you do this it will take a little extra time compared to subsequent builds as it will pull down the core packages from the Ubuntu servers to install inside the container. The next time you create a container using the same template (in this case *ubuntu*), these packages will be reused so it will be quite a bit quicker.

Note
As there is no explicit Lubuntu LXC template we use the Ubuntu one which contains the same basic core packages, and add the LXDE desktop at a later stage.

Once this is done, start the container:

```
# lxc-start -n qvd-vm
```

This will both start the contained and put you straight into the login prompt. Log in using *ubuntu* as both username and password.

Install the QVD Virtual Machine Agent

The QVD Virtual Machine Agent (VMA) is responsible for accepting connections from the client via a QVD Server Node. It facilitates access to the desktop environment running within the virtual machine, including the ability to configure printer access and to configure the virtual machine to stream audio to the client.

Firstly, add the QVD packages public key to your trusted keys (as root):

```
# wget -qO - https://www.theqvd.com/packages/key/public.key | sudo apt-key add -
```

Now, add the repository:

```
# echo "deb http://theqvd.com/packages/ubuntu-trusty QVD-3.5.0 main" > \
/etc/apt/sources.list.d/qvd-35.list
# apt-get update
```

Finally, install the VMA:

```
# apt-get install perl-qvd-vma
```

It is essential that the Virtual Machine Agent starts on boot or the desktop will not be accessible. To do that, we add a symlink to runlevel 2:

```
# ln -s /etc/init.d/qvd-vma /etc/rc2.d/S99qvd-vma
```

The VMA is controlled through the file */etc/qvd/vma.conf*. This file is not created on install, so let's create the directory, and copy the sample VMA configuration file into place.

```
# mkdir /etc/qvd
# cp -v /usr/lib/qvd/config/sample-vma.conf /etc/qvd/vma.conf
```

If you take a look at the vma.conf file it has a couple of sensible presets:

```
vma.audio.enable = 1
vma.printing.enable = 1
```

Other settings are detailed in the QVD Administration Manual. One additional configuration option is needed to get the X session running under Lubuntu, although it should be noted that this requirement isn't necessary with a stock Ubuntu install:

```
command.x-session: /usr/bin/startlubuntu
```

Assuming you want audio and printing, leave those as they are and start the VMA:

```
$ sudo service qvd-vma start
```

Important
At the time of writing, *startlubuntu* does not honour the */etc/default/locale* settings, so if your locale needs to be changed from the default, create a wrapper script, for example in */usr/local/bin/startlubuntu*. In there, we source the */etc/default-/locale* and export the variables, only then invoking startlubuntu:

```
#!/bin/bash
[ -r /etc/default/locale ] && . /etc/default/locale
export LANG LANGUAGE LC_MESSAGES LC_ALL
/usr/bin/startlubuntu
```

Install and Configure Secure Shell and Serial Access

To administer the image once it's been installed, let's also install the openssh server:

```
$ sudo apt-get install openssh-server
```

When the image has been deployed in QVD, this can be accessed using the QVD admin tool for administration / troubleshooting:

```
$~qa vm ssh -f id=<vm_id> -- -l <user>
```

If you wish to have serial access to your VM, that too is readily set up. Within the DI, edit the file _/etc/init/ttyS0.conf_ and add the following configuration:

```
# ttyS0 - getty
#
# This service maintains a getty on ttyS0 from the point the system is
# started until it is shut down again.

start on stopped rc RUNLEVEL=[2345]
stop on runlevel [!2345]

respawn
exec /sbin/getty -L 115200 ttyS0 xterm
```

Once the Serial Port has been configured, the default settings for any Server Node will allow you to access a running Virtual Machine using telnet or the QVD CLI Administration Utility with a command like:

```
$ qa vm console -f id=<vm_id>
```

Customize and Brand the Desktop

Although LXC has copied the rudimentary system over to the container, the template does not allow for a desktop environment, so we will need to install that.

```
# apt-get install lubuntu-desktop
```

This would also be a good time to install any additional desktop software that your users might need, so go ahead and install any office software, graphics software and so on using apt as above.

Before we log out, create a new administrative user for the disk image, and delete the default user and home directory:

```
# adduser qvd
# usermod -aG sudo qvd
```

Log out and back in with the new user and (as root) remove the default user and home directory:

```
# deluser ubuntu
# rm -rf /home/ubuntu
```

Note
At this point we could delete the user with the --remove-home switch but this requires the perl-modules package which brings, along with dependencies, another 32MB to the install. It's up to to the use case and space constraints of course.

Branding and customization of the desktop environment is beyond the scope of this guide, but there is a list of resources at the end to assist with this. Once you have desktop that you want for your users, you can close down the container in preparation for zipping up the image.

```
# poweroff
```

Tar and Zip the Image

Back in your host machine tar and zip (as root) the root file system of the container for deployment in QVD:

```
#~tar czf ubuntu-12-04.tar.gz -C /var/lib/lxc/qvd-vm/rootfs .
```

Deploying the Image

By now you should have your gzipped and tarred image, either KVM or LXC. Upload it to your node server if you have built it on another machine, and then copy the file over to the directory */var/lib/qvd/storage/staging* which it will be available as an *image file* in the QVD Web Administration Tool, ready for deployment as a Disk Image.

Once you have created a new disk image in QVD, using your uploaded disk image and an *Operating System Flavor* which consists of a few runtime settings such as the amount of RAM you wish to allocate and the allotted disk space, go into the *Users* tab in the WAT and add machines to your users. Rollout to multiple users can also be achieved using hooks within the QVD but that is beyond the scope of this guide.

And that concludes our creation of an Ubuntu disk image for QVD. As you can see, it's a really quick and easy process and allows for almost endless customization of the desktop you wish to provide your end user. Another of the many benefits of choosing the QVD as your VDI solution.

Create an LXC Disk Image using Docker

LXC is a fairly recent but rapidly stabilising technology that allows for running multiple isolated Linux systems (containers) on a single host. Around LXC some new technologies have appeared such as Docker that permit an easy creation of new containers based on templates of existing containers.

QVD has created some base templates that you can use to create new LXC images. Currently two templates are available:

- theqvd/qvdimageubuntu:minimal. This is 155 MB image which only has an xterm and gives you the full power to install whatever you need
- theqvd/qvdimageubuntu:basicdesktop. This a 800MB image with XFCE environment + some basic office application such as LibreOffice, Firefox, Thunderbird and Evince

The examples assume that you are using Ubuntu 14.04

Software

You need a base Ubuntu 14.04 system with docker installed

```
$ sudo apt-get install docker.io
```

Create a custom image from the basicdesktop

Let's add the gimp image manipulation tool to the basic desktop image. For that create in a new directory a file called Dockerfile with the following content:

```
FROM theqvd/qvdimageubuntu:basicdesktop
MAINTAINER Me <me@domain.com>

ENV DEBIAN_FRONTEND noninteractive
RUN apt-get update && apt-get install -y gimp

# Cleanup
RUN echo "" > /etc/udev/rules.d/70-persistent-net.rules
RUN apt-get autoremove -y
RUN apt-get clean
```

Then run from the terminal the following commands:

```
$ cd <directory where the Dockerfile is>
$ sudo docker build -t myimage .
$ vmid=$(sudo docker run -d -t -i myimage /bin/bash)
$ sudo docker export $vmid | gzip -c > qvd.img.tar.gz
$ sudo docker kill $vmid
```

Once you got the qvd.img.tar.gz file, please follow the instructions above to create the di in QVD either via command line (qa di add) or via the web administration tool (WAT)

Please see http://www.docker.com for more information on other syntax options for Docker.

Create a custom image from the minimal image

In this case we create a minimal desktop based on the minimal image where we want to create a lubuntu desktop.

Create a new directory and this directory we are going to create two files:

- Dockerfile
- vma.conf

This is the content of the Dockerfile:

```
FROM theqvd/qvdimageubuntu:minimal
MAINTAINER Me <me@domain.com>

LABEL version="1.0"
LABEL description="This is a basic desktop Ubuntu VM image installation for QVD. It   ↩
    includes LibreOffice, Thunderbird, Firefox and Evince"

ENV DEBIAN_FRONTEND noninteractive
# packages
RUN echo "deb http://archive.canonical.com/ubuntu trusty partner" > /etc/apt/sources.list.d ↩
    /partners.list
RUN apt-get update && apt-get install -y \
  lubuntu-desktop \
  adobe-flashplugin \
  cups \
  evince \
  firefox \
  libreoffice \
  icedtea-7-plugin \
  thunderbird
# Config
COPY vma.conf /etc/qvd/vma.conf

# Cleanup
RUN echo "" > /etc/udev/rules.d/70-persistent-net.rules
RUN apt-get autoremove -y
RUN apt-get clean
```

This is the content of the vma.conf file:

```
command.x-session: /usr/bin/startlubuntu
vma.audio.enable: 1
vma.printing.enable: 1
```

Then create the tar.gz file by running the following commands:

```
$ cd <directory where the Dockerfile is>
$ sudo docker build -t myimage .
$ vmid=$(sudo docker run -d -t -i myimage /bin/bash)
$ sudo docker export $vmid | gzip -c > qvd.img.tar.gz
$ sudo docker kill $vmid
```

Once you got the qvd.img.tar.gz file, please follow the instructions above to create the di in QVD either via command line (qa di add) or via the web administration tool (WAT)

Please see http://www.docker.com for more information on other syntax options for Docker.

Appendix

Resources

VMA HOOKS

Guide to LXPanel on the LXDE

Guide to PCManFM

To make your customization and branding global, rather than use the home directory for your user account (which is really just your administrative account for the image), changes need to be made in the */etc/skel* directory, which acts as a template for new

users' home directories when new users are added. QVD conforms to this practice, so each new Virtual Machine will create a home directory based on this folder in the Disk Image. For the PCManFM use */etc/skel/.config/pcmanfm/lubuntu/pcmanfm.conf* and for LXPanel use */etc/skel/.config/lxpanel/Lubuntu/panels/panel*.

Logging Remotely

Logging remotely to a daemon supporting the syslog protocol can be desirable for a couple of reasons. Firstly, it keeps the logs for all the virtual machines that have been configured thus in one place which makes accessing the logs easier and more logical. Secondly, in a situation where the administrator may not be able to access a virtual machine's log for some reason, for example if it is not starting up, logging remotely might help in identifying the problem.

To set up remote logging, you will need a configured remote logging server, and to make some changes within your disk image, both to QVD's vma.conf file, and to the syslog settings to send any syslog messages on to the remote logging server.

To demonstrate we will use Rsyslog which has become the default logging utility for many of the major Linux distributions over recent years, including Ubuntu, SUSE, and Redhat, and is reliable and easy to set up. Because QVD uses Log4perl, it should be syslog server agnostic, so you should be able to use these instructions with syslog-ng amongst other alternatives if needs be.

Should rsyslog not be installed inside your VM, install it as follows:

```
# apt-get install rsyslog rsyslog-relp
```

That done, we will need to configure rsyslog to accept remote connections. In this example, we will use the Reliable Event Logging Protocol (RELP) as we have found it to be just that, but you may of course use TCP or UDP as you see fit. To set up rsyslog to use RELP, create the file *30-remote.conf* in the folder */etc/rsyslog.d/*, and enter the following configuration:

```
$ModLoad imrelp
$InputRELPServerRun 2514

$template remotefile,"/var/log/%HOSTNAME%-%syslogfacility-text%.log"
*.* ?remotefile
```

This loads the RELP input module, and sets the server to listen on port 2514. Next, it tell rsyslog to generate the log filename dynamically, depending on the hostname of the client. The following line tells rsyslog to log all messages to this dynamically formed file. Now, restart rsyslog:

```
# service rsyslog restart
```

Next you will need to configure the QVD image to log remotely to this server. Inside the QVD image that you will be using, create in the folder */etc/rsyslog.d/* a file called *00-remote.conf* and enter the following configuration:

```
$ModLoad omrelp
*.* :omrelp:<hostname or IP address>:2514
```

Make sure to enter the IP address or hostname of the logging server. This configuration will tell rsyslog on the virtual machine to load the RELP output module, and to use port 2514 on your rsyslog server. Furthermore, it will log all output (*.*) to this remote host.

Note
Ensure that the RELP module is available on the server, and if not install it (the package is *rsyslog-relp* on Ubuntu and *rsyslog-module-relp* on SUSE).

Finally, edit the file */etc/qvd/vma.conf* on the virtual machine and enter the following to instruct QVD to log to syslog:

```
log4perl.appender.SYSLOG = Log::Dispatch::Syslog
log4perl.appender.SYSLOG.layout = Log::Log4perl::Layout::PatternLayout
log4perl.appender.SYSLOG.layout.ConversionPattern = %d %P %F %L %c - %m%n
log4perl.rootLogger = DEBUG, SYSLOG
log.level = DEBUG
```

Of course, having set syslog itself to log remotely, this log data will get passed by rsyslog to the remote server you have set up. To test this, simply use the logger command inside your image and the output should be in the logging server's logs.

Software

Installing additional software

Any additional software installed inside your KVM image or within your LXC container prior to zipping and deployment will be available to your desktop users. For most software this can be done using your package manager.

To install LXDE after the initial install, invoke the *apt-get* command:

```
# apt-get install lubuntu-desktop
```

Similarly, office packages and other tools you would like made available to all your users by default can be install through apt:

```
# apt-get install gimp libreoffice thunderbird thunderbird-lightning icedtea-7-plugin
```

Application Optimization

Certain applications optimized for a remote connection. For example, preventing the caret from blinking in Firefox will deliver a smoother experience:

```
# echo 'pref("ui.caretBlinkTime", 0);' >>  /etc/firefox/syspref.js
```

Remove Unneeded Services and Disable udev

It is advisable to remove any unneeded services that may impact on performance within your disk image. Under LXC, disabling udev has been shown to increase performance, particularly with regards to larger installs.

To remove and avahi:

```
# apt-get --purge remove avahi-daemon
```

If you do not wish to log your containers as per the instructions above, you might as well remove syslog:

```
# apt-get --purge rsyslogd
```

Remove uneeded programs to keep your disk image smaller:

```
# apt-get --purge remove xscreensaver
```

Clean both repository metadata and package caches:

```
# apt-get clean
```

Disable *udev* as follows:

```
# sh -c "echo 'manual' > /etc/init/udev.override"
```

Furthermore, prevent any network conflicts by emptying the */etc/udev/rules.d/70-persistent-net.rules* file in your container which can delay boot times and prevent the network from functioning:

```
$ echo "" > /etc/udev/rules.d/70-persistent-net.rules
```