

Managing Virtual Machines on Ubuntu KVM

This article is a dump of my experience with setting up a viable virtual machine management platform on an Ubuntu Hypervisor with following specs:

OS: Ubuntu 14.04.2 LTS

HDD:

Memory:

Preflight

Check for Virtualization Support

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

If 0 it means that your CPU doesn't support hardware virtualization.

If 1 or more it does – but you still need to make sure that virtualization is enabled in the BIOS.

Issue Package Updates

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

It is assumed you have the following packages installed

```
git
```

```
python-pip
```

If No:

```
sudo apt-get install git python-pip
```

Setup libvirt and KVM

Manually

```
sudo apt-get install qemu-kvm libvirt-bin virtinst bridge-  
utils sasl2-bin
```

Via bootstrap script

```
curl http://retspen.github.io/libvirt-bootstrap.sh | sudo sh
```

Once the installation is complete, add a designated user account

```
sudo adduser `id -un` libvirtd
```

Add the option `-l` in the file `/etc/default/libvirt-bin`:
It should look like:

```
libvirtd_opts="-d -l"
```

In the file `/etc/libvirt/libvirtd.conf` uncomment the line (Remove #):

```
listen_tls = 0  
listen_tcp = 1  
tcp_port = "16509"
```

Create a `saslpasword`:

```
sudo saslpasswd2 -a libvirt [username] // where [username]  
is the designated libvirt user account
```

```
Password: xxxxxx
```

```
Again (for verification): xxxxxx
```

Add firewall rule for *TCP port 16509*:

Create a file `/etc/ufw/applications.d/libvirtd` and it add the following lines:

```
[Libvirt]  
title=Virtualization library  
description=Open port for libvirt  
ports=16509/tcp
```

Add a firewall rule in the chain

```
sudo ufw allow from any to any app Libvirt
```

Install Administration Package

```
sudo apt-get install git python-pip python-libvirt python-libxml2 novnc supervisor nginx
```

Validate Installation

```
virsh -c qemu+tcp://127.0.0.1/system nodeinfo
```

Please enter your authentication name: [username]

Please enter your password: [password]

Sample Output:

```
CPU model:          x86_64
CPU(s):            2
CPU frequency:     3611 MHz
CPU socket(s):     1
Core(s) per socket: 2
Thread(s) per core: 1
NUMA cell(s):     1
Memory size:       3019260 kB
```

Install WebvirtMgr

Install required Packages

```
sudo apt-get install python-libvirt python-libxml2 supervisor nginx
```

Clone webvirtmgr project from github

```
cd /var/www
git clone git://github.com/retspen/webvirtmgr.git
```

Set permissions

```
sudo chown -R www-data:www-data /var/www/webvirtmgr
```

Install requirements

```
cd webvirtmgr
sudo pip install -r requirements.txt
```

Update Django Settings

```
./manage.py syncdb
./manage.py collectstatic
```

Enter the user information when prompted:

You just installed Django's auth system, which means you don't have any superusers defined.

Would you like to create one now? (yes/no): yes (Put: yes)

Username (Leave blank to use 'admin'): admin (Put: your username or login)

E-mail address: username@domain.local (Put: your email)

Password: xxxxxx (Put: your password)

Password (again): xxxxxx (Put: confirm password)

Superuser created successfully.

Adding additional superusers

./manage.py createsuperuser

(Optional) Enable remote access to the WebUI via Nginx or SSH Tunnel

Usually WebVirtMgr is only available from localhost on port 8000

You can connect via ssh tunnel, like so:

ssh user@server:port -L localhost:8000:localhost:8000 -L localhost:6080:localhost:6080

You should then be able to access WebVirtMgr by typing localhost:8000 in your browser after completing the install. Port 6080 is forwarded to make noVNC work.

or

You can configure a redirect to have the WebUI accessible via nginx:

if not already done:

sudo mv webvirtmgr /var/www/

sudo vim /etc/nginx/conf.d/webvirtmgr.conf

```
server {
    listen 80 default_server;
    server_name $hostname;
    #access_log /var/log/nginx/webvirtmgr_access_log;
    location /static/ {
        root /var/www/webvirtmgr/webvirtmgr; # or /srv
instead of /var
```

```

        expires max;
    }
    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Proto $remote_addr;
        proxy_connect_timeout 600;
        proxy_read_timeout 600;
        proxy_send_timeout 600;
        client_max_body_size 1024M; # Set higher depending
on your needs
    }
}

```

Comment the Server Section as it is shown in the example:
Either:

sudo vim /etc/nginx/sites-available/

or

sudo vim /etc/nginx/nginx.conf

Note:

The path may differ

The end result should look like this:

```

#    server {
#        listen      80 default_server;
#        server_name localhost;
#        root        /usr/share/nginx/html;
#
#        #charset koi8-r;
#
#        #access_log  /var/log/nginx/host.access.log
main;
#
#        # Load configuration files for the default
server block.
#        include /etc/nginx/default.d/*.conf;
#

```

```

#         location / {
#         }
#
#         # redirect server error pages to the static
page /40x.html
#         #
#         error_page 404                /404.html;
#         location = /40x.html {
#         }
#
#         # redirect server error pages to the static
page /50x.html
#         #
#         error_page 500 502 503 504  /50x.html;
#         location = /50x.html {
#         }
#     }

```

Restart nginx service:

sudo service nginx restart

Setup novnc

vi /etc/init.d/novnc

```

#!/bin/sh
### BEGIN INIT INFO
# Provides:          nova-novncproxy
# Required-Start:    $network $local_fs $remote_fs $syslog
# Required-Stop:     $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Nova NoVNC proxy
# Description:       Nova NoVNC proxy
### END INIT INFO
# PATH should only include /usr/* if it runs after the
mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="WebVirtMgr NoVNC proxy"
NAME='webvirtmgr-console'
DAEMON_PREFIX='/var/www/webvirtmgr/console'
PIDFILE="/run/${NAME}.pid"
#SCRIPTNAME="/etc/init.d/${NAME}"
SCRIPTNAME="/etc/init.d/novnc"

```

```

LOCK_DIR="/run/lock/${NAME}"
USER='www-data'
GROUP='www-data'
# read in defaults if available
[ -f "/etc/default/${NAME}" ] && . "/etc/default/${NAME}"
DAEMON="${DAEMON_PREFIX}/${NAME}"
# Exit if the package is not installed
[ -x $DAEMON ] || exit 0
mkdir -p ${LOCK_DIR}
chown "${USER}:${GROUP}" ${LOCK_DIR}
. /lib/lsb/init-functions
do_start()
{
    start-stop-daemon --start --background --quiet --chuid
"${USER}:${GROUP}" --make-pidfile --pidfile $PIDFILE --startas
$DAEMON --test > /dev/null \
    || return 1
    start-stop-daemon --start --background --quiet --chuid
"${USER}:${GROUP}" --make-pidfile --pidfile $PIDFILE --startas
$DAEMON -- \
    $DAEMON_ARGS \
    || return 2
}
do_stop()
{
    start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --
pidfile $PIDFILE
    RETVAL="$?"
    rm -f $PIDFILE
    return "$RETVAL"
}
case "$1" in
    start)
        log_daemon_msg "Starting $DESC " "$NAME"
        do_start
        case "$?" in
            0|1) log_end_msg 0 ;;
            2) log_end_msg 1 ;;
            esac
        ;;
    stop)

```

```

log_daemon_msg "Stopping $DESC" "$NAME"
do_stop
case "$?" in
  0|1) log_end_msg 0 ;;
  2) log_end_msg 1 ;;
esac
;;
status)
    status_of_proc "$DAEMON" "$NAME" && exit 0 || exit $?
    ;;
restart|force-reload)
log_daemon_msg "Restarting $DESC" "$NAME"
do_stop
case "$?" in
  0|1)
    do_start
    case "$?" in
      0) log_end_msg 0 ;;
      1) log_end_msg 1 ;; # Old process is still running
      *) log_end_msg 1 ;; # Failed to start
    esac
    ;;
    *)
    # Failed to stop
    log_end_msg 1
    ;;
esac
;;
*)
    echo "Usage: $SCRIPTNAME {start|stop|status|restart|force-
reload}" >&2
    exit 3
    ;;
esac

```

Setup Supervisor

sudo service novnc stop

sudo insserv -r novnc

sudo vi /etc/insserv/overrides/novnc

#!/bin/sh


```
### BEGIN INIT INFO
# Provides:          nova-novncproxy
# Required-Start:    $network $local_fs $remote_fs $syslog
# Required-Stop:     $remote_fs
# Default-Start:
# Default-Stop:
# Short-Description: Nova NoVNC proxy
# Description:       Nova NoVNC proxy
### END INIT INFO
```

sudo vi /etc/supervisor/conf.d/webvirtmgr.conf

```
[program:webvirtmgr]
    command=/usr/bin/python /var/www/webvirtmgr/manage.py
run_unicorn -c /var/www/webvirtmgr/conf/gunicorn.conf.py
    directory=/var/www/webvirtmgr
    autostart=true
    autorestart=true
    stdout_logfile=/var/log/supervisor/webvirtmgr.log
    redirect_stderr=true
    user=www-data
[program:webvirtmgr-console]
    command=/usr/bin/python
/var/www/webvirtmgr/console/webvirtmgr-console
    directory=/var/www/webvirtmgr
    autostart=true
    autorestart=true
    stdout_logfile=/var/log/supervisor/webvirtmgr-console.log
    redirect_stderr=true
    user=www-data
```

Restart supervisor daemon

sudo service supervisor restart

WebVirtMgr Post-Installation

I provide the below for additional considerations:

Networking

Before libvirt was installed, virbr0 did not exist. We

only had interfaces for loopback and eth0. virbr0 means “virtual bridge 0” and was automatically created by libvirt during installation. virbr0 was configured as a NAT-only interface. This means virtual machine hosts that use this bridge can get out to the network via the eth0 interface but any devices on the other side cannot initiate requests into virbr0 clients.

Here’s my networking configuration:

```
/etc/network/interfaces
auto eth0
iface eth0 inet dhcp
auto br0
iface br0 inet dhcp
    bridge_ports eth0
    bridge_stp off
auto eth1
iface eth1 inet dhcp
auto eth2
iface eth2 inet dhcp
```

Troubleshooting The WebVirtMgr Installation

Debugging the webapp

```
cd /var/www/webvirtmgr
```

Enable debug mode in the local_settings.py

```
sudo ./manage.py runserver 0:8000
```

Errors:

```
“webvirtmgr” “authentication failed:”
```

Ensure that any users specified in the connections matches what is listed in the local database

Review administrative users

```
sasldblistusers2 -f /etc/libvirt/passwd.db
```

```
saslpasswd2 -cf /etc/libvirt/passwd.db
```

If no users configured:

add user

```
sudo saslpasswd2 -a libvirt <username>
```

Installing Vagrant

Preflight

Install prerequisites

```
sudo apt-get install
```

```
sudo apt-get install gcc libvirt-dev ruby-libvirt
```

Download vagrant package

```
wget
```

```
https://dl.bintray.com/mitchellh/vagrant/vagrant_1.7.2_x86_64.  
deb
```

Install vagrant

```
sudo dpkg -i vagrant_1.7.2_x86_64.deb
```

Install vagrant libvirt/kvm provider

```
sudo vagrant plugin install vagrant-libvirt
```

Recovering a Failed QNAP Raid Volume

How to recover data from QNAP drives using testdisk from SystemRescueCd

Pre-flight

Given the following scenario:

QNAP server was factory reset, clearing the software RAID

information on the QNAP OS.

As such, all drives in the RAID were essentially orphaned. Data on the drives remained intact.

Recovery Options:

In order to recover the information, we could proceed via many troubleshooting pathways, two of which I list below:

- Rebuilding the software RAID
- Recovering the data directly from the drives

I chose the second option, since I wasn't too handy with administration of the Linux Multiple Device Driver (**MD**), aka software RAID.

In this article, we will be recovering the data from **ONE** drive at a time, so it is best to plug in **ONLY ONE** of drives to be recovered, along with a **spare** drive on which the recovered data will be copied to.

Recovery Software:

We will be using [SystemRescueCD](#) to perform the data recovery

I assume the following:

You've already booted the SystemRescueCD

You either have console or ssh access (or whatever other means) to the SystemRescueCD shell

You have the drive to be recovered and a spare plugged in to your system

Lastly, this is key in Understanding QNAP volumes:

QNAP utilizes Logical Volume Management (**LVM**) and the Linux **MD** software RAID technologies to manage its storage devices.

Partition 3 Holds all the **data** on any given drive

Keep this in mind as you start digging for your data on the QNAP drives.

Identify the Destination Drive

Before going through the recovery, you must prep the directory on which you will be copying the recovered data to.

With the specs on your hard drive already in mind, issue the list hardware command (**lshw**) to determine the device name to the drive:

```
lshw -short -c disk
```

Once you match the device information to that of the spare drive, you can proceed to initialize (wipe/clean) the drive or mount it if it's already prepared.

If the drive is already initialized, skip the next step, otherwise proceed ...

Prepare the Destination Drive

You can initialize the drive for use on the SystemRescueCD as follows:

```
fdisk <device_name>, e.g. fdisk /dev/sda
```

Follow the prompts to create a **Linux Partition**

Note: Once the partition is created, the device you'll actually be acting against is <device_namelogical_partition_number>, e.g. **/dev/sda1**

Once you've written the changes to the disk, you can proceed to create the filesystem on the drive:

```
mkfs -t <fs_type> <device_namelogical_partition_number>, e.g.
```

```
mkfs -t ext4 /dev/sda1
```

or

```
mkfs.<fstype> <device_namelogical_partition_number>, e.g.
```

```
mkfs.ext4 ext4 /dev/sda1
```

Once the filesystem has been created, you can mount it.

Do so first by creating a directory on which the drive will be mounted, e.g.:

```
mkdir /mnt/recovery
```

Mount the Destination Drive

Mounting the drive is quite easy, simply invoke the `mount` command, e.g.:

```
mount -t ext4 /dev/sda1 /mnt/recovery
```

Your destination drive is now ready to be used!

Identify the Data Partition on the Source Drive

```
root@sysresccd /root # cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md321 : active raid1 sdb5[0]
      7168000 blocks super 1.0 [2/1] [U_]
      bitmap: 1/1 pages [4KB], 65536KB chunk

md13 : active raid1 sdb4[25]
      458880 blocks super 1.0 [24/1] [U_]
      bitmap: 1/1 pages [4KB], 65536KB chunk

md2 : active raid1 sdb3[0]
      3897063616 blocks super 1.0 [1/1] [U]

md256 : active raid1 sdb2[1]
      530112 blocks super 1.0 [2/1] [U_]
      bitmap: 0/1 pages [0KB], 65536KB chunk

md9 : active raid1 sdb1[25]
      530048 blocks super 1.0 [24/1] [U_]
      bitmap: 1/1 pages [4KB], 65536KB chunk
```

The following commands are to be issued from the SystemRescueCD session:

First, we need to determine what MD volumes the SystemRescueCD has detected.

You can do so by displaying the contents of the `mdstat` file under `/proc` as follows:

```
cat /proc/mdstat
```

Sample Output:

```
Personalities : [linear] [multipath] [raid0] [raid1]
[raid6] [raid5] [raid4] [raid10]
md321 : active raid1 sdb5[0]
      7168000 blocks super 1.0 [2/1] [U_]
      bitmap: 1/1 pages [4KB], 65536KB chunk
```

```
md13 : active raid1 sdb4[25]
        458880 blocks super 1.0 [24/1]
[_U_____]
        bitmap: 1/1 pages [4KB], 65536KB chunk

md2 : active raid1 sdb3[0]
        3897063616 blocks super 1.0 [1/1] [U]

md256 : active raid1 sdb2[1]
        530112 blocks super 1.0 [2/1] [_U]
        bitmap: 0/1 pages [0KB], 65536KB chunk

md9 : active raid1 sdb1[25]
        530048 blocks super 1.0 [24/1]
[_U_____]
        bitmap: 1/1 pages [4KB], 65536KB chunk
```

As you can see from the above output, there is a disk with a 3rd partition that is most likely an MD LVM volume.

I'd say there is a 90% chance that this is the drive and partition we're interested in.

Take note of the device information, in this case **/dev/sdb3**

Invoke Testdisk Partiton Scan

```
root@sysresccd /root & testdisk /dev/sdb3

TestDisk 7.0, Data Recovery Utility, April 2015
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org

TestDisk is free software, and
comes with ABSOLUTELY NO WARRANTY.

Select a media (use Arrow keys, then press Enter):
>Disk /dev/sdb3 - 3990 GB / 3716 GiB - WDC WD40EZRX-00SPEB0

>[Proceed ] [ Quit ]

Note: Disk capacity must be correctly detected for a successful recovery.
If a disk listed above has incorrect size, check HD jumper settings, BIOS
detection, and install the latest OS patches and disk drivers.

Disk /dev/sdb3 - 3990 GB / 3716 GiB - WDC WD40EZRX-00SPEB0

Please select the partition table type, press Enter when done.
[ Intel ] Intel/PC partition
>[EFI GPT] EFI GPT partition map (Mac i386, some x86_64...)
[ Humax ] Humax partition table
[ Mac ] Apple partition map
[ None ] Non partitioned media
[ Sun ] Sun Solaris partition
[ Xbox ] Xbox partition
[ Return ] Return to disk selection

Hint: None partition table type has been detected.
Note: Do NOT select 'None' for media with only a single partition. It's very
rare for a disk to be 'Non-partitioned'.

>[ Analyze ] Analyse current partition structure and search for lost partitions
[ Advanced ] Filesystem Utils
[ Geometry ] Change disk geometry
[ Options ] Modify options
[ Quit ] Return to disk selection

Note: Correct disk geometry is required for a successful recovery. 'Analyze'
process may give some warnings if it thinks the logical geometry is mismatched.

P=Primary D=Deleted
>[Quick Search] Try to locate partition

Disk /dev/sdb3 - 3990 GB / 3716 GiB - CHS 485161 255 63
Analyse cylinder 238/485160: 00%
```

So, again, we've determined the data to be on device `/dev/sda3`

The next step is to run testdisk against this device:

testdisk /dev/sdb3

In the ensuing dialog, choose the following order of actions:

Select a media ...: (choose the device, in this case `/dev/sdb3`)

Proceed

Please select a partition table type ...: (choose **EFI GPT**)

Analyze Quick Search

At this point, the drive scan will commence.

Once it completes, you'll be presented with a partition table as detected by testdisk.

List Files for Recovery & Copy

```
Disk /dev/sdb3 - 3990 GB / 3716 GiB - CHS 485161 255 63
Partition      Start      End      Size in sectors
>P MS Data      41945088  7760570367  7718625280 [DataVol12]

Structure: Ok. Use Up/Down Arrow keys to select partition.
Use Left/Right Arrow keys to CHANGE partition characteristics:
                P=Primary D=Deleted
Keys A: add partition, L: load backup, T: change type, P: list files,
Enter: to continue
ext4 blocksize=4096 Large file Sparse SB Recover, 3951 GB / 3680 GiB

drwxrwxrwx  0  0  4096  9-Jan-2016 19:55 .
drwxrwxrwx  0  0  4096  9-Jan-2016 19:55 ..
drwx-----  0  0  16384 31-Aug-2015 22:58 lost+found
drwxrwxrwx  0  0  4096 27-Dec-2015 18:25
-rw-----  0  0  8192 27-Dec-2015 04:06
drwxrwxrwx  0  0  4096  5-Sep-2015 12:52
drwx-----  0  0  4096 30-Oct-2015 02:53
drwx-----  0  0  4096  9-Jan-2016 19:53
lrwxrwxrwx  0  0  9  9-Jan-2016 19:55
drwxr-xr-x  0  0  4096  9-Jan-2016 19:55

                                Next
Use Right to change directory, h to hide deleted files
q to quit, : to select the current file, a to select all files
C to copy the selected files. c to copy the current file
```

In the resulting partition table option, select the partition you think contains the data

Press **shift + P**

This will print the files on the partition

Read the instructions at the bottom of the file listing ...

q to quit

: to select the current file

a to select all files

shift + C to copy the selected files

c to copy the current file

Once you invoke the copy action, you will be prompted to navigate to the destination path.

Hopefully you've already completed that in steps '**Prepare the Destination Drive**' and '**Mount the Destination Drive**'

Once the copy process is started, you'll be presented with a progress indication.

Sit tight. The wait is worth it.

Sources

[SMB] HOW-TO RECOVER data from LVM volume on a PC (UX-500P)
<http://forum.qnap.com/viewtopic.php?t=93862>