

CERNVM RELEASE TESTING WALKTHROUGH

---

# CernVM Release Testing Walkthrough Developer Manual

---



GNU USER

Technical Report  
Version: 1.0 June 2011

### **Abstract**

The CERNVM RELEASE TESTING project is a testing infrastructure for CernVM images, the usecase for the project is to provide an automated testing environment, which will install and configure CernVM images, run the set of tests and report the results on a web interface.

# Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>CernVM Release Testing Test Client Platform Setup</b>	<b>2</b>
2.1	Introduction . . . . .	2
2.2	Red Hat Based Test Client Setup . . . . .	3
2.2.1	Configuring the system . . . . .	3
2.2.2	Installing libvirt and virsh . . . . .	4
2.2.3	Installing and configuring KVM . . . . .	6
2.2.4	Installing and configuring VirtualBox . . . . .	7
2.2.5	Setting up the Tapper Test Suite . . . . .	8
2.3	Debian Based Test Client Setup . . . . .	12
2.3.1	Installing the system . . . . .	12
2.3.2	Configuring the system . . . . .	12
2.3.3	Installing libvirt and virsh . . . . .	15
2.3.4	Installing and configuring KVM . . . . .	16
2.4	OS X Test Client Setup . . . . .	17
2.4.1	Configuring the system . . . . .	17
2.4.2	Installing libvirt and virsh . . . . .	18
2.4.3	Installing and configuring VirtualBox . . . . .	19
2.4.4	Installing and configuring VMware . . . . .	20
<b>3</b>	<b>CernVM Release Testing Server Platform Setup</b>	<b>22</b>
3.1	Introduction . . . . .	22
3.2	Debian Based Server Setup . . . . .	23
3.2.1	Installing the Tapper Server . . . . .	23
3.2.2	Setting up Tapper Web Interface and Database . . . . .	24
	<b>Bibliography</b>	<b>27</b>
	<b>Index</b>	<b>28</b>

# 1 Overview

CernVM currently supports images for VirtualBox, VMware, Xen, KVM and Microsoft Hyper-V hypervisors, each new release of a CernVM image needs to be thoroughly tested on each supported platform and hypervisor. The CERNVM RELEASE TESTING project is designed to meet this requirement by providing an automated testing environment for CernVM images, which will install and configure CernVM images, run the set of tests and report the results on a web interface.

The intent of this document is to provide a step-by-step guide on setting up an entire CERNVM RELEASE TESTING infrastructure, including instructions on how to set up and configure test clients, the main server running the web interface and database, as well as writing and executing tests. If you are new to release testing and want a document to guide you through the entire process of setting up a working CERNVM RELEASE TESTING infrastructure, then this guide is for you.

All the code needed to setup the entire RELEASE TESTING infrastructure for CernVM image testing, is located at the CERNVM RELEASE TESTING Google Code project page[1] including this document and all other documentation.

While this document is not intended to be a replacement for the reference manual, the following is a brief description of the RELEASE TESTING infrastructure including an introduction to the core component, AMD TAPPER [2]. Figure 1.1 consists of a diagram outlining the TAPPER Architecture, which consists of test clients and a server, the server is what controls the test clients, gathers results, and then displays the results through a web interface.

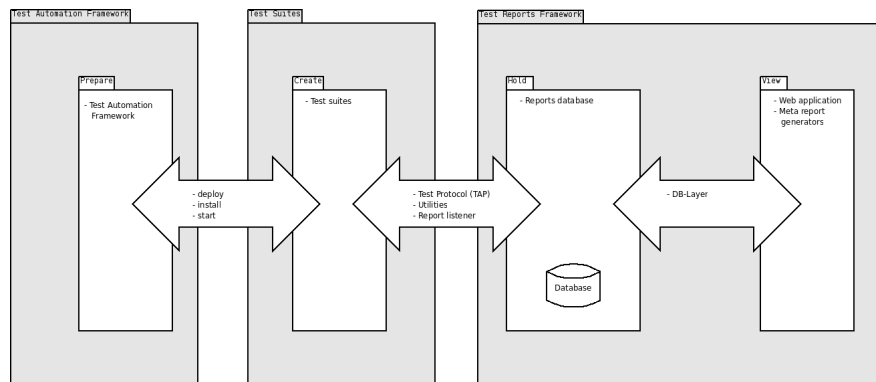


Figure 1.1: Overview of the TAPPER architecture

## 2 CernVM Release Testing Test Client Platform Setup

### 2.1 Introduction

The intent of this document is to provide a step-by-step guide on setting up an entire CERNVM RELEASE TESTING infrastructure, including instructions on how to set up and configure test clients, the main server running the web interface and database, as well as writing and executing tests. If you are new to release testing and want a document to guide you through the entire process of setting up a working CERNVM RELEASE TESTING infrastructure, then this guide is for you.

This section provides complete step by step instructions on how to setup and configure the test clients which are part of a basic working RELEASE TESTING environment by outlining the procedure for setting up test clients on numerous platforms, hence why this is called a *walkthrough* document. This guide is intended for users familiar enough with computers and desktop environments to enter basic commands in a terminal and install various operating systems.

As this guide is directed towards users who are new to CERNVM RELEASE TESTING and TAPPER and are interested in quickly getting a CERNVM testing infrastructure quickly set up, many assumptions regarding the requirements necessary are made. As is the case, these instructions are provided for a generalized audience based on our own experience and the requirements that we feel most users will have, *so feel free to deviate from the instructions.*

## 2.2 Red Hat Based Test Client Setup

### 2.2.1 Configuring the system

1. After the system has booted remove the follow unnecessary startup applications by selecting from the menu **System -> Preferences -> Startup Applications**
  - bluetooth
  - evolution alarm
  - Gnome Login Sound
  - PackageKit Update Applet
  - print queue
  - screensaver
  - visual assistance/aid
  - volume control
  - any others you think are unnecessary based on your own discretion
2. Next enable and configure the remote desktop from the menu **System -> Preferences -> Remote Desktop** and ensure that the following options are configured
  - Enable the option “Allow others to view your desktop”
  - Enable the option “Allow other users to control your desktop”
  - Disable the option “You must confirm access to this machine”
3. Next enable SSH access to the machine, in order for SSH and VNC access to work the firewall will have to be disabled
  - a. First disable the firewall from the menu **System -> Administration -> Firewall** and click the “Disable” button and then click “Apply” to apply the changes!  
**This is a quick solution for now because it’s too much work to configure the firewall for VNC, SSH, Apache, MySQL, PHPMyAdmin, MCP, and all the other network daemons and should not be a problem if this is just being accessed internally.**
  - b. Now that the firewall is disabled, configure sshd, the ssh daemon, to run on startup  

```
$ su -c "chkconfig --level 345 sshd on"
```
4. Next, configure the system to login automatically at boot
  - a. Edit the login screen configuration file for gdm using the following command  

```
$ su -c "gedit /etc/gdm/custom.conf"
```
  - b. Then in the custom.conf file, put the following under the heading [daemon], which will automatically log the system in as the user you created, *make sure you replace the user cernvm with the user that you created.*

## 2 CERNVM RELEASE TESTING *Test Client Platform Setup*

Listing 2.1: Configure Automatic Login

```
AutomaticLoginEnable=true
AutomaticLogin=cernvm
TimedLoginEnable=true
TimedLogin=cernvm
TimedLoginDelay=0
```

5. Next, configure the screen saver from the menu **System -> Preferences -> Screensaver** and ensure that the following options are configured
  - Disable the option “Lock screen when screensaver active”
6. Now, reboot the machine, and ensure that the following work
  - It automatically boots up into the full desktop environment without having to login
  - You have access to the machine using SSH and can login on the root account
  - You have VNC access to the machine and can control the system using VNC
7. Finally, update the system from the menu **System -> Administration -> Software Update** and after it has completed the updates reboot the system

### 2.2.2 Installing libvirt and virsh

1. The virtualization API libvirt and the command line tool virsh [3] are the essential components required for setting up a test client and must be installed and properly configured before any testing can begin. Ensure that you follow the proceeding directions carefully and validate that virsh is working properly before proceeding to install and configure the various hypervisors.
2. First, begin by reviewing the release news listed on the libvirt website, <http://libvirt.org/news.htm> and read through the release notes for the latest version released to make sure that there are no regressions or deprecated support for the platforms you wish to support. If you intend to set up an entire infrastructure and support all of the CERNVM virtualization platforms, which would include *Xen*, *KVM*, *VirtualBox*, and *VMware*, then you must download a version later than *0.8.7* as there was no support for *VMware* prior to that release.
3. Next, download the latest release that is a **src.rpm** file from the libvirt release server, <http://libvirt.org/sources/> based on the latest release which does not have any regressions or deprecations for the virtualization platforms you wish to support <sup>1</sup>. As of this date, the latest release of libvirt is version 0.9.2, this is the release that will be used for the following instructions and examples.

---

<sup>1</sup>This shouldn't be an issue but just in case there is a newer version in which Xen support is deprecated, then you would need to use the last release which has Xen support

## 2 CERNVM RELEASE TESTING *Test Client Platform Setup*

4. Next, install the following dependencies which are required to generate the libvirt rpm files from the src.rpm file that was downloaded, *from now on execute all commands as root.*

### Listing 2.2: Install src.rpm Dependencies

```
# Change to root account, enter password if prompted
$ su

# Install dependencies for using a src.rpm
$ yum install rpm rpm-devel rpm-libs rpmdevtools rpm-python \
rpm-build rpmrebuild
```

5. Next, install the following dependencies which are required to install libvirt.

### Listing 2.3: Install libvirt Dependencies

```
# Change to root account, enter password if prompted
$ su

# Install dependencies for libvirt
$ yum install xhtml1-dtds readline-devel ncurses-devel gettext augeas \
libpciaccess-devel yajl-devel libpcap-devel avahi-devel radvd \
cyrus-sasl-devel parted-devel libcap-ng-devel libssh2-devel \
audit-libs-devel systemtap-sdt-devel gnutls-utils gnutls-devel \
python-devel xen-devel libudev-devel libnl-devel device-mapper-devel \
numactl-devel netcf-devel libcurl-devel libcgrouper
```

6. Next, create the libvirt RPM installation files using the following command, replace the src.rpm file shown in the example with the file you downloaded previously.

### Listing 2.4: Create libvirt RPM Files

```
# Change to root account, enter password if prompted
$ su

$ rpmbuild --rebuild libvirt-0.9.2-1.fc14.src.rpm
```

7. Finally, install the libvirt RPM files by navigating to the `/root/rpmbuild/RPMS` folder and then changing to the directory for your computer architecture *such as x86\_64*. Then install the files in the same order as shown in the example, replacing the version of src.rpm file shown in the example with the version of the files in your directory, most importantly install the files in the following order: *libvirt-client*, *libvirt-devel*, *libvirt-python*, *libvirt*. Finally, if a package does not install or complains about conflicts use the `-force` argument to force the installation.



Listing 2.5: Install libvirt

```
# Change to root account, enter password if prompted
$ su

# Change to location of RPM files
$ cd /root/rpmbuild/RPMS/x86_64/

# Install the files in the following order, if install
# fails or has conflicts use rpm -iv --force
$ rpm -iv libvirt-client-0.9.2-1.fc13.x86_64.rpm
$ rpm -iv libvirt-devel-0.9.2-1.fc13.x86_64.rpm
$ rpm -iv libvirt-python-0.9.2-1.fc13.x86_64.rpm
$ rpm -iv libvirt-0.9.2-1.fc13.x86_64.rpm
```

8. Finally, start the service libvirtd, and ensure that virsh installed correctly and is running by connecting to the test hypervisor and ensuring that the test virtual machine, named “test” is running.

Listing 2.6: Verify virsh was Installed Properly

```
# Change to root account, enter password if prompted
$ su

# Verify virsh is working, test should be running
$ service libvirtd start
$ virsh -c test:///default list --all
```

### 2.2.3 Installing and configuring KVM

1. The first step is to install the KVM hypervisor, start by installing KVM and the other additional packages such as virt-manager, which is a graphical management tool and virt-install, which is a command line interface (CLI) virtual machine creation/installation/configuration tool using the following commands with root privileges. If you receive a message that a package is already installed then simply continue.

Listing 2.7: Installing KVM and Other Related Programs

```
$ yum install virt-manager qemu-kvm python-virtinst virt-viewer
```

2. Next, verify that KVM has been installed properly and that virsh can connect to the KVM hypervisor using the following commands, if you are able to connect to the virsh console without any errors then virsh is able to connect to the KVM hypervisor.

Listing 2.8: Verify that virsh can Access KVM

```
$ su
$ virsh -c qemu:///session
```

3. Finally, to ensure that KVM is properly configured and installed, follow this guide provided on the CernVM website <http://cernvm.cern.ch/portal/kvm> **except, do not create a kvm definition file as the xml template file is provided by the test suite scripts** and verify that you are able to connect to the libvirtd kvm system session.

Listing 2.9: Verify that KVM is Properly Configured

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

## 2.2.4 Installing and configuring VirtualBox

1. First, begin by downloading the latest version of VirtualBox from the VirtualBox download page, <http://www.virtualbox.org/wiki/Downloads> ensure that you select the appropriate Red Hat based distribution, version and architecture for your system. The following instructions for this section of the guide uses VirtualBox 4.0.10 for Fedora 13, AMD64.
2. Next, after downloading the latest version of VirtualBox for your distribution install VirtualBox as the root account using the following command.

```
# Enter the root password when prompted
$ su
$ rpm -iv VirtualBox-*.rpm
```

3. Next, in order to use VirtualBox and have full access to the drivers needed for USB support, ensure that the root account belongs to the group “vboxusers”. Begin by navigating to **System -> Users and Groups** and then from the “User Manager” window click the “Groups” tab, under the column “Group Name” for the group “vboxusers” ensure that root is one of the group members. If the root account is not a group member of “vboxusers” highlight the “vboxusers” entry and click the “Properties” button, then enable the root account from the list of users and click “OK” to apply the changes.
4. Due to an issue with VirtualBox<sup>2</sup>, in order for it to work with virsh the virtual machine(s) must be created and configured as the root account, otherwise when you try to connect or start a VirtualBox virtual machine with virsh you will get an “unknown error”, which is obviously very vague and difficult to resolve.

---

<sup>2</sup>The issues is that VirtualBox looks for virtual machine configuration files (\*.vbox) in the “VirtualBox VMs” folder of the user that launched VirtualBox. The issue is worsened by the fact that there can only be one “VirtualBox VMs” folder which causes conflicts with multiple users.

Therefore **ALWAYS** start VirtualBox as the root account using the following procedure.

Listing 2.10: Always Start VirtualBox as Root

```
# Switch to the root account, enter root password
$ su

# Start VirtualBox as root
$ virtualbox
```

5. Finally, verify that VirtualBox has been installed properly and that virsh can connect to the VirtualBox hypervisor, verify that the VirtualBox module, *vboxdrv* has been loaded and that you are able to connect to the virsh console without any errors.

Listing 2.11: Verify that virsh can Access VirtualBox

```
$ su

# Verify that the vboxdrv module is loaded
$ lsmod | grep -i vboxdrv

# Verify that virsh can connect to virtualbox
$ virsh -c vbox:///session
```

6. If everything so far has worked, then libvirt, virsh, and the hypervisors have been installed and configured properly, if you have any outstanding issues solve them before proceeding further, or go to the section “Server Platform Setup” 3 as the TAPPER server does not require libvirt, virsh, or hypervisor configuration.

## 2.2.5 Setting up the Tapper Test Suite

1. Before proceeding any further ensure that you have all other test clients set up this far, and then proceed to follow the instructions for setting up and configuring the Tapper server in the section “Server Platform Setup” 3.
2. Now that the TAPPER server has been installed and configured and the TAPPER web interface and database have proven to be working, the next step is to verify that the test client can actually send a report to the TAPPER server in the form of a TAP file. After sending the TAP report to the server, ensure that the test client is working by viewing the tapper reports in your browser at the following url: [http://<tapper\\_server>/tapper/reports](http://<tapper_server>/tapper/reports). You should now see a report from the test client, there should be a report from a system named

## 2 CERNVM RELEASE TESTING *Test Client Platform Setup*

whatever the “Tapper-Machine-Name” in `demo_report.tap` was set as. *For the example `demo_report.tap` provided below it would be `cernvm-rhtestclient`.*<sup>3</sup>.

Listing 2.12: Send a Basic Report to the TAPPER Server

*# Save the following in a file named `demo_report.tap`*

```
1..2
# Tapper-Suite-Name: Tapper-Deployment
# Tapper-Suite-Version: 1.001
# Tapper-Machine-Name: cernvm-rhtestclient
ok - Hello World
ok - Just another description

# Send the report to the tapper server using netcat
$ cat demo_report.tap | nc -w10 cernvm-server 7357
```

3. Next, download a copy of `tapper-autoreport` and the CernVM Test Cases from the Google Code svn repository [1] and install the the following dependencies required by `tapper-autoreport`.

Listing 2.13: Install TAPPER AutoReport and Dependencies

```
# Install subversion, required to checkout auto-tapper
$ yum install subversion

# Checkout a copy of auto-tapper and cernvm testcases
$ svn checkout http://cernvm-release-testing.googlecode.com/svn/trunk/tapper/tapper-autoreport/ cernvm-release-testing-read-only

# Install the missing dependencies
$ yum install perl-Module-CoreList
$ yum install perl-CPAN

# Install the required perl modules
$ cpan
$ install prove
$ install XML::XPath

# Install test suite dependencies
$ yum install uuid
$ yum install spawn
$ yum install expect
$ yum install expectk
```

---

<sup>3</sup>This is why a consistent hostname convention was emphasized earlier, as reports are often sorted and organized based on hostnames

## 2 CERNVM RELEASE TESTING *Test Client Platform Setup*

```
$ yum install nmh

# Run install-mh and accept the defaults
$ install-mh
```

4. Now that that tapper-autoreport has been installed, configure the variables in the configuration file for the hypervisors you want to test and according to your TAPPER infrastructure setup. All of the settings provided in the configuration files by default are the **mandatory, minimum configuration options** and in most cases the defaults should be sufficient for testing, the only mandatory settings that are not provided by default **and must be configured manually are CVM\_TS\_REPORT\_SERVER and CVM\_VM\_IMAGE\_VERSION**. Please refer to the developer manual for a more complete detailed list of the mandatory and optional configuration settings.

**CVM\_TS\_SUITENAME** Must ALWAYS be set, the default suite name in the configuration file should be suitable

**CVM\_TS\_SUITEVERSION** Must ALWAYS be set, reflects the release version number of the test suite framework, the default version given in the configuration should only be changed if you customize/update the scripts

**CVM\_TS\_REPORT\_SERVER** Must ALWAYS be set, this is the ip address or hostname of the Tapper report server which the reports from the test results are sent to

**CVM\_TS\_DOWNLOAD\_PAGE** Must ALWAYS be set, normally the default url provided in the configuration file is accurate, but in the event that the internal CERNVM image release page is relocated then this url must be changed.

**CVM\_VM\_HYPERVISOR** Must ALWAYS be set, should not have to change the default hypervisor in the configuration files, valid values (case sensitive) are **kvm,vbox,vmware**

**CVM\_VM\_TEMPLATE** Must ALWAYS be set, normally the default template provided in the configuration file should not be changed, only change this to use a custom template file. The custom template file *must be placed within the templates folder*

**CVM\_VM\_NET\_TEMPLATE** Must ALWAYS be set, normally the default network template provided in the configuration file should not be changed, only change this to use a custom network template file for the CERNVM image, **only applies to kvm and virtualbox**. The custom network template file *must be placed within the templates folder*

**CVM\_VM\_IMAGE\_VERSION** Must ALWAYS be set, specifies the version of the CernVM image to use from the release page

**CVM\_VM\_IMAGE\_TYPE** Must ALWAYS be set, specifies the type of CernVM image, valid image types supported, (case sensitive) are **basic and desktop**

**CVM\_VM\_ARCH** Must ALWAYS be set, specifies the architecture of the CERNVM image, valid architectures (case sensitive) are **x86** and **x86\_64**

5. Finally, now that taper-autoreport has been installed and configured on the test client and the test client and TAPPER Server have proven to be working, the next step is to verify that taper-autoreport works correctly and can actually send a report to the TAPPER server in the form of a TAP file. After the CERNVM Test Cases script, “cernvm-tests.sh” has completed and sent a TAP report to the server, ensure that the test client is working by viewing the taper reports in your browser at the following url: [http://<tapper\\_server>/tapper/reports](http://<tapper_server>/tapper/reports). You should now see a new report from the test client, there should be a report from a system with the same hostname<sup>4</sup>.

Listing 2.14: Run TAPPER -AutoReport for CernVM Test Cases

```
# Simply source the configuration file , and execute the script
$ . ./config/<configuration_file>
$ ./cernvm-tests.sh
```

---

<sup>4</sup>This is why a consistent hostname convention was emphasized earlier, as reports are often sorted and organized based on hostnames

## 2.3 Debian Based Test Client Setup

### 2.3.1 Installing the system

1. Install the system as you would for any other Linux distribution, except pay attention to the following instructions on configuring Debian to use ext4 (if available) instead of ext3 as the performance gains are noticeable.
2. Now, when prompted by the installer to configure the partitioning layout, if there are other operating systems installed on the system select the “Guided - use entire disk” option, and if available select the option “Use Remaining Free Space”. Otherwise, if there are no other operating system installed on the hard drive select the “Manual” option, *beware that doing so will risk erasing everything on the hard drive if you create a new partition table*. Using the manual option, create two primary partitions, with the first taking up the size of the hard drive minus twice the size of the amount of RAM installed, and the second primary partition as a SWAP file using the remaining free space. The following is an example of what the partiton layout would look like for a 40.0 GB hard drive with 2GB of ram.

Listing 2.15: Manual Partition Layout Example

#1	PRIMARY	36.0	GB	B	f	EXT4	/
#2	PRIMARY	4.0	GB		f	SWAP	SWAP

3. Finally, the last important installation setting, when prompted to choose software to install, select the following
  - Graphical desktop environment
  - SSH Server
  - Standard system utilities

### 2.3.2 Configuring the system

1. After the system has booted remove the follow unnecessary startup applications by selecting from the menu  
System -> Preferences -> Startup Applications
  - bluetooth
  - evolution alarm
  - Gnome Login Sound
  - print queue
  - screensaver
  - update notifier
  - visual assistance/aid

## 2 CERNVM RELEASE TESTING *Test Client Platform Setup*

- volume control
  - any others you think are unnecessary based on your own discretion
2. Remove the follow unnecessary services by selecting from the menu  
**System -> Administration -> Services**
    - alsa utils
    - bluetooth
    - CUPS
    - exim4
    - any others you think are unnecessary based on your own discretion
  3. Next enable and configure remote desktop from the menu  
**System -> Preferences -> Remote Desktop** and ensure that the following options are configured
    - Enable the option “Allow others to view your desktop”
    - Enable the option “Allow other users to control your desktop”
    - Disable the option “You must confirm access to this machine”
  4. Next configure the system to login automatically at boot from the menu select  
**System -> Administration -> Login Screen** and then set it to login to the user account you created previously (such as cernvm) automatically.
  5. Next, remove cd-rom support from sources.list, which is used by Debian for updates <sup>5</sup>, execute the following command with root privileges and comment out any lines that start with “deb cdrom” by using a #

Listing 2.16: Removing CD-ROM Requirement for Updates

```
$ su -c "gedit /etc/apt/sources.list"
```

6. Again, continue to edit /etc/apt/sources.list still with root privileges and ensure that each line ends with “main contrib non-free”, then save the file and do the following command with root privileges.

Listing 2.17: Updating the System

```
$ su -c "apt-get update"
```

7. Next, configure the screen saver from the menu  
**System -> Preferences -> Screensaver** and ensure that the following options are configured
  - Disable the option “Lock screen when screensaver active”

---

<sup>5</sup>And is a nuisance for any new user as it forces you to find the CD and put it in the computer for the update to continue



8. The following instructions involve enabling headless support so that you can remote desktop to the machine without having a monitor connected to the computer

- a. Edit the `xorg.conf` file and put the following in it

Listing 2.18: Configuring Xorg for Headless Support

```
$ su -c "gedit /etc/X11/xorg.conf"
```

```
Section "Device"
Identifier "VNC_Device"
Driver "vesa"
EndSection
```

```
Section "Screen"
Identifier "VNC_Screen"
Device "VNC_Device"
Monitor "VNC_Monitor"
SubSection "Display"
Modes "1280x1024"
EndSubSection
EndSection
```

```
Section "Monitor"
Identifier "VNC_Monitor"
HorizSync 30-70
VertRefresh 50-75
EndSection
```

- b. Then edit `grub` and set the option “`nomodeset`”, and proceed to update `grub` and reboot

Listing 2.19: Configuring Grub for Headless Support

```
$ su -c "gedit /etc/default/grub"
```

```
GRUB_CMDLINE_LINUX="nomodeset"
```

```
$ su -c "update-grub"
```

9. Now, reboot the machine, and ensure that the following work
  - It automatically boots up into the full desktop environment without having to login
  - You have access to the machine using SSH and can login on the root account

- You have VNC access to the machine and can control the system using VNC
10. Finally, update the system from the menu  
 System -> Administration -> Update Manager and after it has completed the updates reboot the system

### 2.3.3 Installing libvirt and virsh

1. The virtualization API libvirt and the command line tool virsh [3] are the essential components required for setting up a test client and must be installed and properly configured before any testing can begin. Ensure that you follow the proceeding directions carefully and validate that virsh is working properly before proceeding to install and configure the various hypervisors.
2. First, begin by reviewing the release news listed on the libvirt website, <http://libvirt.org/news.htm> and read through the release notes for the latest version released to make sure that there are no regressions or deprecated support for the platforms you wish to support. If you intend to set up an entire infrastructure and support all of the CERNVM virtualization platforms, which would include *Xen*, *KVM*, *VirtualBox*, and *VMware*, then you must download a version later than 0.8.7 as there was no support for VMware prior to that release.
3. Next, download the latest release that is a **src.rpm** file from the libvirt release server, <http://libvirt.org/sources/> based on the latest release which does not have any regressions or deprecations for the virtualization platforms you wish to support <sup>6</sup>. As of this date, the latest release of libvirt is version 0.9.2, this is the release that will be used for the following instructions and examples.
4. Next, install the following dependencies which are required to generate the libvirt rpm files from the src.rpm file that was downloaded, *from now on execute all commands as root*.
5. Finally, start the service libvirtd, and ensure that virsh installed correctly and is running by connecting to the test hypervisor and ensuring that the test virtual machine, named “test” is running.

Listing 2.20: Verify virsh was Installed Properly

```
# Change to root account, enter password if prompted
$ su

# Verify virsh is working, test should be running
$ service libvirtd start
$ virsh -c test:///default list --all
```

---

<sup>6</sup>This shouldn't be an issue but just in case there is a newer version in which Xen support is deprecated, then you would need to use the last release which has Xen support

### 2.3.4 Installing and configuring KVM

1. The first step is to install the KVM hypervisor, start by installing KVM and the other additional packages such as virt-manager, which is a graphical management tool and virt-install, which is a command line interface (CLI) virtual machine creation/installation/configuration tool using the following commands with root privileges. If you receive a message that a package is already installed then simply continue.

Listing 2.21: Installing KVM and Other Related Programs

```
$ su -c "apt-get install qemu-kvm virt-manager"
```

2. Next, verify that KVM has been installed properly and that virsh can connect to the KVM hypervisor using the following commands, if you are able to connect to the virsh console without any errors then virsh is able to connect to the KVM hypervisor.

Listing 2.22: Verify that virsh can Access KVM

```
$ su
$ virsh -c qemu:///session
```

3. Finally, to ensure that KVM is properly configured and installed, follow this guide provided on the CernVM website <http://cernvm.cern.ch/portal/kvm> **except, do not create a kvm definition file as the xml template file is provided by the test suite scripts** and verify that you are able to connect to the libvirtd kvm system session.

Listing 2.23: Verify that KVM is Properly Configured

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

## 2.4 OS X Test Client Setup

### 2.4.1 Configuring the system

1. After the system has booted, the first thing to configure are the power management settings and other preferences, as this system will be running as a test client, sleep and other automatic energy saving features must be disabled. Begin by navigating to the power options,  
**Apple logo -> System Preferences -> Energy Saver** for the option “Computer sleep” slide the bar to the far right so that it is set to “Never” and ensure that the following options are all disabled.
  - Put the hard disk(s) to sleep when possible
  - Wake for Ethernet network access
  - Allow power button to put the computer to sleep
2. Next, set a hostname for the system from the menu  
**Apple logo -> System Preferences -> Sharing** beside the “Computer Name:” option at the top, click the “Edit...” button. Then enter a relevant hostname for the system based on the hardware or operating system it is running; the hostname should be relevant and unique to better identify the system. A good naming convention should refer to the hardware or operating system and call it a host to differentiate from the virtual machine that will be running as a guest, for example a hostname such as *cernvm-osx-host* could be used, **whatever convention you use make sure it is consistent**.
3. Next, enable SSH access to the system by navigating to  
**Apple logo -> System Preferences -> Sharing** and from the list of services that can be shared, enable “Remote Login”, which is SSH.
4. Now, to enable VNC access to the system, select from the same list of services that can be shared, “Remote Management” and for local access options window that appears, enable all of the options listed such as “Observe” and “Change settings”. Then to enable VNC compatibility so that the OS X system can be accessed by other non-Apple computers, click the “Computer Settings...” button and enable the following options and set a password for the “VNC viewers...” option.
  - Show Remote Management status in menu bar
  - Anyone may request permission to control screen
  - VNC viewers may control screen with password
5. Now, to ensure that your user logs in automatically, navigate to  
**Apple logo -> System Preferences -> Accounts** and click “Login Options”, you may have to click on the lock icon and enter your password in order to make changes to the login options. Then for the option “Automatic login:” select your user from the list of accounts to enable automatic login.

6. Finally, to ensure that the settings were configured properly, reboot the machine and ensure that the following work.
  - It automatically boots up into the full desktop environment without having to login
  - You have access to the machine using SSH and can login
  - You have VNC access to the machine and can control the system using VNC

### 2.4.2 Installing libvirt and virsh

1. The virtualization API libvirt and the command line tool virsh [3] are the essential components required for setting up a test client and must be installed and properly configured before any testing can begin. Ensure that you follow the proceeding directions carefully and validate that virsh is working properly before proceeding to install and configure the various hypervisors.
2. First, to build libvirt from source Xcode must be installed, <http://developer.apple.com/xcode/>, Xcode 4 requires either a paid developer membership, or must be purchased from the App Store; but, Xcode 3 is freely available, **this guide uses Xcode 3 to build libvirt from source.**
3. Next, review the release news listed on the libvirt website, <http://libvirt.org/news.htm> and read through the release notes for the latest version released to make sure that there are no regressions or deprecated support for the platforms you wish to support. If you intend to set up a test client which supports VMware then you must download a version later than 0.8.7 as there was no support for VMware prior to that release.
4. Next, download the latest source release that is a **tar.gz** file from the libvirt release server, <http://libvirt.org/sources/> based on the latest release which does not have any regressions or deprecations for the virtualization platforms you wish to support. As of this date, the latest release of libvirt is version 0.9.2, this is the release that will be used for the following instructions and examples.
5. Next, download and install the following dependencies which are required by libvirt, some of the newer releases are broken on OS X so the follow versions must be used. Before issuing make, configure each with `-prefix=/usr/local` and for libgcrypt also add `-disable-asm`.
  - libpgperror-1.7 - `ftp://ftp.gnupg.org/gcrypt/libpgp-error/libpgp-error-1.7.tar.gz`
  - libgcrypt-1.4.5 - `ftp://ftp.gnupg.org/gcrypt/libgcrypt/libgcrypt-1.4.5.tar.gz`
  - gnutls-2.8.5 - `ftp://ftp.gnu.org/pub/gnu/gnutls/gnutls-2.8.5.tar.bz2`

- Next, configure and install libvirt using the following commands, which builds libvirt with support for VMware and VirtualBox.

Listing 2.24: Configure and Install libvirt

```
# Configure libvirt with VMware/VirtualBox support
./configure --without-xen --without-sasl --without-avahi \
--without-polkit --without-qemu --without-lxc --without-openvz \
--without-remote --with-libvirt --without-uml --with-vmware --with-vbox

# Build/install libvirt
sudo make
sudo make install
```

- Finally, ensure that virsh installed correctly and is running by connecting to the test hypervisor and ensuring that the test virtual machine, named “test” is running.

Listing 2.25: Verify virsh was Installed Properly

```
# Verify virsh is working, test should be running
$ virsh -c test:///default list --all
```

### 2.4.3 Installing and configuring VirtualBox

- First, begin by downloading and installing the latest version of VirtualBox for OS X from the VirtualBox download page, <http://www.virtualbox.org/wiki/Downloads> ensure that you select the appropriate architecture for your system. The following instructions for this section of the guide uses VirtualBox 4.0.10 for AMD64.
- Next, to verify that VirtualBox has been installed properly and that virsh can connect to the VirtualBox hypervisor, verify that the following VirtualBox kernel extensions are loaded and that you are able to connect to the virsh console without any errors.
  - org.virtualbox.kext.VBoxDrv
  - org.virtualbox.kext.VBoxUSB
  - org.virtualbox.kext.VBoxNetFlt
  - org.virtualbox.kext.VBoxNetAdp

Listing 2.26: Verify that virsh can Access VirtualBox

```
# Verify that the kernel extensions are loaded
$ kextstat | grep -i virtualbox

# Verify that virsh can connect to virtualbox
$ virsh -c vbox:///session
```

- Now, proceed to download and extract the desired VirtualBox virtual machine image onto the system from the CERNVM download portal, <http://cernvm.cern.ch/portal/downloads> it is recommended that you download the VirtualBox Desktop image for your architecture. For this guide the Desktop image will be downloaded as it is the most practical image for the majority of users.
- Now that the virtual machine has been created and configured verify that it is able to boot completely without crashing, *you will be presented with a login screen when it has booted completely*. Then shutdown the virtual machine by clicking “Actions” from within the virtual machine and selecting “Shutdown”, after the virtual machine has shutdown close VirtualBox and then connect to the VirtualBox hypervisor and determine that you can view, start, and stop virtual machine.

Listing 2.27: Verify VirtualBox Works with Virsh

```
$ virsh --connect vbox:///session

# Verify the virtualbox virtual machine is accessible
# Name of the virtual machine created should be listed
$ list --all

# Verify the virtual machine can be turned on/off
$ start <name of virtual machine>
# Wait about 2 minutes for the system to boot
$ shutdown <name of virtual machine>
```

- Finally, configure the the virtual machine network to automatically start when the system boots.

Listing 2.28: Configure Network

```
# Configure virsh network for VirtualBox
$ virsh --connect vbox:///session
$ net-start vboxnet0
$ net-autostart vboxnet0
```

#### 2.4.4 Installing and configuring VMware

- First, begin by downloading and installing the latest version of VMware Fusion for OS X from the VMware product page, <http://www.vmware.com/products/>, VMware Fusion requires a license, so you will have to purchase it in order to continue.
- Next, to verify that VMware Fusion has been installed properly, verify that the following VMware kernel extensions are loaded, currently virsh has support to

## 2 CERNVM RELEASE TESTING *Test Client Platform Setup*

connect to the VMware hypervisor, but does not support interacting with VMware through VMware Fusion.

- com.vmware.kext.vmx86
- com.vmware.kext.vhci
- com.vmware.kext.vmioplug
- com.vmware.kext.vmnet

Listing 2.29: Verify VMware Kernel Extensions Loaded

```
# Verify that the kernel extensions are loaded
$ kextstat | grep -i vmware
```

3. Now, proceed to download and extract the desired VMware virtual machine image onto the system from the CERNVM download portal, [urlhttp://cernvm.cern.ch/portal/downloads](http://cernvm.cern.ch/portal/downloads) it is recommended that you download the VMware Desktop image for your architecture. For this guide the Desktop image will be downloaded as it is the most practical image for the majority of users.



## 3 CernVM Release Testing Server Platform Setup

### 3.1 Introduction

This section provides complete step by step instructions on how to setup and configure the TAPPER server which is part of a basic working RELEASE TESTING environment by outlining the procedure for setting up the server, hence why this is called a *walkthrough* document. This guide is intended for developers, for a more complete reference guide to installing and configuring the TAPPER server please refer to the regular walkthrough document.

## 3.2 Debian Based Server Setup

For installing and configuring a Debian based server, follow the instructions outlined in the sections “Configuring the system” 2.3.2 for installation and configuration instructions with the only exception being that the hostname should be something unique such as *cernvm-debian6-server*, to indicate that it is running the TAPPER server, **again keep the hostname convention consistent.**

### 3.2.1 Installing the Tapper Server

1. Next, execute the following commands to install necessary dependencies, *from now on all commands require root privileges.*

Listing 3.1: Install Dependencies

```
$ apt-get update
$ apt-get install make
$ apt-get install subversion
```

2. Now, download the latest copy of the Tapper-Deployment, which is an installer for Tapper from the CERNVM RELEASE TESTING Google Code Project page

Listing 3.2: Download Tapper-Deployment

```
$ svn checkout http://cernvm-release-testing.googlecode.com/svn/trunk/\
installer/tapper-deployment
```

3. Now edit the Makefile in the Tapper-Deployment installer folder and configure variable TAPPER\_SERVER which is the hostname of the machine that is currently installing the starter-kit. For now disregard the TESTMACHINE variables, you should have something similar to this in the Makefile.

Listing 3.3: Makefile Configuration

```
# initial machine names
TAPPER_SERVER=cernvm-server
TESTMACHINE1=johnconnor
TESTMACHINE2=sarahconnor
TESTMACHINE3=bullock
```

4. After you have configured the Makefile in the installer folder, install Tapper-Deployment by executing the following command, for any prompts during the installation leave them as default and press enter. **During the installation, you will be prompted for the mysql password, DO NOT ENTER a password here UNLESS you already have an existing MySQL installation/database with a password set for the “root” account.** Finally, if you have any errors or other issues during the installation, please contact us with a summary of the problem and send us a copy of the installation log “install.log”.

Listing 3.4: Install Tapper-Deployment

```
$ cd installer /
$ make localsetup 2>&1 | tee install.log
```

## 3.2.2 Setting up Tapper Web Interface and Database

1. Next you need to set a password for the root account of the mysql database<sup>1</sup>

Listing 3.5: Set MySQL Root Password

```
# Example: mysqladmin -u root password abc123
$ mysqladmin -u root password <newpassword>
```

2. Now that the installion has completed and the security issue has been dealt with, ensure that you can access the tapper web interface and that it is working by viewing it in your browser using the url, <http://localhost/tapper><sup>2</sup>
3. Next, install PHPMyAdmin so that it's easy to administrate and configure the Tapper databases, when prompted to select the “Web server to reconfigure automatically” select apache2 by pressing the space bar and press enter. *If you are prompted to “configure databases for phpmyadmin with dbconfig-common” select NO .*

Listing 3.6: Install PHPMyAdmin

```
$ apt-get update

# When prompted for the server to reconfigure automatically select apache2
# when prompted to configure the database with dbconfig-common select NO
$ apt-get install phpmyadmin
```

4. Verify that PHPMyAdmin has been installed and configured correctly and that you can access the tapper web interface by viewing it in your browser using the url, <http://localhost/phpmyadmin>. Login to the PHPMyAdmin web interface using the *username root and the MySQL root password you set earlier using mysqladmin.*
5. Now, add all of the configured test machines created in the “Test Client Platform Setup” section to the TAPPER database and set the test clients as active, then add the hardware specifications for each test client to the database. This example is just using a single generic test machine, you will have to repeat these commands for each test client and change the hostname *cernvm-host* and the values for *mem*, *core*, *vendor*, and *has\_ecc* as needed; the vendor can be AMD or Intel.

<sup>1</sup>This will eventually be implemented in the makefile

<sup>2</sup>This can be accessed locally and remotely from other systems using the server hostname or IP address

### 3 CERNVM RELEASE TESTING Server Platform Setup

Listing 3.7: Adding Test Clients to Tapper Database

```
# Add the hostname of the test client to database
$ tapper-testrun newhost --name cernvm-host --active

# Add the hardware specifications for the test client
$ mysql testrundb -utapper -ptapper
$ insert into host_feature(host_id, entry, value) values \
((select id from host where name = 'cernvm-host'), 'mem',
4096);
$ insert into host_feature(host_id, entry, value) values \
((select id from host where name = 'cernvm-host'), 'cores',
4);
$ insert into host_feature(host_id, entry, value) values \
((select id from host where name = 'cernvm-host'), 'vendor', 'AMD');
$ insert into host_feature(host_id, entry, value) values \
((select id from host where name = 'cernvm-host'), 'has-ecc',
0);
```

6. Next, send a sample test report to the tapper server, to ensure that the web interface, MCP, database, and reports framework are all working by viewing the tapper reports in your browser at the following url, <http://localhost/tapper/reports> You should now see a report from whatever the “Tapper-Machine-Name” in demo\_report.tap was set as. *For the example demo\_report.tap provided below it would be cernvm-server.*

Listing 3.8: Send a Report from the TAPPER Server to Itself

```
# Save the following in a file named demo_report.tap
$ vi demo_report.tap

1..2
# Tapper-Suite-Name: Tapper-Deployment
# Tapper-Suite-Version: 1.001
# Tapper-Machine-Name: cernvm-server
ok - Hello test world
ok - Just another description

# Send the report to the tapper server using netcat
$ cat demo_report.tap | netcat -q7 -w1 cernvm-server 7357
```

7. Finally, ssh login to one of the test machine that was set up earlier, *in our examples, cernvm-host* and send another sample test report to the tapper server, to ensure that the web interface, MCP, database, and reports framework are all working by viewing the tapper reports in your browser at the following url: <http://localhost/tapper/reports>. You should now see a report from

### 3 CERNVM RELEASE TESTING *Server Platform Setup*

whatever the “Tapper-Machine-Name” in demo\_report.tap was set as. *For the example demo\_report.tap provided below it would be cernvm-testclient.*

Listing 3.9: Send a Report to the TAPPER Server from a Test Client

*# Save the following in a file named demo\_report.tap*  
\$ vi demo\_report.tap

```
1..2
# Tapper-Suite-Name: Tapper-Deployment
# Tapper-Suite-Version: 1.001
# Tapper-Machine-Name: cernvm-testclient
ok - Hello test world
ok - Just another description

# Send the report to the tapper server using netcat
$ cat demo_report.tap | netcat -q7 -w1 cernvm-server 7357
```

8. Now that it has been verified that the tapper server, including the web interface, MCP, database, and reports framework are all working; return to the sections titled “Setting up the Tapper Test Suite” for each of the test client, as there are unique instructions for each operating system.

# Bibliography

- [1] CernVM Release Testing Google Code Project.  
<https://code.google.com/p/cernvm-release-testing/>.
- [2] Advanced Micro Devices Inc. AMD Tapper. <http://developer.amd.com/zones/opensource/amdtapper/pages/default.aspx>, 2011.
- [3] Red Hat Inc. libvirt Virtualization Api. <http://libvirt.org/>, 2011.

# Index

TAPPER Architecture, [1](#)