ISSN online: 2282-5665

Consiglio Nazionale delle Ricerche

CERIS ISTITUTO DI RICERCA SULL'IMPRESA E LO SVILUPPO

Settembre

2013

Rapporto tecnico N.44

KVM: AN OPEN-SOURCE FRAMEWORK FOR VIRTUALIZATION

Giancarlo Birello, Ivano Fucile,
Valter Giovanetti, Anna Perin

Consiglio Nazionale delle Ricerche

Istituto di Ricerche sull'Impresa e Lo Sviluppo

Rapporti tecnici CERIS
ISSN: 2282-5665 online
Anno 8, N. 44; settembre 2013

Direttore Responsabile
Secondo Rolfo

Direzione e Redazione
CNR-Ceris
Istituto di Ricerca sull'Impresa e lo Sviluppo
Via Real Collegio, 30
10024 Moncalieri (Torino), Italy
Tel. +39 011 6824.911
Fax +39 011 6824.966
segreteria@ceris.cnr.it
http://www.ceris.cnr.it

Sede di Roma
Via dei Taurini, 19
00185 Roma, Italy
Tel. 06 49937810
Fax 06 49937884

Sede di Milano
Via Bassini, 15
20121 Milano, Italy
tel. 02 23699501
Fax 02 23699530

Segreteria di redazione
Enrico Viarisio
e.viarisio@ceris.cnr.it

KVM: AN OPEN-SOURCE FRAMEWORK FOR VIRTUALIZATION

Giancarlo Birello, Ivano Fucile Valter Giovanetti*
(CNR-Ceris, IT Office)

Anna Perin
(CNR-Ceris, Library)

CNR-Ceris
IT Office
Strada delle Cacce, 73
10135 Torino – Italy
Phone: +39 011 3977533/4/5

CNR-Ceris
Library
Via Real Collegio, 30
10024 Moncalieri (Torino) – Italy
Phone: +39 011 6824928

* Corresponding author: g.birello@ceris.cnr.it

ABSTRACT: This report analyses the configuration steps of the open-source hypervisor component KVM, (Kernel-based Virtual Machine). KVM solution is used for digiBESS (open-source project of digital archive) and all network services as web and application servers, user backup, long term archive and e-mail.

KEY WORDS: open-source, hypervisor, virtualization, KVM

Table of Contents

1 Introduction

Two years ago we started a project of digital preservation (digiBESS, www.digibess.it) committed by Bess (Social Science Electronic Library of Piemonte). The project grown up completely open-source, CNR-Ceris has deployed the software and server platforms of the repository in a virtualized and redundant infrastructure, the only software that wasn't open-source at the beginning of the project was the hypervisor component until the adoption of KVM (Kernel-based Virtual Machine).

KVM is an open source full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. The KVM host can run multiple virtual machines and it supports most production operating systems.

To manage network infrastructure underlining KVM framework we used Open vSwitch. Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols.

Soon we adopted KVM solution for all servers at CNR Ceris IT Office which provides network services to CNR Piedmont users. Currently we have about 30 virtualized machines providing network services as web and application servers, user backup, long term archive, e-mail. To manage virtual machine backups and storage we use our open source active/passive two-nodes cluster solution connected by iSCSI protocol to KVM hosts.

2 Hypervisor deployment

The KVM testbed is made by:

- server equipped with 1 AMD Opteron + 4GB RAM + 80GB SATA + 2 Ethernet (Gbit)
- network switch with VLANs capabilities
- two-nodes active/passive cluster for storage (iSCSI interface)

2.1 OS and KVM

We will use 1 Eth as management interface connected at reserved VLAN (eth1) and 1 Eth as Virtual machine network interface (eth0).

Begin installation of "Ubuntu server 12.04 x64" OS from CD ROM onto the server:

- at network interface setting configure eth1 with static IP
- at tasksel step select LVM, OpenSSH server and Virtualization host.

After installation complete restart the server and login using SSH into the server network management interface eth1, then verify KVM installation and CPU compatibility:

```
# kvm-ok

INFO: /dev/kvm exists
KVM acceleration can be used
```

Upgrade packages to last version and add myuser to libvirtd group:

```
# apt-get update
# apt-get upgrade
# adduser myuser libvirtd
# reboot
```

Verify KVM is up:

```
$ virsh -c qemu:///system list

 Id Name                 State
----------------------------------
```

2.2 OpenvSwitch

We chose OpenvSwitch as network infrastructure manager for virtual machine hosted on KVM.

We installed it from Ubuntu packages following this procedure:

```
# aptitude purge ebtables
# virsh net-destroy default
# virsh net-autostart --disable default
# service libvirt-bin stop
# service qemu-kvm stop
# aptitude install openvswitch-switch openvswitch-controller openvswitch-brcompat

# nano -w /etc/default/openvswitch-switch

     BRCOMPAT=yes

# service openvswitch-switch restart
# service openvswitch-controller restart
# reboot
```

Verify modules and service:

```
$ lsmod | grep br

     brcompat_mod           13512  0
     openvswitch_mod        83993  2 brcompat_mod

$ service openvswitch-switch status

     ovsdb-server is running with pid 1349
     ovs-vswitchd is running with pid 1358
     ovs-brcompatd is running with pid 1393
```

Add bridge “br0” (libvirt compatibility) and connect to eth0 (untagged + VLANs tagged):

```
# ovs-vsctl add-br br0
# ovs-vsctl add-port br0 eth0
```

Add virtual bridges br2 (VLAN tag 2) and br3 (VLAN tag 3):

```
# ovs-vsctl add-br br2 br0 2
# ovs-vsctl add-br br3 br0 3
```

Add dummy entries in /etc/network/interfaces:

```
# nano -w /etc/network/interfaces

auto eth0
iface eth0 inet manual
up ifconfig $IFACE 0.0.0.0 up
down ifconfig $IFACE down

auto br0
iface br0 inet manual
up ifconfig $IFACE 0.0.0.0 up
down ifconfig $IFACE down

auto br2
iface br2 inet manual
up ifconfig $IFACE 0.0.0.0 up
down ifconfig $IFACE down

auto br3
iface br3 inet manual
up ifconfig $IFACE 0.0.0.0 up
down ifconfig $IFACE down
```

Edit libvirt upstart script to wait for bridge then reboot:

```
# nano -w /etc/init/libvirt-bin.conf

     - start on runlevel [2345]
     + start on (runlevel [2345] and net-device-up IFACE=br0)

# reboot
```

Verify running configuration:

```
# ovs-vsctl show

    Bridge "br0"
        Port "eth0"
            Interface "eth0"
        Port "br3"
            tag: 3
            Interface "br3"
                type: internal
        Port "br2"
            tag: 2
            Interface "br2"
                type: internal
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "1.4.0+build0"
```

2.3 LVM

OS was installed using LVM partitioning as show here:

```
# fdisk -l /dev/sda
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048      499711      248832   83  Linux
/dev/sda2          501758   156301311    77899777    5  Extended
/dev/sda5          501760   156301311    77899776   8e  Linux LVM

# pvdisplay

  --- Physical volume ---
  PV Name               /dev/sda5
  VG Name               kvm2
  PV Size               74.29 GiB / not usable 2.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              19018
  Free PE               14635
  Allocated PE          4383
  PV UUID               xxxxx

# lvdisplay

  --- Logical volume ---
  LV Name                /dev/kvm2/root
  VG Name                kvm2
  ...
  --- Logical volume ---
  LV Name                /dev/kvm2/swap_1
  VG Name                kvm2
  ...
```

We also use LVM for VM virtual disk. This architecture allows snapshots, flexible partition resize and mixed local and remote storage.

After installed a first VM `tubu1`, LVM configuration will be:

```
# pvdisplay -m

  --- Physical volume ---
  PV Name               /dev/sda5
  VG Name               kvm2
  PV Size               74.29 GiB / not usable 2.00 MiB
  ...
  --- Physical Segments ---
  Physical extent 0 to 1359:
    Logical volume   /dev/kvm2/root
    Logical extents  0 to 1359
  Physical extent 1360 to 2382:
    Logical volume   /dev/kvm2/swap_1
    Logical extents  0 to 1022
  Physical extent 2383 to 4382:
    Logical volume   /dev/kvm2/tubu1
    Logical extents  0 to 1999
  Physical extent 4383 to 19017:
    FREE
```

2.4 iSCSI

Cluster remote storage is connected to KVM host by iSCSI protocol.

Install and configure iSCSI Initiator onto the KVM hypervisor:

```
# apt-get install open-iscsi
# nano -w /etc/iscsi/iscsid.conf

     node.startup = automatic

# /etc/init.d/open-iscsi restart
```

Target discovery:

```
# iscsiadm -m discovery -t st -p iSCSIServerIP
```

Remove unwanted nodes:

```
# rm -R /etc/iscsi/nodes/iqnxxxx
```

Configure node authentication:

```
# nano -w /etc/iscsi/nodes/iqnyyyy/1234/default

     node.session.auth.authmethod = CHAP
     node.session.auth.username = user
     node.session.auth.password = password

# /etc/init.d/open-iscsi restart
```

We use iSCSI partition as backing storage for LVM.
Attach iSCSI target as /dev/sdc on KVM host then partitioning it as follow:

```
# fdisk -l /dev/sdc
  ...
   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1            2048     4194303     2096128   8e  Linux LVM
```

Add Physical Volume `/dev/sdc1` to Volume Group `iscsi1vg` and create Logical Volume `tubu1b`:

```
# pvdisplay -m

  --- Physical volume ---
  PV Name               /dev/sdc1
  VG Name               iscsi1vg
  PV Size               2.00 GiB / not usable 3.00 MiB
  Allocatable           yes (but full)
  PE Size               4.00 MiB
  Total PE              511
  Free PE               0
  Allocated PE          511
  ...

  --- Physical Segments ---
  Physical extent 0 to 510:
    Logical volume   /dev/iscsi1vg/tubu1b
  ...
```

Assign virtual disk `tubu1b` to VM `tubu1` and from `tubu1` SSH session format and mount new disk:

```
# fdisk -l /dev/vdb

Disk /dev/vdb: 2143 MB, 2143289344 bytes
...
   Device Boot      Start         End      Blocks   Id  System
/dev/vdb1            2048     4186111     2092032   83  Linux

# mkfs.ext4 /dev/vdb1
# mount /dev/vdb1 /media/test
```

We checked iSCSI disk reliability vs cluster fail-over.
Trig a cluster fail-over during first copy command, VM freeze for 2 seconds:

```
# pv ubuntu-12.04-server-amd64.iso > /media/test/ubuntu1.iso
```

Trig a second cluster fail-over during second copy command, VM freeze for 2 seconds:

```
# pv ubuntu-12.04-server-amd64.iso > /media/test/ubuntu2.iso
```

Then verify data integrity

```
# diff /media/test/ubuntu1.iso /media/test/ubuntu2.iso
```

3 Maintenance

There are a lot of tools to perform maintenance operations. Main tools we use are: SSH connection to KVM host and virsh CLI client, VM console access provided by virsh console command from hypervisor and virt-manager GUI from remote desktop client.

3.1 VM serial console

Into VM define a serial console:

```
# nano -w /etc/init/ttyS0.conf

# 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
```

On KVM host, verify in VM XML file is present serial tag:

```
<domain>
  ...
  <devices>
    ...
    <serial type='pty'>
      <source path='/dev/pts/7'/>
      <target port='0'/>
      <alias name='serial0'/>
    </serial>
    ...
  </devices>
  ...
</domain>
```

SSH into KVM host then access VM `tubu1` console:

```
$ virsh console tubu1
```

3.2 Virtual Machine Manager

Virtual Machine Manager is a GUI to manage KVM hosts and VMs.

Install Virt-Manager from packages:

```
# apt-get install virt-manager
```

Launch Virt-Manager and connect to remote KVM host using Hypervisor=QEMU/KVM, Method=SSH and Hostname=KVMHOSTIP.

From within GUI you can manage hypervisor storage, not network. Also you can manage existing VM, create new VM or remove existing one and access graphic console of VM.

NOTE: due a bug (http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=663931) you need ***`/etc/init.d/udev restart`*** *on KVM host if virt-manager client timeout.*

3.3 Clone

Libvirt library provides useful code included one to clone existing VM. The command line tool clone is included in package virtinst based on libvirt library.

```
# apt-get install virtinst
```

To clone existing VM `tubu1`, first you have to create a virtual disk for VM clone same size as original:

```
# lvdisplay kvm2
...
  --- Logical volume ---
  LV Name                /dev/kvm2/tubu1
  VG Name                kvm2
  LV UUID                ******
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                7.81 GiB
  Current LE             2000
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:2

# lvcreate -l 2000 -n tubu1-clone iscsi2
```

Then you can invoke the command:

```
# virt-clone --original tubu1 --name tubu1-clone --file /dev/mapper/iscsi2-tubu1-
  clone --prompt
```

The new guest `tubu1-clone` will be listed in KVM hypervisor and ready to run.

3.4 Backup procedure

If VM virtual disk is KVM based then you can take a live backup using LVM snapshot feature.

The `virt-backup.pl` script from Daniel Berteaud is really a good solution to fully backup a libvirt managed virtual machine.

Download, edit settings for Ubuntu distribution and install required packages:

```
# wget http://repo.firewall-services.com/misc/virt/virt-backup.pl
# chmod +x virt-backup.pl

# nano -w virt-backup.pl

     # lvcreate path
     - $opts{lvcreate} = '/usr/sbin/lvcreate -c 512';
     + $opts{lvcreate} = '/sbin/lvcreate -c 512';
     # lvremove path
     - $opts{lvremove} = '/usr/sbin/lvremove';
     + $opts{lvremove} = '/sbin/lvremove';


# apt-get install libxml-simple-perl libsys-virt-perl libfile-which-perl
```

Define a partition for backup:

```
# lvcreate -l 3000 -n backup iscsi2
# mkfs.ext4 /dev/iscsi2/backup
# mount /dev/mapper/iscsi2-backup /var/lib/libvirt/backup
```

To mount it at boot:

```
# blkid
    ...
  /dev/mapper/iscsi2-backup: UUID="0a413f6c-50cd-45c7-ba1c-0a413f6c1ac6" TYPE="ext4"

# nano -w /etc/fstab

  +  UUID=0a413f6c-50cd-45c7-ba1c-0a413f6c1ac6    /var/lib/libvirt/backup        ext4
  defaults,auto,_netdev   0       0
```

To take a live backup of VM tubu1-clone using LVM snapshot:

```
# ./virt-backup.pl --dump --vm=tubu1-clone --state --debug

Connecting to libvirt daemon using qemu:///system as URI

Checking tubu1-clone status

Running dump routine for tubu1-clone

Locking tubu1-clone

Saving XML description for tubu1-clone to /var/lib/libvirt/backup/tubu1-clone/tubu1-clone.xml
tubu1-clone is running, saving state....
tubu1-clone state saved as /var/lib/libvirt/backup/tubu1-clone/tubu1-clone.state

Analysing disk /dev/iscsi2/tubu1-clone connected on tubu1-clone as vda

Running:  /sbin/lvcreate  -c  512  -p  r  -s  -n  /dev/iscsi2/tubu1-clone_1344612236  -L  5G
  /dev/iscsi2/tubu1-clone > /dev/null 2>&1
/dev/iscsi2/tubu1-clone seems to be a valid logical volume (LVM), a snapshot has been taken as
  /dev/iscsi2/tubu1-clone_1344612236
Adding /dev/iscsi2/tubu1-clone_1344612236 to the list of disks to be backed up


The following disks will be dumped:

Source: /dev/iscsi2/tubu1-clone_1344612236       Dest: /var/lib/libvirt/backup/tubu1-clone/tubu1-
  clone_vda.img

We can run a live backup

Trying to restore tubu1-clone from /var/lib/libvirt/backup/tubu1-clone/tubu1-clone.state
Waiting for restoration to complete

Starting dump of /dev/iscsi2/tubu1-clone_1344612236 to /var/lib/libvirt/backup/tubu1-clone/tubu1-
  clone_vda.img

32000+0 records in
32000+0 records out
8388608000 bytes (8.4 GB) copied, 244.731 s, 34.3 MB/s
Removing snapshot /dev/iscsi2/tubu1-clone_1344612236

Cannot start domain restoration, tubu1-clone is running (maybe already restored after a live
  backup ?)
Removing lock file for tubu1-clone
```

The backup creates a directory with these files:

```
# ls -l /var/lib/libvirt/backup/tubu1-clone

-rw------- 1 root root  284363322 Aug 10 17:23 tubu1-clone.state
-rw-r--r-- 1 root root 8388608000 Aug 10 17:28 tubu1-clone_vda.img
-rw-r--r-- 1 root root       2453 Aug 10 17:23 tubu1-clone.xml
```

Restore from backup to same VM:

```
# virsh shutdown tubu1-clone
# dd if=/var/lib/libvirt/backup/tubu1-clone/tubu1-clone_vda.img
  of=/dev/iscsi2/tubu1-clone
# virsh restore tubu1-clone.state
```

Restore from backup to different VM:

- Check original disk size

```
# qemu-img info /var/lib/libvirt/backup/tubu1-clone/tubu1-clone_vda.img

image: /var/lib/libvirt/backup/tubu1-clone/tubu1-clone_vda.img
file format: raw
virtual size: 7.8G (8388608000 bytes)
disk size: 7.8G
```

- Create new volume same size as original

```
lvcreate -L 8388608000B -n tubu1-restore kvm2
```

- Restore disk

```
# dd if=/var/lib/libvirt/backup/tubu1-clone/tubu1-clone_vda.img of=/dev/kvm2/tubu1-
  restore
```

- Edit metadata with new name and UUID

```
# cp /var/lib/libvirt/backup/tubu1-clone/tubu1-clone.xml ./tubu1r.xml

# uuidgen
6702d0ee-0775-4ff9-bb75-d4a62ebc82b7

# nano -w tubu1r.xml

# diff tubu1r.xml /var/lib/libvirt/backup/tubu1-clone/tubu1-clone.xml
1,3c1,3
< <domain type='kvm' id='107'>
<   <name>tubu1r</name>
<   <uuid>6702d0ee-0775-4ff9-bb75-d4a62ebc82b7</uuid>
---
> <domain type='kvm' id='7'>
>   <name>tubu1-clone</name>
>   <uuid>4d9fad50-7206-3117-18f3-dd620c402b9b</uuid>
31c31
<       <source dev='/dev/kvm2/tubu1-restore'/>
---
>       <source dev='/dev/iscsi2/tubu1-clone'/>
```

- Define new virtual machine from metadata

```
# virsh define tubu1r.xml
```

3.5 Migrate from esxi 4.1

For *nix OS guests is more simple than Windows. This example is for **Linux (Ubuntu) OS**.

- Remove vmware-tools from guest OS.

```
# vmware-unistall-tools.pl
```

- Create a VM flat disk backup on esxi hypervisor, e.g. by ghettoVCB script:

```
# ghettoVCB.sh -f vmltemp -c vmspec
```

where vmltemp

```
Aserver
```

and vmspec/Aserver

```
VM_BACKUP_VOLUME=/vmfs/volumes/Backup
DISK_BACKUP_FORMAT=zeroedthick
VM_BACKUP_ROTATION_COUNT=1
POWER_VM_DOWN_BEFORE_BACKUP=0
ENABLE_HARD_POWER_OFF=0
ITER_TO_WAIT_SHUTDOWN=4
POWER_DOWN_TIMEOUT=5
SNAPSHOT_TIMEOUT=15
ENABLE_COMPRESSION=0
ADAPTER_FORMAT=lsilogic
VM_SNAPSHOT_MEMORY=0
VM_SNAPSHOT_QUIESCE=1
VMDK_FILES_TO_BACKUP=all
EMAIL_LOG=0
```

- Prepare KVM storage for copy virtual flat disk:

```
# lvcreate -L 20G -n vmware kvm2
# mkdir /vmware
# mkfs.ext4 /dev/kvm2/vmware
# mount /dev/kvm2/vmware /vmware
```

- Copy virtual flat disk from esxi to KVM host:

```
# cd /vmware/
# mkdir Aserver
# cd Aserver/
#  scp  root@EsxiHypervisor.my.org:/vmfs/volumes/Backup/Aserver/Aserver-2012-09-02_12-
  00-01/Aserver_1-flat.vmdk ./
```

- Verify flat disk:

```
# file Aserver_1-flat.vmdk

/vmware/Aserver/Aserver_1-flat.vmdk:  x86  boot  sector;  GRand  Unified  Bootloader,
  stage1  version  0x3,  stage2  address  0x2000,  stage2  segment  0x200;  partition  1:
  ID=0x83,  active,  starthead  1,  startsector  63,  15952482  sectors;  partition  2:
  ID=0x5, starthead 0, startsector 15952545, 819315 sectors, code offset 0x48
```

- For virtual machine file based, create qcow2 file:

```
# qemu-img convert Aserver_1-flat.vmdk -O qcow2 Aserver_1-flat.qcow2
```

- For virtual machine LVM based, create LV:

```
# ls -l Aserver_1-flat.vmdk

-rw------- 1 root root 8589934592 Sep  3 13:49 /vmware/Aserver/Aserver_1-flat.vmdk

#lvcreate -L 8589934592B -n aserver iscsi2
#dd if=Aserver_1-flat.vmdk | pv -s 8G | dd of=/dev/iscsi2/aserver
```

- Define the new virtual machine using this virtual disk (qcow2 or LV).

Here an example of migration of **Windows Server 2003** from esxi to KVM.

- On esxi remove vmware-tools from guest VM and reboot.
- Enable boot from IDE executing the mergeide.reg at Microsoft KB, http://support.microsoft.com/kb/314082/en-us.
- Shutdown guest and create virtual machine flat disk backup on esxi hypervisor, e.g. by ghettoVCB script.
- Copy virtual flat disk from esxi to KVM host.
- Verify the flat disk and create virtual disk (LVM based).
- Define a new virtual machine using
  this virtual disk as IDE,
  a dummy disk as VirtIO,
  a CD ISO from KVM project wiki downloads
  (http://www.linux-kvm.org/page/WindowsGuestDrivers/Download_Drivers)
  and a network card as VirtIO.
- Start the new virtual machine and install requested drivers from CD then shutdown
- Change main virtual disk to VirtIO and remove dummy disk
- Start virtual machine
- TCP/IP parameters for best performance
  (http://www.linux-kvm.org/page/WindowsGuestDrivers/kvmnet/registry):

```
net.reg

Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\AFD\Parameters]
"DefaultSendWindow"=dword:00100000
"DefaultReceiveWindow"=dword:00100000
"FastSendDatagramThreshold"=dword:00004000


[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters]
"Tcp1323Opts"=dword:00000001
"TcpWindowSize"=dword:00100000
```

4 Upgrade LIBVIRT to 1.0.0 from source (experimental)

We used libvirt and openvswitch from packages, for this Ubuntu release (12.04 LTS) libvirt package is 0.9.8 version and openvswitch is 1.4.0.

We experimented on the test machine libvirt 1.0.0 from source because libvirt 1.x release is better integrated with OpenvSwitch.

Download souce code:

```
# wget http://libvirt.org/sources/libvirt-1.0.0.tar.gz
# tar -xvzf libvirt-1.0.0.tar.gz
```

Install extra packages:

```
# apt-get install libxml2-dev libgnutls-dev libyajl-dev libnl-dev pkg-config
  libdevmapper-dev python-dev
```

Compile:

```
# ./configure --prefix=/usr --localstatedir=/var --sysconfdir=/etc --with-esx=no
# make
# make install
```

Edit libvirtd config:

```
# diff /etc/libvirt/libvirtd.conf /etc/libvirt/libvirtd.conf.ORI

< unix_sock_group = "libvirtd"
---
> #unix_sock_group = "libvirt"

< unix_sock_rw_perms = "0770"
---
> #unix_sock_rw_perms = "0770"

< auth_unix_ro = "none"
---
> #auth_unix_ro = "none"

< auth_unix_rw = "none"
---
> #auth_unix_rw = "none"
```

Remove libvirt default network:

```
# virsh net-destroy default
# virsh net-autostart --disable default
```

Edit OpenvSwitch conf:

```
# nano -w /etc/default/openvswitch-switch

     BRCOMPAT=no
```

Edit OVS bridges and leave only main bridge:

```
# ovs-vsctl del-br br2
# ovs-vsctl del-br br3
```

Edit interfaces and remove dummy entries in /etc/network/interfaces then reboot:

```
# nano -w /etc/network/interfaces

auto eth0
iface eth0 inet manual
up ifconfig $IFACE 0.0.0.0 up
down ifconfig $IFACE down

auto br0
iface br0 inet manual
up ifconfig $IFACE 0.0.0.0 up
down ifconfig $IFACE down

# reboot
```

Configure KVM network:

```
# nano -w mynet-ovs.xml

<network>
  <name>ovs-network</name>
  <forward mode='bridge'/>
  <bridge name='br0'/>
  <virtualport type='openvswitch'/>
  <portgroup name='vmport0' default='yes'>
  </portgroup>
  <portgroup name='vmport2'>
    <vlan trunk='yes'>
      <tag id='0'/>
      <tag id='2'/>
      <tag id='3'/>
    </vlan>
  </portgroup>
</network>

# virsh net-define mynet-ovs.xml
# virsh net-start
```

VM network configuration example:

```
...
    <interface type='network'>
      <mac address='52:54:00:bd:6c:2a'/>
      <source network='ovs-network' portgroup='vmport0'/>
      <target dev='vnet0'/>
      <alias name='net0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
    </interface>
...
```

5 Webography

[1] *DigiBESS developer site*, http://dev.digibess.it, visited September 2013.

[2] *GhettoVCB.sh*, http://www.virtuallyghetto.com/2013/01/new-updates-for-ghettovcb-ghettovcb.html, visited September 2013.

[3] *Kernel Based Virtual Machine*, http://www.linux-kvm.org/page/Main_Page, visited September 2013.

[4] *Libvirt*, http://libvirt.org/, visited September 2013.

[5] *LVM2*, http://sourceware.org/lvm2/, visited September 2013.

[6] *Microsoft KB*, http://support.microsoft.com/kb/314082/en-us, visited September 2013.

[7] *Open vSwitch*, http://openvswitch.org/, visited September 2013.

[8] *Ubuntu*, http://www.ubuntu.com, visited September 2013.

[9] *Virt-backup*, https://github.com/vazhnov/virt-backup.pl, visited September 2013.