

openQRM Enterprise Administrator Guide

Enterprise System Administration and IaaS Datacenter Automation with openQRM 5.2

Document Version: 01122014 - Released: 01. December 2014

Table of Contents

Table of Content	2
openQRM base architecture - "in short"	5
In general	5
Objects in openQRM	5
Object dependencies	5
openQRM hooks	5
Recommendation for selecting the operating system for the openQRM Server:	5
Basic openQRM Installation	6
Recommendation for the first Steps after the initial installation	7
Local deployment versus network deployment	8
Examples for "local-deployment":	8
Examples for "network-deployment":	8
Local deployment – automatic installation and disk cloning	9
Recipe: Local VM deployment with Citrix (localboot)	9
Cloning the Citrix VM image	11
Recipe: Local VM deployment with KVM (localboot)	12
Cloning/snapshotting the KVM VM image	14
Clones and snapshots can be efficiently used for system deployments.	15
Recipe: Local VM deployment with Libvirt (localboot)	16
Cloning/snapshotting the Libvirt VM image	18
Clones and snapshots can be efficiently used for system deployments.	18
Recipe: Local VM Deployment with LXC (localboot)	19
Cloning/snapshotting the LXC VM image	20
Clones and snapshots can be efficiently used for system deployments.	21
Recipe: Local VM deployment with openVZ (localboot)	22
Cloning/snapshotting the openVZ VM image	24
Clones and snapshots can be efficiently used for system deployments.	24
Recipe: Local VM deployment with Xen (localboot)	25
Cloning/snapshotting the Xen VM image	27
Clones and snapshots can be efficiently used for system deployments.	27
Recipe: Local VM deployment with VMware ESX (localboot)	29
Recipe: Integrating existing, local-installed systems with Local-Server	31
Recipe: Integrating existing, local-installed VM with Local-Server	31
Recipe: Local deployment with Cobbler (automatic Linux installations)	32
Recipe: Local deployment with Clonezilla (automatic disk cloning)	33
Recipe: Local deployment with FAI (automatic Linux installations)	35
Recipe: Local deployment with LinuxCOE (automatic Linux installations)	36
Recipe: Local deployment with Opsi (automatic Windows installations)	37
Network Deployment - DHCPD/TFTPD/Network-Storage	38
Summary of Network-Deployment	38
Recipe: Network VM deployment with Citrix (networkboot)	38
Recipe: Network VM deployment with KVM (networkboot)	40
Recipe: Network VM deployment with Libvirt (networkboot)	42
Recipe: Network VM deployment with Xen (networkboot)	45
Recipe: Network VM deployment with VMware ESX (networkboot)	47
Recipe: Network Deployment with AOE-Storage	49
Recipe: Network deployment with iSCSI-Storage	50

Recipe: Network deployment with NFS-Storage	51
Recipe: Network deployment with LVM-Storage	52
Recipe: Network deployment with SAN-Boot-Storage	53
Recipe: Network Deployment with TmpFs-Storage	54
Recipe: Populating network deployment images via the Image-Shelf	55
Recipe: Creating new kernel for network deployment	56
High availability	57
High availability for Servers (infrastructure level) – built-in	57
High availability for services (application level) - LCMC	57
High availability for openQRM Server	58
Workflow automation	59
Automated IP assignment for the openQRM network with DHCPD	59
Automated DNS management for the openQRM network with DNS	59
Automated application deployment with Puppet	59
Automated monitoring with Nagios and Icinga	60
Automatic configuration	60
Custom Server service check configuration	60
Automated monitoring with Zabbix	60
Automated monitoring with Collectd	60
Automated power management with WOL (wake-up-on-lan)	61
.	61
Administration	62
Accessing remote desktops and VM consoles with NoVNC	62
Web SSH-Access to remote systems with SshTerm	62
LVM Device and Volume Group Configuration	62
Network-device and Bridge Configuration	62
.	62
.	62
openQRM Cloud	63
Public and Private Cloud Computing	63
openQRM Cloud SOAP-Webservice	65
Hybrid Cloud Computing – Migrating services between Public and Private Clouds	66
Manage and automate public and private clouds	66
Enterprise Features	69
Cloud-Zones: Managing openQRM Clouds in multiple IT Locations	69
Cloud integration with E-Commerce systems via Cloud-Shop	70
Centralized user-management with LDAP	70
Automated IP/Network/VLAN management with IP-Mgmt	70
.	71
Automated IT-Documentation with I-do-it	71
Support for the Windows Operating System	71
Automated power-management with IPMI	72
Network card bonding for network deployment	72
Automated event mailer to forward critical events in openQRM	73
Secure, remote access for the openQRM Enterprise support team	73
Development	74
Development Plugin	74
Template Plugin	74

Debugging	75
Debugging openQRM actions and commands	75
Appendix	76
Explanation of the openQRM Cloud configuration parameters:	76
Contact	79

openQRM base architecture - “in short”

In general

A Server (previously named “Appliance” in openQRM Version < 5.2) in openQRM represents a “service” e.g. a Web Application Server running on a some kind of operating system on a specific type of a Virtual Machine, on a specific host with a specific complete pre-configuration.

Objects in openQRM

Master Object: server

Server Sub Objects: kernel, Image, resource and SLA definition

Other Objects: storage, event

Object dependencies

1. A server is created from the sub components.
2. An Image object depends on a Storage object (an Image is always located on a Storage).
3. The Resource of an Server can be "ex-changed"

openQRM hooks

When a Server is created/started/stopped/removed openQRM triggers an “server hook” which can be implemented by plugins. By this “server hook” the openQRM server transfers the responsibility to do specific actions defined by the plugin. Hooks are also available for other openQRM objects such as resources and events. Please check the development plugin for a detailed openQRM Hooks documentation.

Recommendation for selecting the operating system for the openQRM Server:

The deployment and management functionalities of openQRM are independent from the operating system of the openQRM server itself e.g. openQRM running on a Debian system can of course deploy and manage CentOS/RH based systems. While openQRM is designed to run and work on all kinds of different Linux distributions we still recommend the following operating system for the openQRM Server:

Debian, Ubuntu:

full supported, all functionalities works out-of-the-box

CentOS, Redhat Enterprise, Suse:

full supported, some functionalities may need additional post-configuration

Basic openQRM Installation

Install the following packages: make, subversion

Checkout openQRM (or obtain the openQRM Enterprise Edition):

```
svn checkout http://svn.code.sf.net/p/openqrm/code/trunk openqrm
```

Change to the „trunk/openqrm/src/“ directory

```
cd openqrm/src/
```

To build openQRM run „make“

```
make
```

To install openQRM in /usr/share/openqrm now run „make install“

```
make install
```

By default the next step will initialize openQRM server with standard HTTP Protocol.

If you would like to initialize openQRM server with HTTPS please edit:

```
vi /usr/share/openqrm/etc/openqrm-server.conf
```

and adjust:

```
OPENQRM_WEB_PROTOCOL="http"
```

to

```
OPENQRM_WEB_PROTOCOL="https"
```

To start (and initialize openQRM at first startup) run „make start“

```
make start
```

To stop openQRM and all its plugin services run „make stop“

```
make stop
```

To update a running openQRM server please run „svn up && make update“

```
svn up && make update
```

openQRM version =< 5.0:

Please notice that after „make update“ you need to re-configure the plugin boot-services!

Hint:

For Redhat-based Linux Distributions (e.g. CentOS) SELinux and the iptables firewall must be disabled before the “make start” action!

Hint:

openQRM supports a Mysql or Postgres database backend. If Postgres is used please make sure to have the “php-pgsql” PHP Apache module package installed.

Recommendation for the first Steps after the initial installation

Update `/usr/share/openqrm/plugins/dns/etc/openqrm-plugin-dns.conf` and set the `OPENQRM_SERVER_DOMAIN` parameter to a domain name for your openQRM network. By default it is set to:

```
OPENQRM_SERVER_DOMAIN="oqnet.org"
```

To configure this parameter please use the “Configure” Action in the Plugin-Manager.

Then enable and start the DNS plugin via the plugin manager. This will full automatically pre-configure and start a Bind DNS server on openQRM for the configured domain.

The DNS Plugin now serves the hostname resolving on the openQRM management network and fully automatically adds/removes Server names and their resources ip addresses to the DNS zones.

As a second step please enable the DHCPD plugin. Enabling the DHCPD plugin will automatically create a dhcpd configuration at

```
/usr/share/openqrm/plugins/dhcpd/etc/dhcpd.conf
```

By default the pre-configuration provides the full range of IP address from your openQRM management network. You may want to take a look at the automatic generated configuration and adjust it to your needs. Now start the DHCPD plugin and it will automatically provide IP addresses from your management network for every new system (resource) in openQRM.

If you decide for network deployment please also enable and start the TFTP plugin.

To get more information what the difference is between “local deployment” and “network deployment” in openQRM please continue with the next chapter about “Local deployment versus network deployment”.

Local deployment versus network deployment

Using any kind of “local-deployment” method in openQRM results in a system with its operating system deployed to its local disk. For physical system this is normally one (or more) physical harddisks, for virtual machines the local disk can be any type of storage attached to the virtualization host running the VM (local disk, iSCSI, SAN, NAS, NFS, a distributed and/or clustered storage etc.). Recommended is to use one or more “remote” high available network storage systems to host the storage space for the virtualization hosts.

Examples for “local-deployment”:

- Automatic installation (or disk-cloning) to the harddisk of a physical server (e.g. FAI, LinuxCOE, Cobbler, Opsi, Clonezilla)
- Automatic installation (or disk-cloning) to a virtual harddisk of a virtual machine (e.g. FAI, LinuxCOE, Cobbler, Opsi, Clonezilla in combination with Citrix (localboot), KVM (localboot), Xen (localboot))
- Image-based provisioning of a virtual harddisk of a virtual machine (clone/snap)(e.g. Citrix (localboot), KVM (localboot), Libvirt (localboot), Xen (localboot), openVZ (localboot), LXC (localboot))

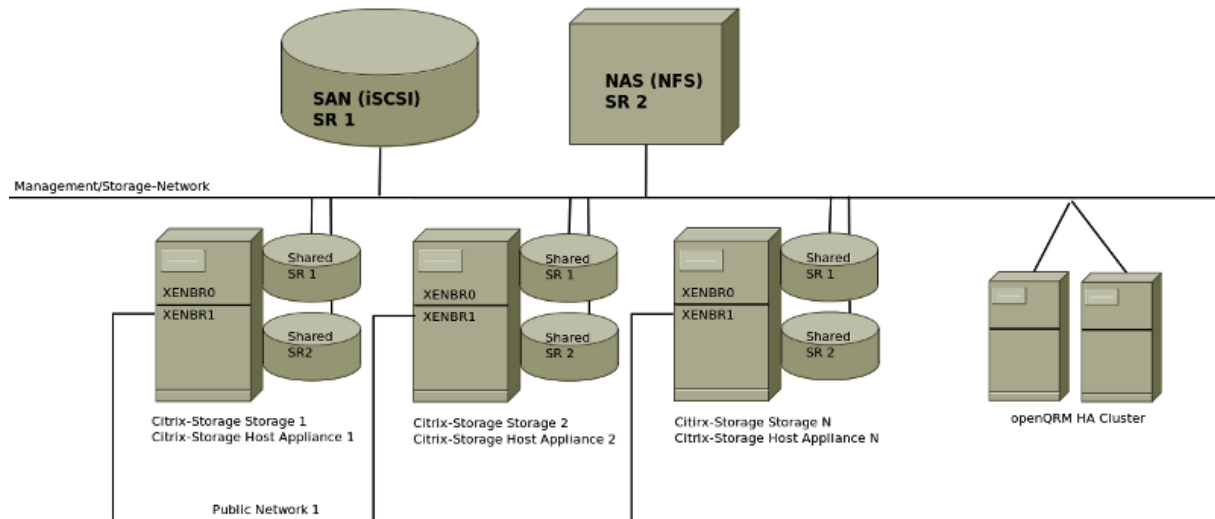
Using any kind of “network-deployment” in openQRM results in a system with its operating system directly located on and deployed from a “remote” storage system. That means those systems won't use any kind of local disk but a remote network storage for its root-filesystem.

Examples for “network-deployment”:

- Image-based provisioning of a physical systems (clone/snap) (e.g. a physical server in combination with any kind of network-storage plugin such as AOE-Storage, iSCSI-Storage, NFS-Storage, LVM-Storage and TMPFS-Storage)
- Image-based provisioning of a virtual machine (clone/snap) (e.g. Citrix (networkboot), KVM (networkboot), Libvirt (networkboot) Xen (networkboot), VMware (networkboot) in combination with any kind of network-storage plugin such as AOE-Storage, iSCSI-Storage, NFS-Storage, LVM-Storage and TMPFS-Storage)

Local deployment – automatic installation and disk cloning

Recipe: Local VM deployment with Citrix (localboot)



1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns (optional), dhcpd, citrix, tftpd, local-server, novnc
3. Configure the citrix boot-service

Please check the Citrix help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration.

Adapt the following configuration parameters according to your virtualization host bridge configuration

CITRIX_MANAGEMENT_INTERFACE

CITRIX_EXTERNAL_INTERFACE

In case Citrix (localboot) virtual machines should be deployed via the openQRM Cloud please also configure:

CITRIX_DEFAULT_SR_UUID

CITRIX_DEFAULT_VM_TEMPLATE

4. Select a separated physical server as the Citrix virtualization host (needs VT/Virtualization Support available and enabled in the system BIOS)

Install Citrix XenServer on the physical system dedicated as the Citrix virtualization host

Hint:

Enable SSH login on the Citrix XenServer

The openQRM NoVNC plugin provides access to the Citrix VM console (requires SSH Login enabled on the Citrix XenServer configuration)

5. Copy the Citrix xe" command line-utility to the openQRM system at /usr/bin/xe
6. Auto discover the Citrix XenServer system via the Citrix "autodiscovery". The integration via the auto discovery automatically creates a Citrix virtualization host Server.
7. Create a "Citrix" storage (storage object for the virtual machine images)
 - Use the "Storage Create" Wizard to create a new storage object using the same resource as selected for the "Citrix" Server and set the deployment type to "citrix-deployment"

Hint:

Citrix XenServer virtual machines are stored on a Citrix SR (storage resource).

- You can use the "nfs-storage" and/or the "iscsi-storage" plugin to easily create a NAS- and/or iSCSI datastore to be used as a Citrix storage resource!
- Then use the included SR datastore manager to connect a NAS- and/or iSCSI Datastore.

1. Create a "Citrix (localboot)" virtual machine

- Use the "Server Create" wizard to create a new Server object
- Create a New Citrix (localboot) VM resource in the 2. step
- Select the previously created "Citrix" Server object for creating the new Citrix (localboot) VM.
- Create a New Citrix VM image in the 3. step
- Select the previously created "Citrix" storage object for creating the new Citrix VM image volume. Creating a new volume automatically creates a new image object in openQRM

Please notice:

For an initial installation of the image of the virtual machine you may want to edit the image details to attach an automatic operating system installation on first start up.

Hint:

Easily create an automatic installation profiles which can be attached via openQRM "Install-from-Template" mechanism to a Citrix VM image object via the following plugins: Cobbler, FAI, LinuxCOE, Opsi, Clonezilla

- Start the Server

Starting the Server automatically combines the Citrix (localboot) VM (Resource Object) and the Citrix VM volume (image object). Stopping the Server will "uncouple" resource and image.

Hint:

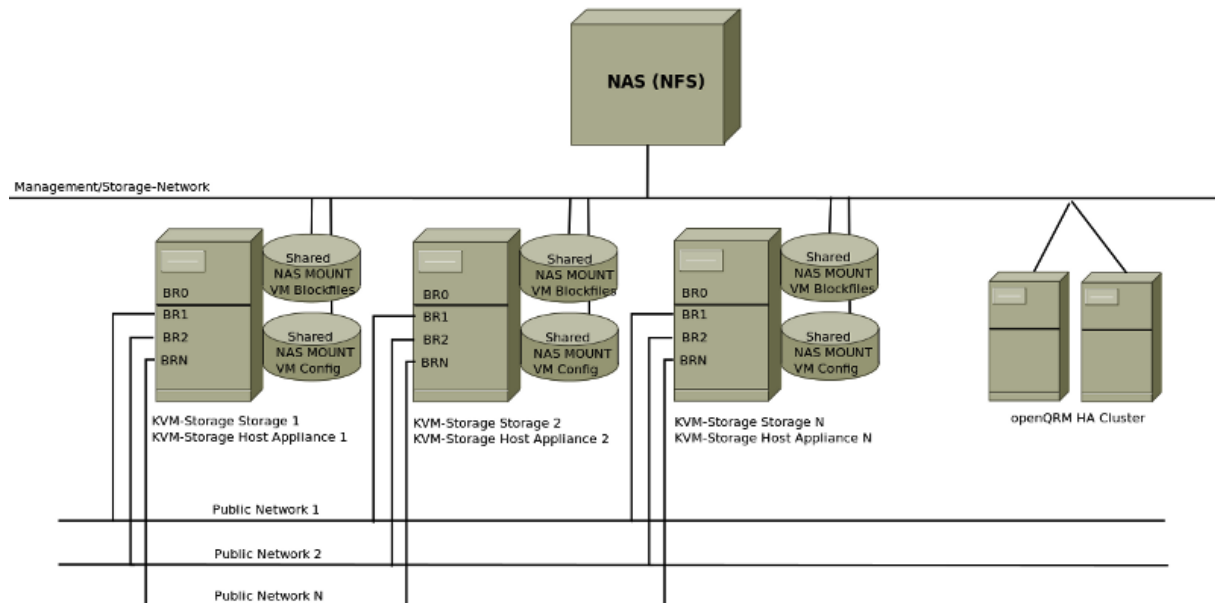
To enable further management functionalities of openQRM "within" the virtual machines operating system please install the "openqrm-local-vm-client" in the VM. Please refer to "Integrating existing, local-installed VM with Local-Server".

Cloning the Citrix VM image Through the integrated Citrix volume management existing, pre-installed images can be duplicated with the “Clone” mechanism. Clones can be efficiently used for system deployments.

Clone

Cloning a volume of a Citrix VM image object results in a “1-to-1 copy” of the source volume. A new image object for this volume copy is automatically created in openQRM. Creating a clone depends on the actual volume size, hardware and network performance, etc. All data from the origin volume are transferred/copied to the source volume.

Recipe: Local VM deployment with KVM (localboot)



Install and setup openQRM on a physical system or on a virtual machine

1. Enable the following plugins: dns (optional), dhcpd, tftpd, local-server, novnc, device-manager, network-manager, kvm
2. Configure the KVM boot-service

Please check the KVM help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration. This network-configuration is only used by the openQRM Cloud to auto-create VMs.

OPENQRM_PLUGIN_KVM_BRIDGE_NET1

OPENQRM_PLUGIN_KVM_BRIDGE_NET2

OPENQRM_PLUGIN_KVM_BRIDGE_NET3

OPENQRM_PLUGIN_KVM_BRIDGE_NET4

OPENQRM_PLUGIN_KVM_BRIDGE_NET5

In case "kvm-bf-deployment" (blockfile deployment) is used also:

OPENQRM_PLUGIN_KVM_FILE_BACKEND_DIRECTORIES

3. Select a physical server as the virtualization host (needs VT/virtualization support available and enabled in the system BIOS)

Please notice:

This system dedicated to be the virtualization host can be the openQRM server system itself or it can be a “remote” physical server integrated via the “Local-Server plugin”.

Please check “Integrating existing, local-installed systems with Local-Server”

Please make sure this system meets the following requirements:

1.
 - Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!
 - For KVM LVM storage: One (or more) LVM volume group(s) with free space dedicated for the KVM VM storage
 - For KVM blockfile storage: free space dedicated for the KVM VM storage , eventually using remote NAS/NFS storage space
 - One or more bridges configured for the virtual machines (e.g. br0, br1, etc.)

1. Create a “KVM” Server (Server object for the virtualization host)
In case the openQRM server is the physical system dedicated to be the virtualization host please use the “Server Create” wizard to create a new Server object using the openQRM server resource.
2. After creating the Server edit it and set “Virtualization” to “KVM Host”

In case a “remote” physical system, integrated via the “Local-Server” Plugin, is the virtualization host the system integration already created an Server for this system. Simply edit it and set “Virtualization” to “KVM Host”

3. Create a “KVM” storage (storage object for the virtual machine images)

Use the “Storage Create” wizard to create a new storage object using the same resource as selected for the “KVM” Server and set the deployment type to either “kvm-lvm-deployment”

The image volumes are created as LVM logical volumes on available LVM volume groups on the storage system

or “kvm-bf-deployment”

The image volumes are created as blockfiles within configurable directories

Please check `/usr/share/openqrm/plugins/kvm/etc/openqrm-plugin-kvm.conf` for the configuration options

4. Create a “KVM (localboot)” virtual machine

- Use the “Server Create” wizard to create a new Server object
- Create a new KVM (localboot) VM resource in the 2. step
- Select the previously created “KVM” Server object for creating the new KVM (localboot) VM.

Please notice:

For an initial installation of the image of the virtual machine you may want to configure the VM to boot from a prepared ISO image to start a manual or automatic operating system installation on first start up.

Hint:

Easily create an automatic installation ISO image with the LinuxCOE plugin! The automatic installation ISO images of LinuxCOE will be automatically available to be selected on the virtualization host Server at /linuxcoe-iso

- Create a new KVM VM image in the 3. step
- Select the previously created "KVM" storage object for creating the new KVM VM image volume. Creating a new volume automatically creates a new image object in openQRM

Please notice:

For an initial installation of the image of the virtual machine you may want to edit the image details to attach an automatic operating system Installation on first start up.

Hint:

Easily create an automatic installation profiles which can be attached via openQRM "Install-from-Template" mechanism to a KVM VM image object via the following plugins: Cobbler, FAI, LinuxCOE, Opsi, Clonezilla

- Start the Server

Starting the Server automatically combines the KVM (localboot) VM (resource object) and the KVM VM volume (image object).

Stopping the Server will "uncouple" resource and image.

Hint:

After the operating system installation on the KVM VM volume the VM is normally still set to boot from the ISO Image. To reconfigure the VM to directly boot from its local installed virtual disk follow the steps below:

Stop the Server

Update the virtual machine via the VM manager

Start the Server again

To enable further management functionalities of openQRM "within" the virtual machines operating system please install the "openqrm-local-vm-client" in the VM. Please refer to "Integrating existing, local-installed VM with Local-Server".

Cloning/snapshotting the KVM VM image Through the integrated KVM volume management existing, pre-installed images can be duplicated with the "Clone" and "Snapshot" mechanisms.

Clones and snapshots can be efficiently used for system deployments.

Clone

Cloning a volume of a KVM VM image object results in a “1-to-1 copy” of the source volume. A new image object for this volume copy is automatically created in openQRM. Creating a clone depends on the actual volume size, hardware and network performance, etc. All data from the origin volume are transferred/copied to the source volume.

Snapshot

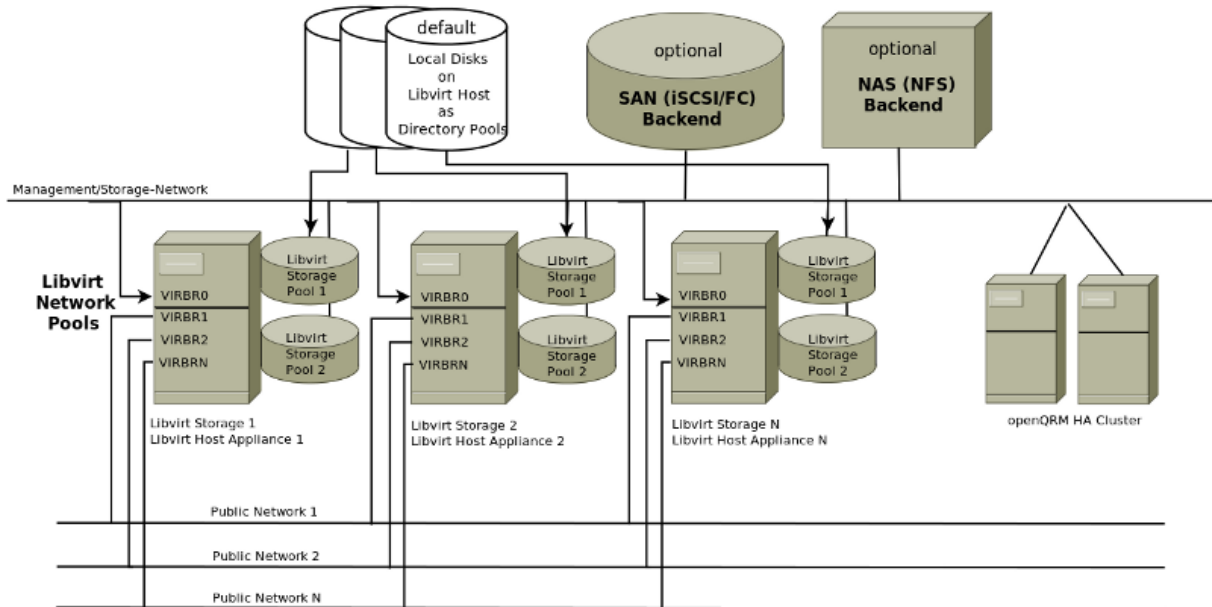
In case of “kvm-lvm-deployment” the underlying storage layer (LVM) provides the snapshot-functionality. A snapshot results in a “1-to-1 copy-on-write” (COW) volume which redirects all “read” calls to the origin and its “write” calls to the preserved storage space for the snapshot. Creating a snapshot just takes a second. It does not involve data transfer nor does it use any actual storage space on the storage. That means a snapshot only saves “changes” to the origin and is a great way to efficiently save storage space for deployments.

Howto

„Virtualization with KVM and openQRM 5.2 on Debian Wheezy“ at <http://www.openqrm-enterprise.com/resources/documentation-howtos/howtos/virtualization-with-kvm-and-openqrm-51-on-debian-wheezy.html>

Recipe: Local VM deployment with Libvirt (localboot)

Libvirt Deployment



1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns (optional), dhcpd, tftpd, local-server, novnc, libvirt
3. Configure the Libvirt boot-service

Please check the Libvirt help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration. This network-configuration is only used by the openQRM Cloud to auto-create VMs.

```
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET1
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET2
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET3
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET4
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET5
OPENQRM_PLUGIN_LIBVIRT_PRIMARY_NIC_TYPE
OPENQRM_PLUGIN_LIBVIRT_ADDITIONAL_NIC_TYPE
OPENQRM_PLUGIN_LIBVIRT_VM_DEFAULT_VNCPASSWORD
```


4. Select a physical server as the virtualization host (needs VT/virtualization support available and enabled in the system BIOS)

Please notice:

This system dedicated to be the virtualization host can be the openQRM server system itself or it can be a "remote" physical server integrated via the "Local-Server plugin".

Please check "Integrating existing, local-installed systems with Local-Server"

Please make sure this system meets the following requirements:

Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!

5. Create a "Libvirt" Server (Server object for the virtualization host)
In case the openQRM server is the physical system dedicated to be the virtualization host please use the "Server Create" wizard to create a new Server object using the openQRM server resource.
6. After creating the Server edit it and set "Virtualization" to "Libvirt Host"

In case a "remote" physical system, integrated via the "Local-Server" Plugin, is the virtualization host the system integration already created an Server for this system. Simply edit it and set "Virtualization" to "Libvirt Host"

7. A "Libvirt" storage (storage object for the virtual machine images) is automatically created when updating the "Libvirt" Server to a "Libvirt Host".
8. Create a "Libvirt (localboot)" virtual machine

- Use the "Server Create" wizard to create a new Server object
- Create a new Libvirt (localboot) VM resource in the 2. step
- Select the previously created "Libvirt" Server object for creating the new Libvirt (localboot) VM.

Please notice:

For an initial installation of the image of the virtual machine you may want to configure the VM to boot from a prepared ISO image to start a manual or automatic operating system installation on first start up.

Hint:

Easily create an automatic installation ISO image with the LinuxCOE plugin! The automatic installation ISO images of LinuxCOE will be automatically available to be selected on the virtualization host Server at /linuxcoe-iso

- Create a new Libvirt VM image in the 3. step
- Select the previously created "Libvirt" storage object for creating the new Libvirt VM image volume.
Creating a new volume automatically creates a new image object in openQRM

Please notice:

For an initial installation of the image of the virtual machine you may want to edit the image details to attach an automatic operating system Installation on first start up.

Hint:

Easily create an automatic installation profiles which can be attached via openQRM "Install-from-Template" mechanism to a Libvirt VM image object via the following plugins: Cobbler, FAI, LinuxCOE, Opsi, Clonezilla

- Start the Server

Starting the Server automatically combines the Libvirt (localboot) VM (resource object) and the Libvirt VM volume (image object).

Stopping the Server will "uncouple" resource and image.

Hint:

After the operating system installation on the Libvirt VM volume the VM is normally still set to boot from the ISO Image. To reconfigure the VM to directly boot from its local installed virtual disk follow the steps below:

Stop the Server

Update the virtual machine via the VM manager

Start the Server again

To enable further management functionalities of openQRM "within" the virtual machines operating system please install the "openqrm-local-vm-client" in the VM. Please refer to "Integrating existing, local-installed VM with Local-Server".

Cloning/snapshotting the Libvirt VM image Through the integrated Libvirt volume management existing, pre-installed images can be duplicated with the "Clone" and "Snapshot" mechanisms.

Clones and snapshots can be efficiently used for system deployments.

Clone

Cloning a volume of a Libvirt VM image object results in a "1-to-1 copy" of the source volume. A new image object for this volume copy is automatically created in openQRM. Creating a clone depends on the actual volume size, hardware and network performance, etc. All data from the origin volume are transferred/copied to the source volume.

Snapshot

In case of "libvirt-lvm-deployment" the underlying storage layer (LVM) provides the snapshot-functionality. A snapshot results in a "1-to-1 copy-on-write" (COW) volume which redirects all "read" calls to the origin and its "write" calls to the preserved storage space for the snapshot. Creating a snapshot just takes a second. It does not involve data transfer nor does it use any actual storage space on the storage. That means a snapshot only saves "changes" to the origin and is a great way to efficiently save storage space for deployments.

Recipe: Local VM Deployment with LXC (localboot)

1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns(optional), dhcpd, local-server, device-manager, network-manager, lxc
3. Configure the LXC boot-service

Please check the LXC help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration. This network-configuration is only used by the openQRM Cloud to auto-create VMs.

OPENQRM_PLUGIN_LXC_BRIDGE

OPENQRM_PLUGIN_LXC_BRIDGE_NET1

OPENQRM_PLUGIN_LXC_BRIDGE_NET2

OPENQRM_PLUGIN_LXC_BRIDGE_NET3

OPENQRM_PLUGIN_LXC_BRIDGE_NET4

4. Select a physical server as the Virtualization Host

Please notice:

This system dedicated to be the virtualization host can be the openQRM server system itself or it can be a "remote" physical server integrated via the "Local-Server Plugin".

Please check "Integrating existing, local-installed Systems with Local-Server"

Please make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!
- For LXC LVM storage: One (or more) lvm volume group(s) with free space dedicated for the LXC VM storage
- One or more bridges configured for the virtual machines (e.g. br0, br1, etc.)

1. Create a "LXC" Server (Server object for the virtualization host)

In case the openQRM server is the physical system dedicated to be the virtualization host please use the "Server Create" wizard to create a new Server object using the openQRM server resource.

2. After creating the Server edit it and set "Virtualization" to "OpenVZ Host"

In case a “remote” physical system, integrated via the “Local-Server” plugin, is the virtualization host the system integration already created an Server for this system. Simply edit it and set “Virtualization” to “LXC Host”

3. Create a “LXC” storage (storage object for the virtual machine images)

- Use the “Storage Create” wizard to create a new storage object using the same resource as selected for the “LXC” Server and set the deployment type to “lxc-deployment”
- The image volumes are created as LVM logical volumes on available LVM volume groups on the storage system

1. Create a “LXC (localboot)” virtual machine

- Use the “Server Create” wizard to create a new Server object
- Create a New LXC (localboot) VM resource in the 2. step

Select the previously created “LXC” Server object for creating the new LXC (localboot) VM.

- Create a new LXC VM image in the 3. step
- Select the previously created “LXC” storage object for creating the new LXC VM image volume. Creating a new volume automatically creates a new image object in openQRM

Hint:

For an initial installation of the image of the virtual machine the LXC volume manager provides an easy way to upload “ready-made” LXC operating system templates which can be then directly deployed to the LXC volumes. Please check <http://wiki.openvz.org/Download/template/precreated>

- Start the Server

Starting the Server automatically combines the LXC (localboot) VM (resource object) and the LXC VM volume (image object).

Stopping the Server will “uncouple” resource and image.

To enable further management functionalities of openQRM “within” the virtual machines operating system please install the “openqrm-local-vm-client” in the VM. Please refer to “Integrating existing, local-installed VM with Local-Server”.

Cloning/snapshotting the LXC VM image Through the integrated LXC volume management existing, pre-installed images can be duplicated with the “Clone” and “Snapshot” mechanisms.

Clones and snapshots can be efficiently used for system deployments.

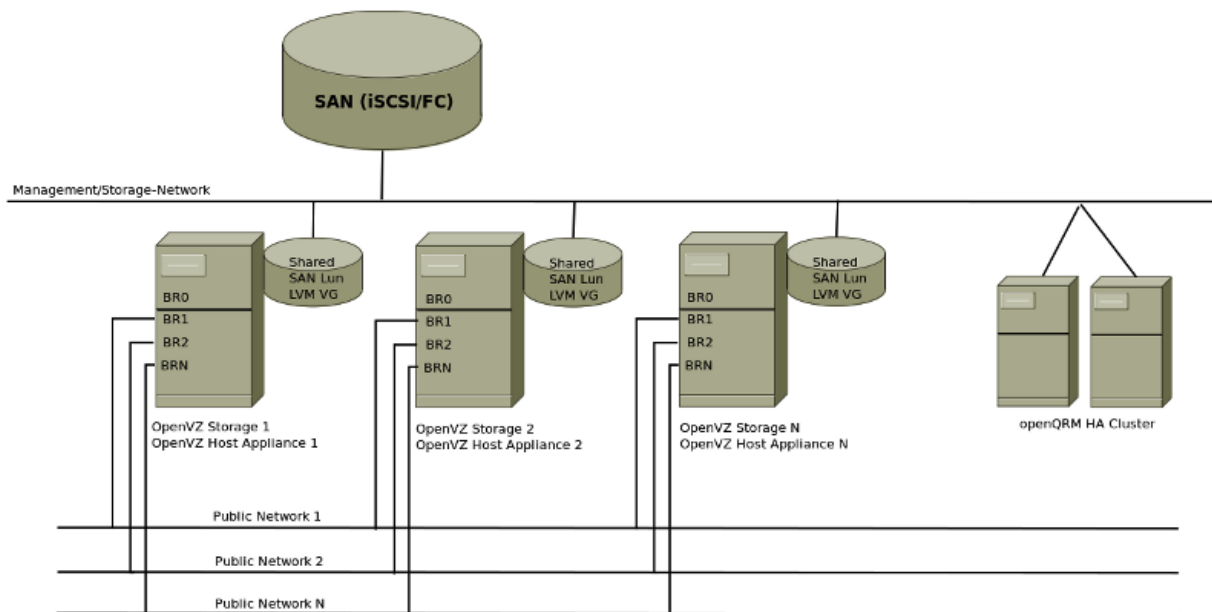
Clone

Cloning a volume of a LXC VM image object results in a “1-to-1 copy” of the source volume. A new image object for this volume copy is automatically created in openQRM. Creating a clone depends on the actual volume size, hardware and network performance, etc. All data from the origin volume are transferred/copied to the source volume.

Snapshot

In case of “lxc-lvm-deployment” the underlying storage layer (LVM) provides the snapshot-functionality. A snapshot results in a “1-to-1 copy-on-write” (COW) volume which redirects all “read” calls to the origin and its “write” calls to the preserved storage space for the snapshot. Creating a snapshot just takes a second. It does not involve data transfer nor does it use any actual storage space on the storage. That means a snapshot only saves “changes” to the origin and is a great way to efficiently save storage space for deployments.

Recipe: Local VM deployment with openVZ (localboot)



Install and setup openQRM on a physical system or on a virtual machine

1. Enable the following plugins: dns(optional), dhcpd, local-server, device-manager, network-manager, openvz
2. Configure the openVZ boot-service

Please check the openVZ help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration. This network-configuration is only used by the openQRM Cloud to auto-create VMs.

```
OPENQRM_PLUGIN_OPENVZ_BRIDGE
OPENQRM_PLUGIN_OPENVZ_BRIDGE_NET1
OPENQRM_PLUGIN_OPENVZ_BRIDGE_NET2
OPENQRM_PLUGIN_OPENVZ_BRIDGE_NET3
OPENQRM_PLUGIN_OPENVZ_BRIDGE_NET4
```

1. Select a physical server as the virtualization host

Please notice:

This System dedicated to be the virtualization host can be the openQRM server system itself or it can be a "remote" physical server integrated via the "Local-Server Plugin".

Please check "Integrating existing, local-installed Systems with Local-Server"

Make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!
- For openVZ LVM storage: One (or more) lvm volume group(s) with free space dedicated for the openVZ VM storage
- One or more bridges configured for the virtual machines (e.g. br0, br1, etc.)

1. Create a “openVZ” Server (Server object for the virtualization host)

In case the openQRM server is the physical system dedicated to be the virtualization host please use the “Server Create” wizard to create a new Server object using the openQRM server resource.

2. After creating the Server edit it and set “Virtualization” to “OpenVZ Host”

In case a “remote” physical system, integrated via the “Local-Server” plugin, is the virtualization host the system integration already created an Server for this system. Simply edit it and set “Virtualization” to “openVZ Host”

3. Create a “openVZ” storage (storage object for the virtual machine images)

Use the “Storage Create” wizard to create a new storage object using the same resource as selected for the “openVZ” Server and set the deployment type to “openvz-deployment”

The image volumes are created as LVM logical volumes on available LVM volume groups on the storage system

4. Create a “openVZ (localboot)” virtual machine

- Use the “Server Create” wizard to create a new Server object
- Create a new openVZ (localboot) VM resource in the 2. step
- Select the previously created “openVZ” Server object for creating the new openVZ (localboot) VM.
- Create a new openVZ VM image in the 3. step
- Select the previously created “openVZ” storage object for creating the new openVZ VM image volume. Creating a new volume automatically creates a new image object in openQRM.

Hint:

For an initial installation of the image of the virtual machine the openVZ manager provides an easy way to upload “ready-made” openVZ operating system templates which can be then directly deployed to the openVZ volumes. Please check <http://wiki.openvz.org/Download/template/precreated>

- Start the Server

Starting the Server automatically combines the openVZ VM (resource object) and the openVZ VM volume (image object).

Stopping the Server will “uncouple” resource and image.

To enable further management functionalities of openQRM “within” the virtual machines operating system please install the “openqrm-local-vm-client” in the VM. Please refer to “Integrating existing, local-installed VM with Local-Server”.

Cloning/snapshotting the openVZ VM image Through the integrated openVZ volume management existing, pre-installed Images can be duplicated with the “Clone” and “Snapshot” mechanism.

Clones and snapshots can be efficiently used for system deployments.

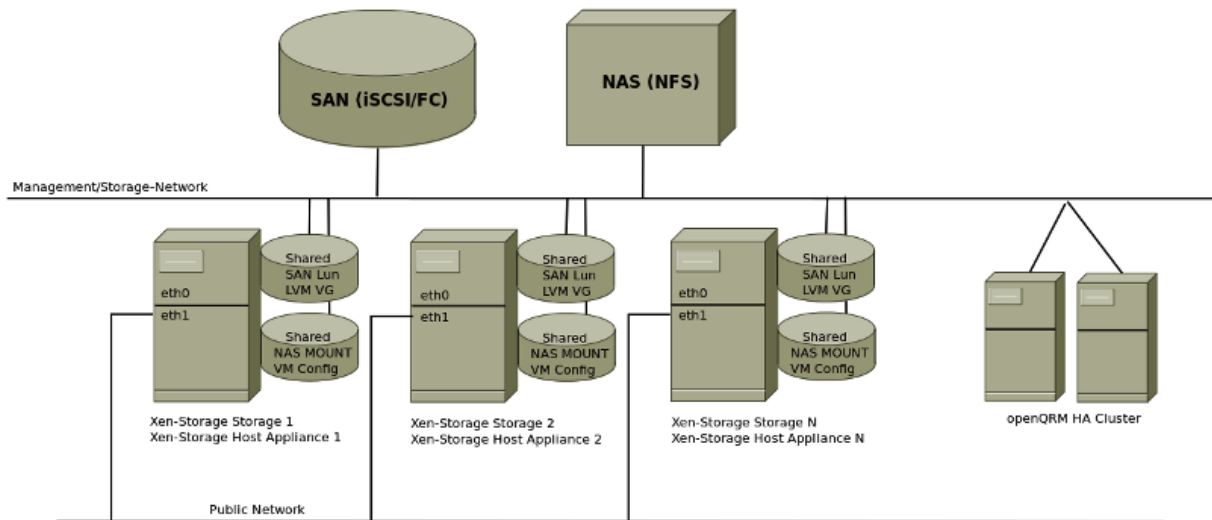
Clone

Cloning a volume of a openVZ VM image object results in a “1-to-1 copy” of the source volume. A new image object for this volume copy is automatically created in openQRM. Creating a clone depends on the actual volume size, hardware and network performance, etc. All data from the origin volume are transferred/copied to the source volume.

Snapshot

In case of “openvz-deployment” the underlying storage layer (LVM) provides the snapshot-functionality. A snapshot results in a “1-to-1 copy-on-write” (COW) volume which redirects all “read” calls to the origin and its “write” calls to the preserved storage space for the snapshot. Creating a snapshot just takes a second. It does not involve data transfer nor does it use any actual storage space on the storage. That means a snapshot only saves “changes” to the origin and is a great way to efficiently save storage space for deployments.

Recipe: Local VM deployment with Xen (localboot)



Install and setup openQRM on a physical system or on a virtual machine

1. Enable the following plugins: dns(optional), dhcpd, tftpd, local-server, novnc, device-manager, network-manager, xen
2. Configure the Xen boot-service

Please check the Xen help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration

OPENQRM_PLUGIN_XEN_INTERNAL_BRIDGE

OPENQRM_PLUGIN_XEN_EXTERNAL_BRIDGE

In case "xen-bf-deployment" (blockfile deployment) is used also:

OPENQRM_PLUGIN_XEN_FILE_BACKEND_DIRECTORIES

3. Select a physical server as the virtualization host (needs VT/virtualization support available and enabled in the system BIOS)

Please notice:

This system dedicated to be the virtualization host can be the openQRM server system itself or it can be a "remote" physical server integrated via the "Local-Server Plugin".

Please check "Integrating existing, local-installed Systems with Local-Server"

Please make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!

- For Xen LVM storage: One (or more) lvm volume group(s) with free space dedicated for the Xen VM storage
- For Xen blockfile storage: free space dedicated for the Xen VM storage , eventually using remote NAS/NFS storage space
- One or more bridges configured for the virtual machines (e.g. br0, br1, etc.)

1. Create a “Xen” Server (Server object for the virtualization host)

In case the openQRM server is the physical system dedicated to be the virtualization host please use the “Server Create” wizard to create a new Server object using the openQRM server resource.

2. After creating the Server edit it and set “Virtualization” to “Xen Host”

In case a “remote” physical system, integrated via the “Local-Server” plugin, is the virtualization host the system integration already created an Server for this system. Simply edit it and set “Virtualization” to “Xen Host”.

3. Create a “Xen VM” storage (storage object for the virtual machine images)

Use the “Storage Create” wizard to create a new storage object using the same resource as selected for the “Xen” Server and set the deployment type to either

“xen-lvm-deployment”.

The image volumes are created as LVM logical volumes on available LVM volume groups on the storage system or

“xen-bf-deployment”

The image volumes are created as blockfiles within configurable directories

Please check `/usr/share/openqrm/plugins/xen/etc/openqrm-plugin-xen.conf` for the configuration options

4. Create a “Xen (localboot)” virtual machine

- Use the “Server Create” wizard to create a new Server object
- Create a New Xen (localboot) VM resource in the 2. step
- Select the previously created “Xen” Server object for creating the new Xen (localboot) VM.

Please notice:

For an initial installation of the image of the virtual machine you may want to configure the VM to boot from a prepared ISO image to start a manual or automatic operating system installation on first start up.

Hint:

Easily create an automatic installation ISO image with the LinuxCOE plugin!

- The automatic installation ISO images of LinuxCOE will be automatically available to be selected on the virtualization host Server at `/linuxcoe-iso`
- Create a new Xen VM image in the 3. step
- Select the previously created “Xen” storage object for creating the new Xen VM image volume. Creating a new volume automatically creates a new image object in openQRM

Please notice:

For an initial installation of the image of the virtual machine you may want to edit the image details to attach an automatic operating system installation on first start up.

Hint:

Easily create an automatic installation profile that can be attached via openQRM “Install-from-Template” mechanism to a Xen VM image object via the following plugins: Cobbler, FAI, LinuxCOE, Opsi, Clonezilla

- Start the Server

Starting the Server automatically combines the Xen (localboot) VM (resource object) and the Xen VM volume (image object).

Stopping the Server will “uncouple” resource and image.

Hint:

After the operating system installation on the Xen VM volume the VM is normally still set to boot from the ISO image. To reconfigure the VM to directly boot from its local installed virtual disk follow the steps below:

Stop the Server

Update the virtual machine via the plugins VM manager

Start the Server again

To enable further management functionalities of openQRM “within” the virtual machines operating system please install the “openqrm-local-vm-client” in the VM. Please refer to “Integrating existing, local-installed VM with Local-Server”.

Cloning/snapshotting the Xen VM image Through the integrated Xen volume management existing, pre-installed images can be duplicated with the “Clone” and “Snapshot” mechanism.

Clones and snapshots can be efficiently used for system deployments.

Clone

Cloning a volume of a Xen VM image object results in a “1-to-1 copy” of the source volume. A new image object for this volume copy is automatically created in openQRM. Creating a clone depends on the actual volume size, hardware and network performance, etc. All data from the origin volume are transferred/copied to the source volume.

Snapshot

In case of “xen-lvm-deployment” the underlying storage layer (LVM) provides the snapshot-functionality. A snapshot results in a “1-to-1 copy-on-write” (COW) volume which redirects all “read” calls to the origin and its “write” calls to the preserved storage space for the snapshot. Creating a snapshot just takes a second. It

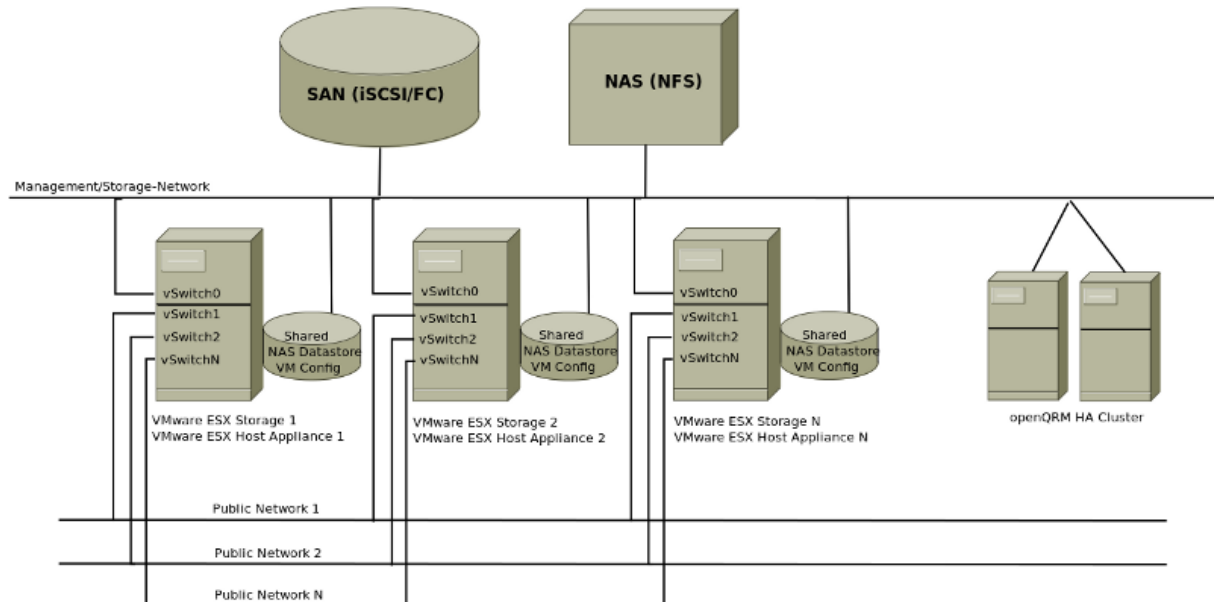
does not involve data transfer nor does it use any actual storage space on the storage. That means a snapshot only saves “changes” to the origin and is a great way to efficiently save storage space for deployments.

Howto

„Virtualization with Xen and openQRM 5.2 on Debian Wheezy“ at <http://www.openqrm-enterprise.com/resources/documentation-howtos/howtos/virtualization-with-xen-and-openqrm-51-on-debian-wheezy.html>

Recipe: Local VM deployment with VMware ESX (localboot)

VMware ESX Deployment



1. Install and setup openQRM on a physical system or on a virtual machine
2. Install the latest VMware Vsphere SDK on the openQRM server system
3. Enable the following plugins: dns (optional), dhcpd, tftpd, local-server, novnc, vmware.esx
4. Configure the VMware ESX Boot-service

Please check the VMware ESX help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration

OPENQRM_VMWARE_ESX_INTERNAL_BRIDGE

OPENQRM_VMWARE_ESX_EXTERNAL_BRIDGE_2

OPENQRM_VMWARE_ESX_EXTERNAL_BRIDGE_3

OPENQRM_VMWARE_ESX_EXTERNAL_BRIDGE_4

OPENQRM_VMWARE_ESX_EXTERNAL_BRIDGE_5

In case you are using openQRM Cloud with Xen (network-deployment) please also set

OPENQRM_VMWARE_ESX_CLOUD_DATASTORE

OPENQRM_VMWARE_ESX_GUEST_ID

OPENQRM_VMWARE_ESX_CLOUD_DEFAULT_VM_TYPE

5. Install one (or more) VMware ESX server within the openQRM management network
6. Use the VMware ESX Discovery to discover VMware-ESX system(s)

After the discovery please integrate the VMware-ESX system(s) by providing the administrative credentials.

Please notice:

Integrating the VMware ESX system automatically creates and pre-configures a VMware ESX Host Server and Storage in openQRM.

7. Create a “VMware ESX (localboot)” virtual machine
8. Use the “Server Create” wizard to create a new Server object
9. Create a new VMware ESX (localboot) VM resource in the 2. step
10. Select the previously created “VMware ESX” Server object for creating the new VMware ESX (localboot) VM.
11. **Please notice:**
For an initial installation of the image of the virtual machine you may want to configure the VM to boot from a prepared ISO image to start a manual or automatic operating system installation on first start up.
12. **Hint:**
Easily create an automatic installation ISO image with the LinuxCOE plugin! The automatic installation ISO images of LinuxCOE will be automatically available via NFS at openQRM-server: /linuxcoe-iso. It is recommended to add this NFS export as ISO Datastore to the ESX Host.
13. The VMware ESX VM image is automatically created when creating the VMware ESX VM in the 2. step. The Server Wizard will therefore skip the step 3 (Selecting/Creating a new Image).
14. **Please notice:**
For an initial installation of the image of the virtual machine you may want to edit the image details to attach an automatic operating system Installation on first start up.
15. **Hint:**
Easily create an automatic installation profiles which can be attached via openQRM “Install-from-Template” mechanism to a VMware ESX VM image object via the following plugins: Cobbler, FAI, LinuxCOE, Opsi, Clonezilla
16. Start the Server
17. Starting the Server automatically combines the VMware ESX (localboot) VM (resource object) and the VMware ESX VM volume (image object).
18. Stopping the Server will “uncouple” resource and image.
19. **Hint:**
After the operating system installation on the VMware ESX VM volume the VM is normally still set to boot from the ISO Image. To reconfigure the VM to directly boot from its local installed virtual disk follow the steps below:
20. *Stop the Server*
21. *Update the virtual machine via the VM manager*
22. *Start the Server again*
23. To enable further management functionalities of openQRM “within” the virtual machines operating system please install the “openqrm-local-vm-client” in the VM. Please refer to “Integrating existing, local-installed VM with Local-Server”.

Recipe: Integrating existing, local-installed systems with Local-Server

1. Copy (scp) the "openqrm-local-server" utility to an existing, local-installed server in your network

```
scp /usr/share/openqrm/plugins/local-server/bin/openqrm-local-server [ip-address-of-existing-server]:/tmp/
```

Login to the remote system via ssh and execute the "openqrm-local-server" utility on the remote system:

```
ssh [ip-address-of-existing-server]
```

```
/tmp/openqrm-local-server integrate -u openqrm -p openqrm -q [ip-address-of-openQRM-server] -i eth0  
[-s http/https]
```

The system now appears in the openQRM server as new resource . It should be now set to "network-boot" in its BIOS to allow dynamic assign- and deployment. The resource can now be used to e.g. create a new "storage-server" within openQRM.

After setting the system to "network-boot" in its BIOS it also can be used to deploy server-images from different types.

To remove a system from openQRM integrated via the local-server plugin run the "openqrm-local-server" utility again on the remote system

```
/tmp/openqrm-local-server remove -u openqrm -p openqrm -q [ip-address-of-openQRM-server] [-s http/https]
```

Recipe: Integrating existing, local-installed VM with Local-Server

For local-installed virtual machines (e.g. KVM (localboot), Xen (localboot), LXC (localboot), openVZ (localboot)) which have access to the openQRM network there is an "openqrm-local-vm-client" available. This "openqrm-local-vm-client" just starts and stops the plugin-boot-services to allow further management functionality. Monitoring and openQRM actions are still running on behalf of the VM host.

2. Download/Copy the "openqrm-local-vm-client" from the Local-Server help section (Local VMs) to a local installed VM

```
scp openqrm-local-vm-client [ip-address-of-existing-server]:/tmp/
```

3. Execute the "openqrm-local-vm-client" on the VM

```
/tmp/openqrm-local-vm-client
```

The "openqrm-local-vm-client" fully automatically configures itself.

Recipe: Local deployment with Cobbler (automatic Linux installations)

1. Install a Cobbler install server on a dedicated system (physical server or VM)

Notice:

Configure the Cobbler install service to not run the "dhcpd" service on the Cobbler system itself. The openQRM "dhcpd" service provided by the Dhcpd-Plugin will be used instead.

1. Additionally install the "screen" package on the Cobbler server.
2. Integrate a Cobbler install server into openQRM via the "local-server" plugin
Please check "Integrating existing, local-installed Systems with Local-Server"
3. Create a new storage server from the type "cobbler-deployment" using the Cobbler systems resource
4. Add Cobbler "snippets" (kickstart-templates) to the Cobbler install server via the Cobbler Web UI and combine them to Installation profiles.

Hint:

Add the Cobbler snippet `openqrm_client_auto_install.snippets` from the Cobbler plugins help section to your Cobbler profiles to automatically install the openQRM client on the provisioned systems.

VM images for local-deployment (e.g. Citrix (localboot), KVM (localboot), Xen (localboot)) can now be set to "install-from-template" via Cobbler in the image "Edit" section.

Local-deployment Images for physical system are created through the Cobbler plugins "Image Admin" section. In the image "Edit" section they can be set to "install-from-template" via Cobbler in the same way as virtual machines.

Starting an Server with a Local-deployment Image configured via an "Install-from-Template" Cobbler installation profile automatically "transfers" the system to the Cobbler server for the initial OS installation (PXE/Network-boot). While the automatic OS installation is running on behalf of the Cobbler servers responsibility openQRM takes back the control of the system and prepares it for local-booting.

After rebooting from the automatic installation the system is fully integrated into openQRM.

Hint:

The "Install-from-Template" mechanism with Cobbler can be also used in the openQRM Cloud! Just make sure the image masters provided in the openQRM Cloud are configured with "Install-from-Template".

Recipe: Local deployment with Clonezilla (automatic disk cloning)

Install and setup openQRM on a physical system or on a virtual machine

1. Enable the following plugins: dns (optional), dhcpd, tftpd, device-manager, clonezilla
2. Create a new storage server from the type "clonezilla-deployment"

Please notice:

This system dedicated to be the Clonezilla template storage can be the openQRM server system itself or it can be a "remote" physical server integrated via the "Local-Server Plugin".

Please check "Integrating existing, local-installed systems with Local-Server"

Please make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!
- One or more lvm volume group(s) with free space dedicated for the Clonezilla Templates

1. Create Clonezilla template using the "Template Admin" menu
2. Populate the Clonezilla template

- "Grab" a local-installed system (populating the template)
- Network-boot (PXE) a physical system or a virtual machine into the "idle" state in openQRM (network-boot and automatic integration)
- In the "Template Admin" use the "Deploy" Button to activate the "grab" phase
- In the second step select the "idle" resource (the system with the local-installed OS on its harddisk)
- The system now reboots into the "grab" phase, starts clonezilla, mounts the template storage and transfers its disk content to the Clonezilla template

Hint:

The "Drain" action will empty the specific selected template so it can be re-used for the "grab" phase.

1. Deployment of Clonezilla templates

- Use the "Image Admin" to create a new Clonezilla volume
- Creating a new volume automatically creates a new image object in openQRM

Clonezilla images can now be set to "install-from-template" via Clonezilla in the image "Edit" section. Edit the created Clonezilla image and select "Automatic Clone from Template" for the automatic installation. Then select the Clonezilla storage server and the template previously created and populated.

Starting a Server with a Clonezilla image configured via an "Install-from-Template" Clonezilla template automatically "transfers" the system to Clonezilla for the initial OS installation (PXE/Network-boot). While the automatic OS installation is running on behalf of Clonezilla openQRM takes back the control of the system and prepares it for local-booting. After rebooting from the automatic installation the system is fully integrated into openQRM.

Hint:

The "Install-from-Template" mechanism with Clonezilla can be also used in the openQRM Cloud! Just make sure the image masters provided in the openQRM Cloud are configured with "Install-from-Template".

Recipe: Local deployment with FAI (automatic Linux installations)

Install and setup openQRM on a physical system or on a virtual machine

1. Enable the following plugins: dns(optional), dhcpd, tftpd, local-server, fai
2. Install a FAI install server on a dedicated system (physical server or VM)

Notice:

Configure the FAI install service to not run the "dhcpd" service on the FAI system itself. The openQRM "dhcpd" service provided by the Dhcpd-Plugin is used instead.

Additionally install the "screen" package on the FAI server.

Integrate a FAI install server into openQRM via the "local-server" plugin

Please check "Integrating existing, local-installed systems with Local-Server"

3. Create a new storage server from the type "fai-deployment" using the FAI systems resource

Add FAI "snippets" (preseed-templates) to the FAI install server and combine them to installation profiles.

Hint:

Add the FAI snippet `openqrm_client_fai_auto_install.snippets` from the FAI plugin help section to your FAI profiles to automatically install the openQRM client on the provisioned systems

VM images for local-deployment (e.g. Citrix (localboot), KVM (localboot), Xen (localboot)) can now be set to "install-from-template" via FAI in the image "Edit" section.

Local-deployment images for physical system are created through the FAI plugins "Image Admin" section. In the image "Edit" section they can be set to "install-from-template" via FAI in the same way as virtual machines.

Starting a Server with a Local-deployment Image configured via an "Install-from-Template" FAI installation profile automatically "transfers" the system to the FAI Server for the initial OS installation (PXE/Network-boot). While the automatic OS installation is running on behalf of the FAI servers responsibility openQRM takes back the control of the system and prepares it for local-booting.

After rebooting from the automatic installation the system is fully integrated into openQRM.

Hint:

The "Install-from-Template" mechanism with FAI can be also used in the openQRM Cloud! Just make sure the image masters provided in the openQRM Cloud are configured with "Install-from-Template".

Recipe: Local deployment with LinuxCOE (automatic Linux installations)

Install and setup openQRM on a physical system or on a virtual machine

1. Enable the following plugins: dns(optional), dhcpd, tftpd, linuxcoe

Notice:

In opposite to Cobbler, FAI and Opsi the LinuxCOE automatic install server is automatically provided by the LinuxCOE plugin. After enabling and starting the LinuxCOE plugin the LinuxCOE automatic install server is automatically configured and its Web UI embedded into the openQRM LinuxCOE plugin.

There is no need for a dedicated system for the install server.

2. Create a new storage server from the type "LinuxCOE-deployment" using the openQRM server systems resource
3. Create LinuxCOE automatic installation profiles via the LinuxCOE Web UI (LinuxCOE plugin – create templates)

Provide a description for each created template via the LinuxCOE template manager

VM images for local-deployment (e.g. Citrix (localboot), KVM (localboot), Xen (localboot)) can now be set to "install-from-template" via LinuxCOE in the Image "Edit" section.

Local-deployment images for physical system are created through the LinuxCOE plugins "Image Manager" section. In the image "Edit" section they can be set to "install-from-template" via LinuxCOE in the same way as virtual machines.

Starting a Server with a Local-deployment image configured via an "Install-from-Template" LinuxCOE installation profile automatically "transfers" the system to the LinuxCOE server for the initial OS installation (PXE/Network-boot). While the automatic OS installation is running on behalf of the LinuxCOE servers responsibility openQRM takes back the control of the system and prepares it for local-booting. After rebooting from the automatic installation the system is fully integrated into openQRM.

Hint:

The "Install-from-Template" mechanism with LinuxCOE can be also used in the openQRM Cloud! Just make sure the image masters provided in the openQRM Cloud are configured with "Install-from-Template".

Hint:

After creating an installation template the resulting ISO image can be burned on an CD to automatically install a physical server (the initial goal of the LinuxCOE Project).

Hint:

In openQRM the LinuxCOE ISO images are also automatically available on virtualization host from the type "local-deployment VMs" (e.g. "KVM (localboot)" and "Xen (localboot)") in the /linuxcoe-iso directory. Simply configure a virtual machine to boot from such a LinuxCOE ISO image for an fully automatic VM installation.

Please notice that after a successful installation the VM will most likely try to boot from the ISO image again after the automatic install procedure finished! Please stop the VMs Server after the initial automatic installation, then re-configure the virtual machine to boot from "local" and start the Server again.

Recipe: Local deployment with Opsi (automatic Windows installations)

Install and setup openQRM on a physical system or on a virtual machine

1. Enable the following plugins: dns(optional), dhcpd, tftpd, local-server, opsi
2. Install a Opsi install server on a dedicated system (physical server or VM)

Notice:

Configure the Opsi install service to not run the "dhcpd" service on the Opsi system itself. The openQRM "dhcpd" service provided by the Dhcpd-Plugin will be used instead.

Additionally install the "screen" package on the Opsi server.

Integrate a Opsi install server into openQRM via the "local-server" plugin

Please check "Integrating existing, local-installed systems with Local-Server"

3. Create a new storage server from the type "opsi-deployment" using the Opsi systems resource

Create and configure automatic Windows installations and optional Windows application packages via the Opsi Web UI.

VM images for local-deployment (e.g. Citrix (localboot), KVM (localboot), Xen (localboot)) can now be set to "install-from-template" via Opsi in the image "Edit" section.

Local-deployment images for physical system are created through the Opsi plugins "Image Manager" section. In the image "Edit" section they can be set to "install-from-template" via Opsi in the same way as virtual machines.

Starting a Server with a Local-deployment image configured via an "Install-from-Template" Opsi installation Profile automatically "transfers" the system to the Opsi server for the initial OS installation (PXE/Network-boot). While the automatic OS installation is running on behalf of the Opsi servers responsibility openQRM takes back the control of the system and prepares it for local-booting.

After rebooting from the automatic installation the system is fully integrated into openQRM via the openQRM client for Windows.

Hint:

The "Install-from-Template" mechanism with Opsi can be also used in the openQRM Cloud! Just make sure the image masters provided in the openQRM Cloud are configured with "Install-from-Template".

Network Deployment - DHCPD/TFTPD/Network-Storage

Network-deployment in openQRM is a combination of:

- a network-booted physical server

- or

- a virtualization plugin (for network-deployment)

- and a storage plugin (also for network-deployment).

In case of network-deployment of a physical server the physical system itself is available as “idle” resource after network-booting it via PXE.

In case of virtual machine network-deployment the virtualization plugin (e.g. Citrix, KVM, VMware-ESX, Xen) provides JUST the “Resource” object of a Server!

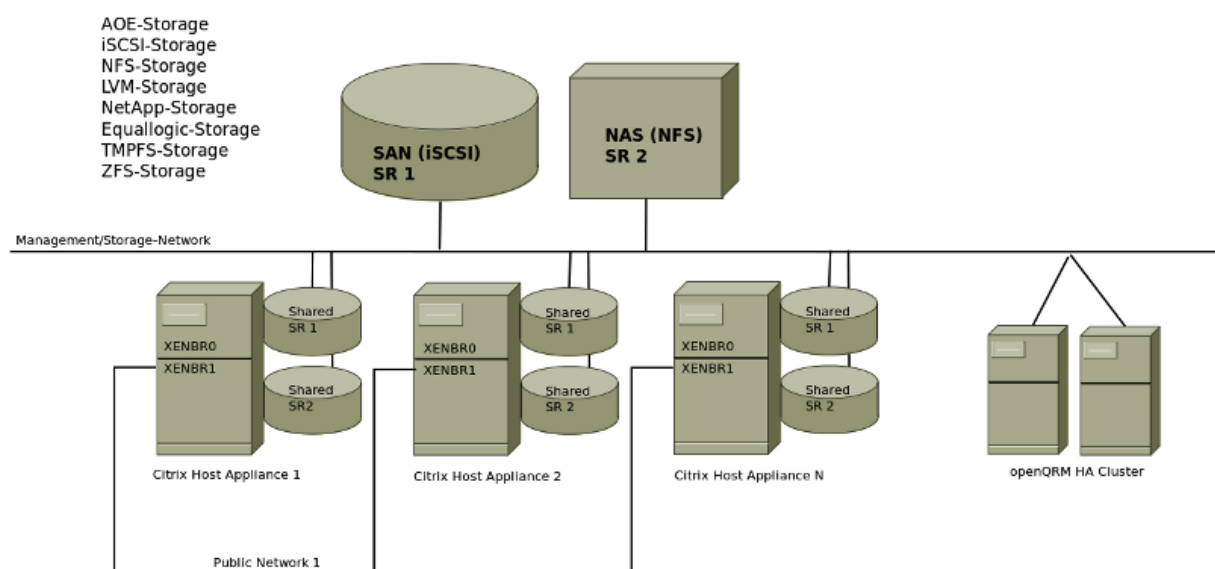
The storage plugin (e.g. Aoe-Storage, iSCSI-Storage, NFS-Storage, LVM-Storage, Tempfs-Storage etc.) provides JUST the “Image” (and its Storage”) object of a Server!

Summary of Network-Deployment

A physical network-booted system can be deployed with ANY storage plugin (for network-deployment)

A network-booted virtual machine provided by a virtualization plugin (for network-deployment) can be deployed with ANY storage plugin (for network-deployment)

Recipe: Network VM deployment with Citrix (networkboot)



Install and setup openQRM on a physical system or on a virtual machine

1. Install the Citrix “xe” command line utility on the openQRM server system to /usr/bin/xel

(e.g. use "scp" to copy it from your Citrix XenServer system)

Enable the following plugins: dns (optional), dhcpd, tftpd, local-server, novnc, citrix

2. Configure the Citrix Boot-service

Please check the Citrix help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration

CITRIX_MANAGEMENT_INTERFACE

CITRIX_EXTERNAL_INTERFACE

CITRIX_DEFAULT_VM_TEMPLATE

In case you are using openQRM Cloud with Xen (network-deployment) please also set

CITRIX_DEFAULT_SR_UUID

3. Install one (or more) Citrix XenServer within the openQRM management network

Hint:

Enable SSH login on the Citrix XenServer

The openQRM NoVNC plugin provides access to the Citrix VM console (requires SSH Login enabled on the Citrix XenServer configuration). Use the Citrix discovery to discover Citrix XenServer system(s). After the discovery please integrate the Citrix system(s) by providing the administrative credentials.

Please notice:

Integrating the Citrix XenServer system automatically creates and pre-configures a Citrix host Server in open-QRM!

4. Create a "Citrix" virtual machine
5. Use the "Server Create" wizard to create a new Server object
6. Create a new Citrix VM resource in the 2. step

- Select the previously created "Citrix" host Server object for creating the new Citrix VM.

The new created Citrix VM automatically performs a network-boot and is available as "idle" Citrix VM resource soon. Then select the new created resource for the Server.

- Creating an image for network-deployment

In the 3. step of the Server wizard please create a new Image using one of the storage plugins for network-deployment. E.g. AOE-Storage, iSCSI-Storage, NFS-Storage, LVM-Storage, Netapp-Storage, TMPFS-Storage, ZFS-Storage

Hint:

The creation of the image for network-deployment depends on the specific storage plugin being used. Please refer to the section about the storage plugin you have chosen for network-deployment in this document.

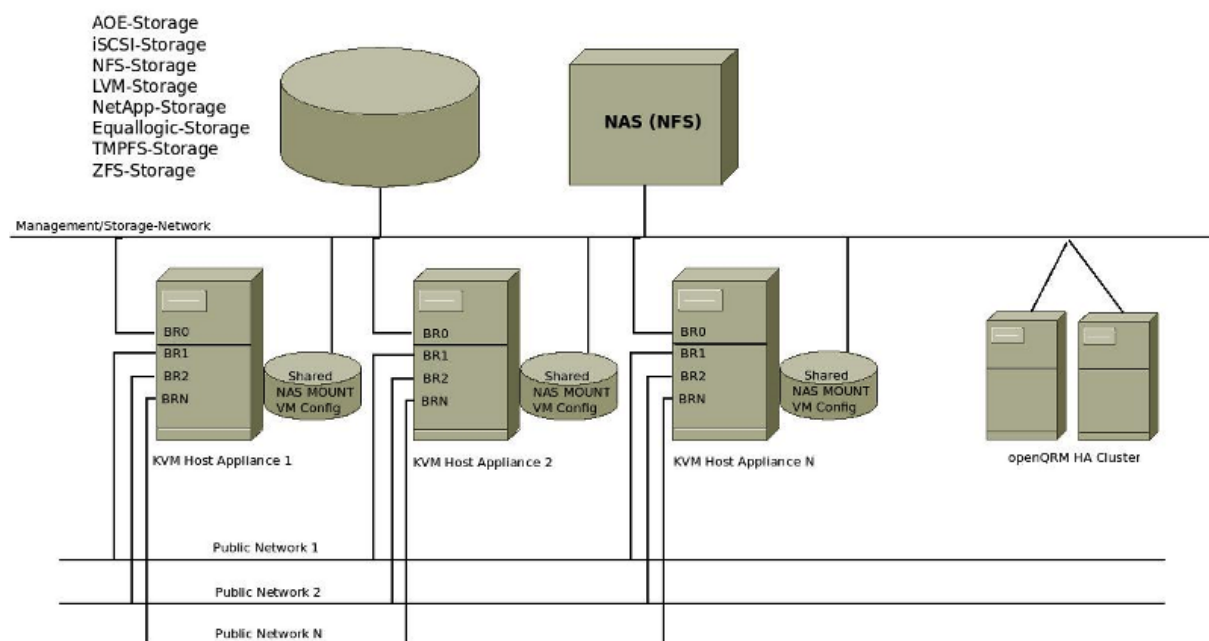
After creating the new network-deployment image please select it in the Server wizard.

- Save the Server
- Start the Server

Starting the Server automatically combines the Citrix VM (resource object) and the image object (volume) abstracted via the specific storage plugin. (stopping the Server will “uncouple” resource and image)

To enable further management functionalities of openQRM “within” the virtual machines operating system the “openqrm-client” is automatically installed in the VM during the deployment phase. There is no need to further integrate it by e.g. the “Local-Server” plugin!

Recipe: Network VM deployment with KVM (networkboot)



1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns (optional), dhcpd, tftpd, local-server, novnc, device-manager, network-manager, kvm
3. Configure the KVM Boot-service

Please check the KVM help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration. This network-configuration is only used by the openQRM Cloud to auto-create VMs.

OPENQRM_PLUGIN_KVM_BRIDGE_NET1

OPENQRM_PLUGIN_KVM_BRIDGE_NET2

OPENQRM_PLUGIN_KVM_BRIDGE_NET3

OPENQRM_PLUGIN_KVM_BRIDGE_NET4

OPENQRM_PLUGIN_KVM_BRIDGE_NET5

4. Select a physical server as the virtualization host
(needs VT/virtualization support available and enabled in the system BIOS)

Please notice:

This system dedicated to be the virtualization host can be the openQRM server system itself or it can be a "remote" physical server integrated via the "Local-Server Plugin".

Please check "Integrating existing, local-installed systems with Local-Server"

Please make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!
- One or more bridges configured for the virtual machines (e.g. br0, br1, etc.)

1. Create a "KVM" Server (Server object for the virtualization host)

In case the openQRM server is the physical system dedicated to be the virtualization host please use the "Server Create" wizard to create a new Server object using the openQRM server resource.

2. After creating the Server edit it and set "Virtualization" to "KVM Host"

In case a "remote" physical system, integrated via the "Local-Server" plugin, is the virtualization host the system integration already created a Server for this system. Simply edit it and set "Virtualization" to "KVM Host"

3. Create a "KVM" virtual machine

- Use the "Server Create" wizard to create a new Server object

- Create a new KVM VM resource in the 2. step
- Select the previously created “KVM” Server object for creating the new KVM VM.

The new created KVM VM automatically performs a network-boot and is available as “idle” KVM VM resource soon. Then select the new created resource for the Server.

- Creating an image for network-deployment

In the 3. step of the Server wizard please create a new image using one of the storage plugins for network-deployment. E.g. AOE-Storage, iSCSI-Storage, NFS-Storage, LVM-Storage, Netapp-Storage, TMPFS-Storage, ZFS-Storage

Hint:

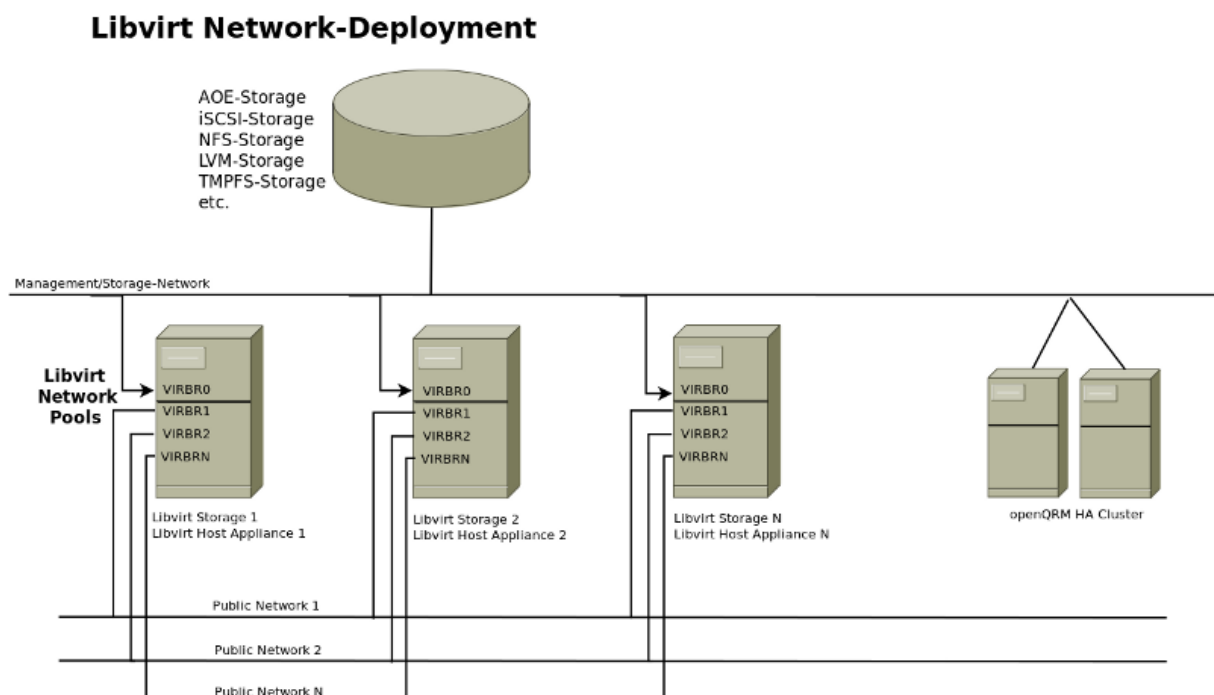
The creation of the image for network-deployment depends on the specific storage plugin being used. Please refer to the section about the storage plugin you have chosen for network-deployment in this document.

- After creating the new network-deployment Image please select it in the Server wizard.
- Save the Server
- Start the Server

Starting the Server automatically combines the KVM VM (resource object) and the image object (volume) abstracted via the specific storage plugin. Stopping the Server will “uncouple” resource and image.

To enable further management functionalities of openQRM “within” the virtual machines operating system the “openqrm-client” is automatically installed in the VM during the deployment phase. There is no need to further integrate it by e.g. the “Local-Server” plugin.

Recipe: Network VM deployment with Libvirt (networkboot)



1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns (optional), dhcpd, tftpd, local-server, novnc, libvirt
3. Configure the Libvirt Boot-service

Please check the Libvirt help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration. This network-configuration is only used by the openQRM Cloud to auto-create VMs.

```
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET1
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET2
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET3
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET4
OPENQRM_PLUGIN_LIBVIRT_BRIDGE_NET5
OPENQRM_PLUGIN_LIBVIRT_PRIMARY_NIC_TYPE
OPENQRM_PLUGIN_LIBVIRT_ADDITIONAL_NIC_TYPE
OPENQRM_PLUGIN_LIBVIRT_VM_DEFAULT_VNCPASSWORD
```

Select a physical server as the virtualization host

(needs VT/virtualization support available and enabled in the system BIOS)

Please notice:

This system dedicated to be the virtualization host can be the openQRM server system itself or it can be a "remote" physical server integrated via the "Local-Server Plugin".

Please check "Integrating existing, local-installed systems with Local-Server"

Please make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!
- One or more bridges configured for the virtual machines (e.g. br0, br1, etc.)

Create a "Libvirt" Server (Server object for the virtualization host)

In case the openQRM server is the physical system dedicated to be the virtualization host please use the "Server Create" wizard to create a new Server object using the openQRM server resource.

After creating the Server edit it and set "Virtualization" to "Libvirt Host"

In case a "remote" physical system, integrated via the "Local-Server" plugin, is the virtualization host the system integration already created a Server for this system. Simply edit it and set "Virtualization" to "Libvirt Host"

Create a "Libvirt" (networkboot) virtual machine

- Use the “Server Create” wizard to create a new Server object
- Create a new Libvirt VM resource in the 2. step
- Select the previously created “Libvirt” Server object for creating the new Libvirt VM.

The new created Libvirt VM automatically performs a network-boot and is available as “idle” Libvirt VM resource soon. Then select the new created resource for the Server.

- Creating an image for network-deployment

In the 3. step of the Server wizard please create a new image using one of the storage plugins for network-deployment. E.g. AOE-Storage, iSCSI-Storage, NFS-Storage, LVM-Storage, Netapp-Storage, TMPFS-Storage, ZFS-Storage

Hint:

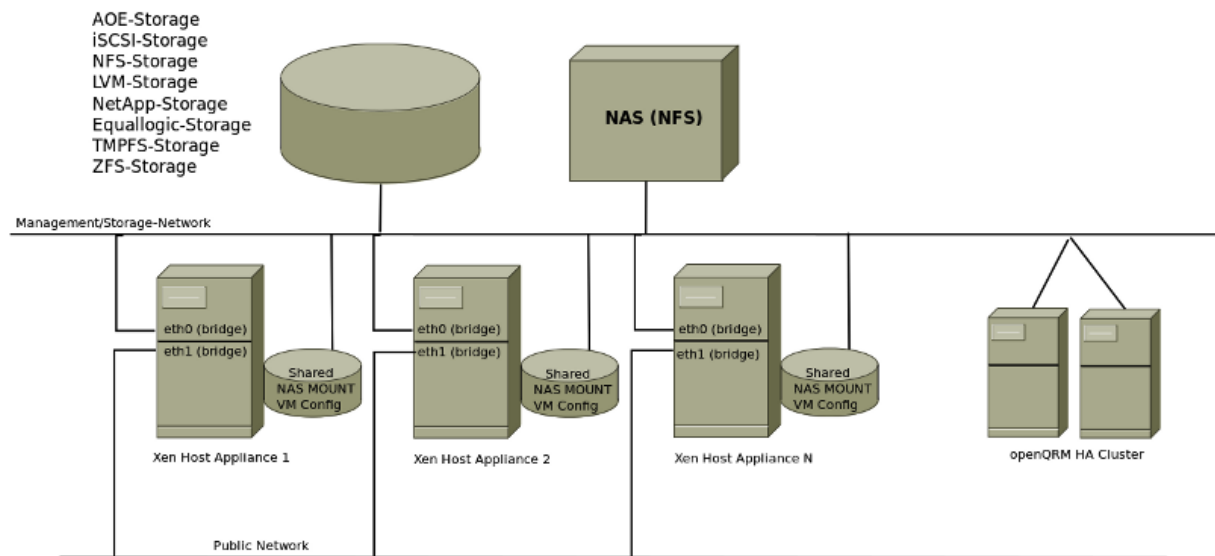
The creation of the image for network-deployment depends on the specific storage plugin being used. Please refer to the section about the storage plugin you have chosen for network-deployment in this document.

- After creating the new network-deployment Image please select it in the Server wizard.
- Save the Server
- Start the Server

Starting the Server automatically combines the Libvirt VM (resource object) and the image object (volume) abstracted via the specific storage plugin. Stopping the Server will “uncouple” resource and image.

To enable further management functionalities of openQRM “within” the virtual machines operating system the “openqrm-client” is automatically installed in the VM during the deployment phase. There is no need to further integrate it by e.g. the “Local-Server” plugin.

Recipe: Network VM deployment with Xen (networkboot)



1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns (optional), dhcpd, tftpd, local-server, novnc, device-manager, network-manager, xen
3. Configure the Xen Boot-service

Please check the Xen help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration

OPENQRM_PLUGIN_XEN_INTERNAL_BRIDGE

OPENQRM_PLUGIN_XEN_EXTERNAL_BRIDGE

4. Select a physical server as the virtualization host (needs VT/virtualization support available and enabled in the system BIOS)

Please notice:

This system dedicated to be the virtualization host can be the openQRM server system itself or it can be a "remote" physical server integrated via the "Local-Server Plugin".

Please check "Integrating existing, local-installed Systems with Local-Server"

Please make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!
- One or more bridges configured for the virtual machines (e.g. br0, br1, etc.)

1. Create a “Xen” Server (Server object for the virtualization host)

In case the openQRM server is the physical system dedicated to be the virtualization host please use the “Server Create” wizard to create a new Server object using the openQRM server resource.

2. After creating the Server edit it and set “Virtualization” to “Xen Host”

In case a “remote” physical system, integrated via the “Local-Server” plugin, is the virtualization host the system integration already created a Server for this system. Simply edit it and set “Virtualization” to “Xen Host”

3. Create a “Xen” virtual machine

- Use the “Server Create” wizard to create a new Server object
- Create a new Xen VM resource in the 2. step
- Select the previously created “Xen” Server object for creating the new Xen VM.
- The new created Xen VM automatically performs a network-boot and is available as “idle” Xen VM resource soon. Then select the new created resource for the Server.
- Creating an image for network-deployment

In the 3. step of the Server wizard please create a new image using one of the storage plugins for network-deployment. E.g. AOE-Storage, iSCSI-Storage, NFS-Storage, LVM-Storage, Netapp-Storage, TMPFS-Storage, ZFS-Storage

Hint:

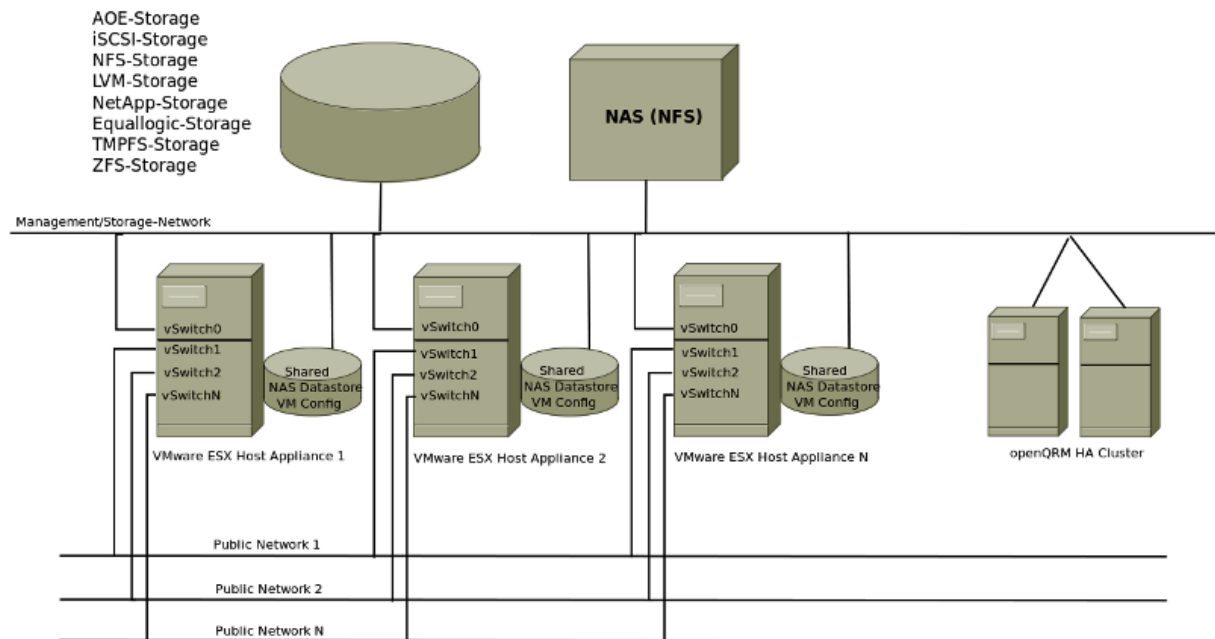
The creation of the image for network-deployment depends on the specific storage plugin being used. Please refer to the section about the storage plugin you have chosen for network-deployment in this document.

- After creating the new network-deployment Image please select it in the Server wizard.
- Save the Server
- Start the Server

Starting the Server automatically combines the Xen VM (resource object) and the image object (volume) abstracted via the specific storage plugin. Stopping the Server will “uncouple” resource and image.

To enable further management functionalities of openQRM “within” the virtual machines operating system the “openqrm-client” is automatically installed in the VM during the deployment phase. There is no need to further integrate it by e.g. the “Local-Server” plugin!

Recipe: Network VM deployment with VMware ESX (networkboot)



1. Install and setup openQRM on a physical system or on a virtual machine
2. Install the latest VMware Vsphere SDK on the openQRM server system
3. Enable the following plugins: dns (optional), dhcpd, tftpd, local-server, novnc, vmware.esx
4. Configure the VMware ESX Boot-service

Please check the VMware ESX help section how to use the Plugin-Manager "configure" option or the "openqrm" utility to apply a custom boot-service configuration

Adapt the following configuration parameters according to your virtualization host bridge configuration

OPENQRM_VMWARE_ESX_INTERNAL_BRIDGE

OPENQRM_VMWARE_ESX_EXTERNAL_BRIDGE_2

OPENQRM_VMWARE_ESX_EXTERNAL_BRIDGE_3

OPENQRM_VMWARE_ESX_EXTERNAL_BRIDGE_4

OPENQRM_VMWARE_ESX_EXTERNAL_BRIDGE_5

In case you are using openQRM Cloud with Xen (network-deployment) please also set

OPENQRM_VMWARE_ESX_CLOUD_DATASTORE

OPENQRM_VMWARE_ESX_GUEST_ID

OPENQRM_VMWARE_ESX_CLOUD_DEFAULT_VM_TYPE

5. Install one (or more) VMware ESX server within the openQRM management network
6. Use the VMware ESX Discovery to discover VMware-ESX system(s)

After the discovery please integrate the VMware-ESX system(s) by providing the administrative credentials.

Please notice:

Integrating the VMware ESX system automatically creates and pre-configures a VMware ESX host Server in openQRM.

7. Create a “VMware ESX” virtual machine

- Use the “Server Create” wizard to create a new Server object
- Create a new VMware ESX VM resource in the 2. step
- Select the previously created “VMware ESX” Server object for creating the new VMware ESX VM.

The new created VMware ESX VM automatically performs a network-boot and is available as “idle” VMware ESX VM Resource soon. Then select the new created resource for the Server.

- Creating an image for network-deployment

In the 3. step of the Server wizard please create a new image using one of the storage plugins for network-deployment. E.g. AOE-Storage, iSCSI-Storage, NFS-Storage, LVM-Storage, Netapp-Storage, TMPFS-Storage, ZFS-Storage

Hint:

The creation of the image for network-deployment depends on the specific storage plugin being used. Please refer to the section about the storage plugin you have chosen for network-deployment in this document.

- After creating the new network-deployment Image please select it in the Server wizard.
- Save the Server
- Start the Server

Starting the Server automatically combines the VMware ESX VM (resource object) and the image object (volume) abstracted via the specific storage plugin. Stopping the Server will “uncouple” resource and image.

To enable further management functionalities of openQRM “within” the virtual machines operating system the “openqrm-client” is automatically installed in the VM during the deployment phase. There is no need to further integrate it by e.g. the “Local-Server” plugin.

Recipe: Network Deployment with AOE-Storage

1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns (optional), dhcpd, tftpd, aoe-storage
3. Select a physical server or virtual machine for the AOE storage

Please notice:

This system dedicated to be the AOE storage can be the openQRM server system itself or it can be a “remote” physical server integrated via the “Local-Server Plugin”.

Please check “Integrating existing, local-installed systems with Local-Server”

Please make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!

1. Create a AOE storage object in openQRM

- go to Base – Components – Storage – Create
- provide a name for the storage object
- select “aoe-deployment” as deployment type
- select the resource dedicated for the AOE storage

1. Create a new AOE storage volume

- goto Base – Components – Storage – List, click on “manage”
- create a new AOE volume
Creating a new volume automatically creates a new image object in openQRM

The AOE volumes are physically created on the storage at /aoe-storage/ as blockfiles and are added to /etc/vblade.conf automatically.

Please notice:

The created volume (abstracted as “Image” in openQRM) can now be used to create an Server and deploy either a (network-booted) physical system or a (network-booted) virtual machine provided by one of the following virtualization plugins: Citrix, KVM, VMware ESX, Xen

Hint:

To populate new created AOE storage volumes (images) it is recommended to use the “Install-from-NFS” method. To configure “Install-from-NFS” edit the Image and set “Install-from-NFS” to one of the NFS- or LVM-NFS based Images.

Please check “Network Deployment with NFS-Storage” or “Network Deployment with LVM-Storage”

During initial startup the new created, empty AOE storage image is then populated from the NFS- or LVM-NFS based image selected for “Install-from-NFS”.

Recipe: Network deployment with iSCSI-Storage

1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns (optional), dhcpd, tftpd, iscsi-storage
3. Select a physical server or virtual machine for the iSCSI storage

Please notice:

This System dedicated to be the iSCSI storage can be the openQRM server system itself or it can be a "remote" physical server integrated via the "Local-Server Plugin".

Please check "Integrating existing, local-installed systems with Local-Server"

Please make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!

1. Create a iSCSI storage object in openQRM

- go to Base – Components – Storage – Create
- provide a name for the storage object
- select "iscsi-deployment" as deployment type
- select the resource dedicated for the iSCSI storage

1. Create a new iSCSI storage volume

- goto Base – Components – Storage – List, click on "manage"
- create a new iSCSI volume
Creating a new volume automatically creates a new image object in openQRM

The iSCSI volumes are physically created on the storage at /target/ as blockfiles and are added to /etc/iet/ietd.conf automatically.

Please notice:

The created volume (abstracted as "Image" in openQRM) can now be used to create an Server and deploy either a (network-booted) physical system or a (network-booted) virtual machine provided by one of the following virtualization plugins: Citrix, KVM, VMware ESX, Xen

Hint:

To populate new created iSCSI storage volumes (images) it is recommended to use the "Install-from-NFS" method. To configure "Install-from-NFS" edit the image and set "Install-from-NFS" to one of the NFS- or LVM-NFS based images.

Please check "Network Deployment with NFS-Storage" or "Network Deployment with LVM-Storage"

During initial startup the new created, empty iSCSI storage image is then populated from the NFS- or LVM-NFS based image selected for "Install-from-NFS".

Recipe: Network deployment with NFS-Storage

1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns (optional), dhcpd, tftpd, nfs-storage
3. Select a physical server or virtual machine for the NFS storage

Please notice:

This system dedicated to be the NFS storage can be the openQRM server system itself or it can be a “remote” physical server integrated via the “Local-Server Plugin”.

Please check “Integrating existing, local-installed Systems with Local-Server”

Please make sure this system meets the following requirements:

Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!

4. Create a NFS storage object in openQRM
 - goto Base – Components – Storage – Create
 - provide a name for the storage object
 - select “nfs-deployment” as deployment type
 - select the resource dedicated for the NFS Storage

1. Create a new NFS storage volume

- goto Base – Components – Storage – List, click on “manage”
 - create a new NFS volume
- Creating a new volume automatically creates a new image object in openQRM

The NFS volumes are physically created on the storage at `/exports/` as directories and are added to `/etc/exports` automatically.

Please notice:

The created volume (abstracted as “Image” in openQRM) can now be used to create an Server and deploy either a (network-booted) physical system or a (network-booted) virtual machine provided by one of the following virtualization plugins: Citrix, KVM, VMware ESX, Xen

Hint:

To populate new created NFS storage volumes (images) it is recommended to use the “Image-Shelf” method provided by the Image-Shelf plugin. Please check “Populating Network Deployment Images via the Image-Shelf”

Recipe: Network deployment with LVM-Storage

Hint:

The “lvm-storage” plugin combines the AOE (AOE-Storage Plugin), iSCSI (iSCSI-Storage Plugin) and the NFS (NFS-Storage Plugin) Network-Storage technologies with an underlaying “LVM” (Logical Volume Management via LVM2).

It provides 3 different deployment types:

- lvm-aoe-deployment
- lvm-iscsi-deployment
- lvm-nfs-deployment

1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns(optional), dhcpd, tftpd, lvm-storage
3. Select a physical server or virtual machine for the LVM storage

Please notice:

This system dedicated to be the LVM storage can be the openQRM server system itself or it can be a “remote” physical server integrated via the “Local-Server Plugin”.

Please check “Integrating existing, local-installed systems with Local-Server”

Please make sure this system meets the following requirements:

- Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!
- One (or more) lvm volume group(s) with free space dedicated for the LVM storage

1. Create a LVM storage object in openQRM

- goto Base – Components – Storage – Create
- provide a name for the storage object
- select “nfs-deployment” as deployment type
- select the resource dedicated for the LVM storage

1. Create a new LVM storage volume

- goto Base – Components – Storage – List, click on “manage”
- create a new LVM volume

Creating a new Volume automatically creates a new image object in openQRM

The LVM volumes are physically created on the storage as logical volumes within the selected volume group. Depending of the deployment type the logical volume device is added to the AOE/NFS/iSCSI configuration automatically.

Please notice:

The created volume (abstracted as "Image" in openQRM) can now be used to create an Server and deploy either a (network-booted) physical system or a (network-booted) virtual machine provided by one of the following virtualization plugins: Citrix, KVM, VMware ESX, Xen

Hint:

To populate new created LVM NFS storage volumes (images) it is recommended to use the "Image-Shelf" method provided by the Image-Shelf plugin.

Please check "Populating Network Deployment Images via the Image-Shelf".

Hint:

To populate new created LVM AOE or LVM iSCSI storage volumes (images) it is recommended to use the "Install-from-NFS" Method. To configure "Install-from-NFS" edit the image and set "Install-from-NFS" to one of the NFS- or LVM-NFS based images.

Please check "Network Deployment with NFS-Storage" or "Network Deployment with LVM-Storage"

Recipe: Network deployment with SAN-Boot-Storage

The "Sanboot-Storage" plugin integrates with gPXE and supports booting and deploying Windows systems directly from SAN storage (iSCSI or AOE) without using any local disk.

1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns (optional), dhcpd, tftpd, sanboot-storage
3. Select a physical server or virtual machine for the SAN-Boot storage

Please make sure this system meets the following requirements:

Enable automatic packages installation capabilities so that it can fetch package dependencies. All eventual required package dependencies are resolved automatically during the first action running!

One (or more) lvm volume group(s) with free space dedicated for the SAN-Boot-Storage

4. Initial Windows installation on the SAN volume
 - Create a SAN-Boot-Storage storage AOE/iSCSI (please check the requirements)
 - Use the SAN-Boot-Storage volume manager to create a new SAN-Boot-Storage volume. Creating a new volume automatically creates a new image object in openQRM
 - Boot one (or more) physical systems via the network (PXE)
 - Set the boot-order of the physical system in its BIOS to

- o Network-boot
- o Boot-from-local-DVD/CD
- Insert a Windows installation media in the physical systems DVD/CD
- Create a new Server using the (idle) physical system as resource and the previously created SAN-Boot-Storage image as the image
- Start the Server

The system now reboots assigned to gPXE network-boot. In the (network-) bootloader stage it connects the SAN-Boot-Storage volume (either per iSCSI or AOE). It will fail to boot from the connected SAN volume since it is still empty. It will move on to the next boot device (the CD/DVD) and start the Windows installation.

You can now directly install into the SAN volume.

For a more detailed documentation about how to setup Windows to directly boot from a SAN storage please refer to the Etherboot Wiki - SAN Boot and SAN Install.

Hint:

It is recommended to install the "Windows openQRM Client" after the initial installation of the SAN volume. The installation media can now be removed from the physical system (also the local disk can be removed).

Hint:

After the initial installation of the Windows operating system on a SAN volume is it recommended to use this as a "Master Template" and to not deploy it any more but just "snapshots" or "clones" of it. You can easily create "snapshots/clones" of the SAN-Bootboot-Storage volumes via the volume manager.

1. Deployment with SAN-Boot-Storage

The created volume (abstracted as "Image" in openQRM) can now be used to create an Server and deploy either a (network-booted) physical system

- Create an Server using a physical, network-booted "idle" resource, a SAN-Boot-Storage image and kernel
- Start the Server

Hint:

Windows deployment to physical systems (and VMs) via the openQRM Cloud is fully supported for SAN-Boot-Storage.

Recipe: Network Deployment with TmpFs-Storage

The "TmpFs-Storage" plugin provisions physical systems and virtual machines "in-Memory" (tmpfs). This is a very useful deployment method to e.g. provision several virtualization hosts systems with a small, firmware-like OS image provided from a single read-only master template.

1. Install and setup openQRM on a physical system or on a virtual machine
2. Enable the following plugins: dns(optional), dhcpd, tftpd, tmpfs-storage

3. Create a Tmpfs-Storage object in openQRM

- goto Base – Components – Storage – Create
- provide a name for the storage object
- select “tmpfs-deployment” as deployment type
- select the openQRM server as the resource dedicated for the Tmpfs-Storage

1. Create a new Tmpfs-Storage volume with the Tmpfs-Storage “Volume Admin”

- Provide the size of memory (RAM) dedicated for the operating system

1. Creating a new volume automatically creates a new image object in openQRM

- Edit the image object of the Tmpfs-Storage volume
 - o goto Base – Components – Image – List, click on “edit”
 - o set “Install from NAS/NFS” to an existing NFS-Storage or LVM-NFS-Storage image
- Create an Server using a network-booted physical system or a network-boot virtual machine as resource, the previously created Tmpfs-Storage image and a kernel
- Start the Server

The provisioned system now reboots, reserves the selected amount of memory for the OS deployment, installs itself from the “install-from-nfs” Image and boots up the OS in memory.

Hint:

Tmpfs Deployment to physical systems (and VMs) via the openQRM Cloud is fully supported

Recipe: Populating network deployment images via the Image-Shelf

The image-shelf plugin provides an easy way to populate new network-deployment images from operating system templates. The operating system templates are simply tar.gz files containing a Linux root-filesystem (with an empty /proc and /sys dir). The image-shelf plugin comes with a pre-configured Image-shelf sponsored by openQRM Enterprise. Custom Image-shelves can be added using various different protocols (local/FTP/HTTP/HTTPS/NFS). The Image-shelf content is configured by a configuration file “image-shelf.conf” located in the same directory as the templates. (here an example self-explaining image-shelf.conf)

Please notice:

The Image-Shelves templates can be “only” used to populate Images from type “lvm-nfs-deployment” (“LVM-Storage” plugin) and “nfs-deployment” (“NFS-Storage” plugin). From those NFS-based storage types images can be easily transferred to e.g. iSCSI- or AOE-deployment images via the INSTALL_FROM deployment parameters.

1. Create a new NFS- or LVM-NFS image
2. Please check “Network Deployment with NFS-Storage” and “Network Deployment with LVM-Storage”
3. Select an Image-Shelf from the list
4. Select an Server-Template from the list
5. Select an (empty) NFS- or LVM-NFS image to deploy the template

Recipe: Creating new kernel for network deployment

New kernels can be added on the openQRM server with the following command:

```
/usr/share/openqrm/bin/openqrm kernel add -n name -v version -u username -p password [-l location] [-i initramfs/ext2] [-t path-to-initrd-template-file]
```

name can be any identifier as long as it has no spaces or other special characters; it is used as part of the filename.

version should be the version for the kernel you want to install. If the filenames are called vmlinuz-2.6.26-2-amd64 then 2.6.26-2-amd64 is the version of this kernel.

username and password are the credentials to openqrm itself.

location is the root directory for the kernel you want to install. The files that are used are

```
${location}/boot/vmlinuz-${version}, ${location}/boot/initrd.img-${version}
```

and

```
${location}/lib/modules/${version}/*
```

initramfs/ext2 should specify the type of initrd image you want to generate. Most people want to use initramfs here.

path-to-initrd-template-file should point to an openqrm initrd template. These can be found in the openqrm base dir under etc/templates.

Example:

```
/usr/share/openqrm/bin/openqrm kernel add -n openqrm-kernel-1 -v 2.6.29 -u openqrm -p openqrm -i initramfs -l / -t /usr/share/openqrm/etc/templates/openqrm-initrd-template.debian.x86_64.tgz
```


High availability

High availability for Servers (infrastructure level) – built-in

High availability for Servers deployed by openQRM is a built-in feature. Even more openQRMs architecture is designed for high availability on the infrastructure level.

In openQRM the actual “service” is represented by an Server that simply can be marked as “high available” in the high availability plugins Server configuration.

In case the resource of the Server gets into an error state openQRM triggers a “high availability hook” which is implemented by the high availability plugin. The responsibility to fail-over and restart the “service” (the Server) is then transferred to the plugin.

The high availability plugin provides a “N-to-1 fails-over” by stopping the Server, finding a new resource fitting to the Server SLA parameters, updating the Server with the new resource and starting the Server again. All Server stop- and start hooks will be executed in case of a fail-over.

Please notice:

“N-to-1 Fail-over” means that only a single “hot stand-by” is needed for a group of high available Servers. The administrator just needs to take care that the HA pool of “idle” resources fits to the specific required HA needs.

Please notice:

High availability for Servers is also supported in the openQRM Cloud! Simply created a “high availability” Cloud product and enable HA in the Cloud configuration.

Hint:

The high availability plugin even provides a transparent fail-over from e.g. physical systems to virtual machines. This can be configured in the Server by setting the virtualization type to the type of the resource dedicated as “hot standbys”

Hint:

For virtual machines openQRM automatically creates new VM resources for fail-over. Active “hot standby” resources are therefore not needed for high availability for VMs. One (or more) active virtualization host Servers are sufficient for VM HA.

Hint:

The “high availability” plugin is integrated with the “Out-of-Band-Management” plugins (Wake-Up-On-LAN and IPMI plugin) and is able to automatically power-on physical systems configured as virtualization host. Having “Out-of-Band-Management” available and configured openQRM is capable to automatically a physical server in emergency situations to provide a “standby” for Fail-over.

The integration with the IPMI plugin also provides the capability to “fence” a physical systems in case of a fail-over via the “high availability” plugin (STONITH).

High availability for services (application level) - LCMC

For high availability of services (application level) openQRM integrated with the “Linux Cluster Management Console” LCMC. It provides a convenient way to configure high availability (pacemaker/heartbeat/corosync) for applications on Servers via an intuitive GUI. No manual configuration for this plugin is needed.

For detailed information how to use LCMC please refer to the LCMC documentation.

Please notice:

High availability for services is also supported in the openQRM Cloud! If enabled the LCMC plugins provides its functionalities also for the Cloud users.

High availability for openQRM Server

openQRM is designed to make an high availability setup for the openQRM server easy and robust!

The openQRM server consists of standard technologies like:

- Apache webserver plus PHP
- A database backend (MySQL, Postgres)
- A filesystem (by default /usr/share/openqrm)

An high availability Setup for the openQRM server can be archived by:

- Sharing the openQRM filesystem (/usr/share/openqrm) between two or more nodes

Any type of shared/distributed/cluster-filesystem will do the job

- Using a remote, high available database

Any type of HA MySQL or Postgres database will do the job

- Use a global cluster IP address for the openQRM server installation and setup

Setup fail-over of the global cluster IP address "openqrm" service

Any type of HA technology will do the job (e.g. linux-ha, pacemaker, corosync)

Please notice:

Please make sure to have all openQRM run-time dependencies installed on all cluster nodes!

Workflow automation

Automated IP assignment for the openQRM network with DHCPD

The dhcpd plugin automatically manages your IP-address assignment and network-boot environment for the rapid-deployment features of openQRM. Since some of the dynamic deployment methods in openQRM are based on network-booting (PXE) a dhcpd-server is a fundamental service to assign ip-addresses to booting resources. An automatic configured Dhcpd-server is provided by the DHCPD plugin. No manual configuration is needed. The dhcpd plugin configures a dhcpd.conf file during initialization.

The main configuration file (dhcpd.conf) of the dhcpd server provided by this plugin you can find at:

```
/usr/share/openqrm/plugins/dhcpd/etc/dhcpd.conf
```

Automated DNS management for the openQRM network with DNS

The DNS-plugin automatically manages IP-Address to hostname resolving via bind/named for the entire openQRM network. It configures the hostname/ip entries for the DNS database and reloads the name-server during start/stop of an Server. The hostnames are automatically set to the Server name with the ip address of the Server-resource. An automatic configured DNS-server is provided by this plugin.

Automated application deployment with Puppet

The Puppet plugin provides automated configuration-management for Servers in openQRM. It seamlessly integrates Puppet within the openQRM GUI and assists to put specific Servers into pre-made or custom Puppet-classes. By enabling the plugin the Puppet-environment (server and client) is pre-configured and initialized automatically according to best-practice experiences e.g. by keeping the puppet-configuration within a svn-repository.

The Puppet-configuration is organized in "classes", "groups" and "Servers". Custom classes can be added to the class-directory. Classes should be combined to "groups" these will be automatically displayed in the Puppet-plugin.

The Puppet-configuration repository is also available for external svn clients. To check out the Puppet-repo please run

```
svn co svn+ssh://[user]@[openqrm-server-ip]/usr/lib/openqrm/plugins/puppet/etc/puppet/
```

Assigning applications to Servers

- Go to the "Servers" in the Puppet plugin menu
- Select an Server to be configured via puppet
- Select the Puppet-groups the Server should belong to

Please notice:

Automated application deployment for Servers is also supported in the openQRM Cloud! Simply created "Puppet" Cloud products from the Puppet recipes and enable Puppet in the Cloud configuration.

Automated monitoring with Nagios and Icinga

The "Nagios" plugin integrates Nagios into openQRM. It provides a convenient method to both fully automatically map and monitor your complete openQRM management network or to easily manually configure custom service checks per Server.

Automatic configuration

To generate and/or update the Nagios configuration for the openQRM-network and managed servers use the "Config / AutoConfig" link in the Nagios-plugin menu. The nagios-configuration will be created automatically by scanning the network via the "nmap" utility. The output of the nmap run is used by "nmap2nagios-ng" to generate the Nagios-configuration.

Custom Server service check configuration

Use:

Plugins -> Monitoring -> Nagios3 -> Config -> Services

to create new service objects to monitor

Use:

Plugins -> Monitoring -> Nagios3 -> Config -> Servers

to setup custom service checks

Hint:

Since Nagios and Icinga are "configuration file compatible" the Nagios plugin provides an additional menu section for Icinga.

Automated monitoring with Zabbix

The Zabbix-Plugin integrates the Zabbix monitoring server and automatically monitors the systems and services managed by the openQRM-server. After enabling and starting the Zabbix plugin you can login to Zabbix as "Admin" with an empty password. Please make sure to set password for the "Admin" account at first login! All managed systems by openQRM will be automatically discovered and monitored by Zabbix. A detailed configuration of the systems service checks can be setup via the intuitive Zabbix UI.

Please notice:

Please make sure to have the Zabbix Server packages installed on the openQRM Server! The Zabbix plugin (for now) does not setup or preconfigure the Zabbix Server.

Automated monitoring with Collectd

The "Collectd" plugin automatically provides system statistics for Servers. It seamlessly integrates Collectd within openQRM and provides hourly, daily, weekly and monthly system graphs created from the collected data via rrdtool.

By enabling the plugin collectd is pre-configured and initialized automatically. The system graphs are updated sequentially via a cron-job. It may take some minutes after the start of an Server to collect enough data to create the graphs.

Automated power management with WOL (wake-up-on-lan)

The WOL plugin automatically controls the power state of physical systems (resources) in openQRM via the WOL (Wake-upOn LAN) technology. It supports to wake up physical systems (resources) in openQRM by sending a "magic network package". Via a provided "Server start hook" the "WOL" plugin is capable to startup the Server physical resources from the "off" state.

Please notice:

Please enable "Wake-up On LAN" in the BIOS of physical systems managed by openQRM.

Hint:

The "WakeUpOnLAN" Plugin is integrated with the openQRM Cloud! It provides the capability to automatically "scale up" and "scale down" the openQRM Cloud backend virtualization hosts. If configured in the Cloud openQRM automatically shuts down virtualization hosts that are not in use (no VM running on them). If needed (if the virtualization host limit is reached on all other available Hosts) openQRM starts a powered-off virtualization host Server via the WOL technology.

Administration

Accessing remote desktops and VM consoles with NoVNC

The NoVNC plugin integrates NoVNC into openQRM. It provides a remote console login to the openQRM server and the managed systems through the web interface. No manual configuration is needed.

Hint:

The “NoVNC” plugin provides the “remote-console” hook which is implemented for the following virtualization plugins: Citrix, KVM, Xen and VMware ESX

If enabled (and started) the NoVNC plugin features a “remote console” icon (and action) in the specific VM manager of the listed virtualization plugins.

Web SSH-Access to remote systems with SshTerm

The SshTerm plugin integrates webshell into openQRM. It provides a secure login to the openQRM Server and the managed systems through the web interface

Please notice:

The SshTerm plugin is also supported in the openQRM Cloud. If enabled the SshTerm plugins provides its functionalities also for the Cloud users.

LVM Device and Volume Group Configuration

Since the 5.1. version openQRM contains a “device-manager” Plugin which allows to configure physical Disk devices with LVM. This Plugin is used by several Virtualization and Storage plugins to allow an easy pre-configuration of the LVM Volume groups on the Storage.

Please notice:

While running “device-manager” LVM commands the plugin will automatically check the required package dependencies so the first action may take some time to resolve the dependencies.

Hint:

The “device-manager” is accessible through the Storage-Managers of the Virtualization and Storage plugins which are using LVM via a link “Add new Volume Group”.

Network-device and Bridge Configuration

Since the 5.1. version openQRM contains a “network-manager” Plugin which allows to configure network-interfaces and bridging. This Plugin is used by several Virtualization and Storage plugins to allow an easy pre-configuration of the network-devices and bridges for the Virtualization Hosts.

Please notice:

While running “network-manager” commands the plugin will automatically check the required package dependencies so the first action may take some time to resolve the dependencies.

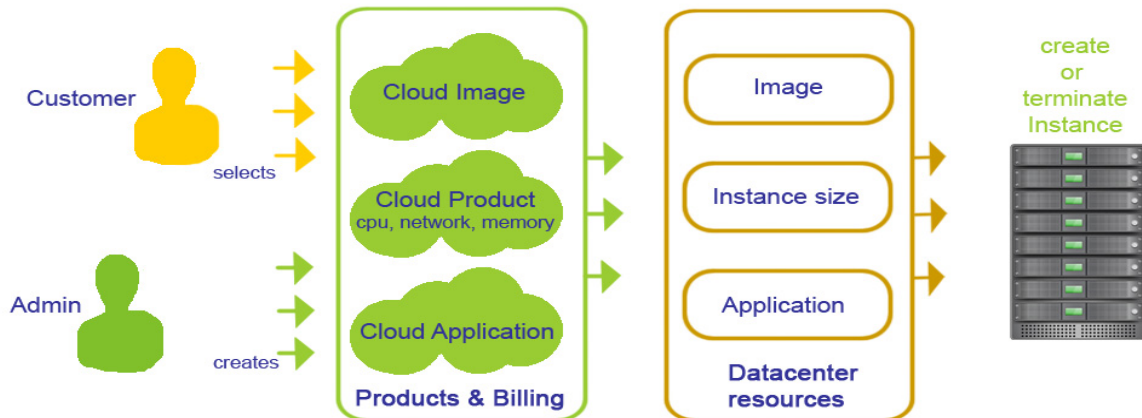
Hint:

The “network-manager” is accessible through the Network Section in “Edit Server”.

openQRM Cloud

Public and Private Cloud Computing

IaaS Cloud Computing “view” of openQRM



The openQRM Cloud can be used as public or private Cloud. It is fully based on openQRM's Server model and provides a fully automated deployment life-cycle for systems requested by Cloud users.

openQRM Cloud supports all virtualization and storage technologies supported by openQRM itself. Even physical systems can be deployed via openQRM Cloud in several different ways.

To use openQRM Cloud:

1. Install and setup openQRM on a physical system or on a virtual machine. Enable the following plugins: dns (optional), dhcpd, tftpd (only for network-deployment), cloud.
2. Decide on a deployment method (and eventual a virtualization technology), enable and start the specific needed plugins.

Please refer to the specific documentation for deployment method selected

- Configure the openQRM Cloud (main config). *See appendix for details.*
- Create Cloud products via the Cloud Product-Manager

1. Create custom CPU, Disk, high availability, kernel, memory, network, applications and virtualization products

Sort the products to your needs. The first 3 are always mapped to the “Small”, “Medium” and “Big” Cloud Server templates

Hint:

Disable Cloud products if they are still in use but you do not want to provide them any more. This way they are going to be still charged.

2. Select master images for Cloud deployment via the "Private Image" feature
3. Map Cloud usergroups to virtualization host resource via the "Resource Pool" feature
4. Configure virtualization host limits via the "Resource Limits" feature

If "Out-of-Band-Management" is setup and configured (please refer to the "WakeUpOnLAN" plugin and the "IPMI" plugin) configure the "Power Saver" functionality in the openQRM Cloud to automatically scale the physical virtualization host Servers up and down.

5. Decide on either "IP-Mgmt" or "Cloud NAT" for the "public" IP address assignment for the Cloud Servers.
 - For "IP-Mgmt" please create one or more IP networks and map them to the specific Cloud usergroups
 - For "Cloud NAT" please provide the private (nat'ed) and public IP address space in the Cloud NAT configuration (both Class C)

Please notice:

For Cloud NAT is required to manually setup the IP-Address translation on the out-going Firewall/Gateway e.g. with iptables and post/pre-routing

1. Setup Cloud users via the integrated user-management

provide "CCU's" (the virtual Cloud currency) to the Cloud users

eventually set per Cloud User limits

2. The openQRM Cloud is now available on the openQRM server at:

[http://\[openqrm-server\]/cloud-portal](http://[openqrm-server]/cloud-portal)

Hint:

The Cloud creates a lockfile at /usr/share/openqrm/web/action/cloud-conf/cloud-monitor.lock to ensure transactions.

Hint:

The Cloud zones User-management is integrated with the LDAP plugin and can use a centralized user-management provided by a LDAP (or Active Directory) server.

openQRM Cloud SOAP-Webservice

To easily integrate with third-party provision environments the openQRM Cloud provides a SOAP-WebService for the Cloud administrator and the Cloud users.

The openQRM WebService is developed in PHP using its integrated SOAP functions. It is implemented conform to the SOAP standard version 1.2. <http://www.w3.org/TR/soap12-part1/>

The openQRM Cloud SOAP-Server works in WSDL mode and provides the (automatic) provisioning- and de-provisioning functionality to a partner application.

Its webservice expose the Cloud user- and request-management of the openQRM Cloud. The functions (methods) handled by the SOAP-Server are combined into two separated PHP-Class for administrators and Cloud users. The classes also include methods to provide openQRM data (information about objects in the openQRM Cloud) to a partner application.

Since the openQRM webservice exposes administrative actions its (SOAP-) clients needs to be authenticated. The SOAP-Client will need to provide both a valid openQRM username and password of an openQRM user belonging to the administrator role (in case the "Administrator part of the Cloud WebService is used) or a valid Cloud-Username plus password (in case the "User" part of the Cloud web-service is used).

Please refer to the openQRM Cloud SOAP webservice documentation for more details.

Hybrid Cloud Computing – Migrating services between Public and Private Clouds

The Hybrid-Cloud plugin provides a seamless migration-path "from" and "to" Public-Cloud providers such as Amazon EC2 and Eucalyptus Cloud.

Please notice:

Make sure to have the latest "ec2-api-tools" and "ec2-ami-tools" installed on the openQRM server! To import/export from/to e.g. Amazon EC2 first setup a Hybrid-Cloud account.

Please get the latest EC2-API-Tools from <http://aws.amazon.com/developertools/351>

Please get the latest EC2-AMI-Tools from <http://aws.amazon.com/developertools/368>

The ec2- packages supplied by the linux distribution are usually outdated. Do NOT install the ec2 tools from there, but rather use the latest EC2 API/AMI tools directly from Amazon (see above).*

For a custom installation location of those Tools please configure `OPENQRM_CUSTOM_JAVA_HOME`, `OPENQRM_CUSTOM_EC2_API` and `OPENQRM_CUSTOM_EC2_AMI_HOME` in `openqrm-plugin-hybrid-cloud.conf`.

For additional or custom Hybrid-Cloud regions please configure `OPENQRM_PLUGIN_HYBRID_CLOUD_REGIONS` in `openqrm-plugin-hybrid-cloud.conf`

For an Eucalyptus Cloud type please install the latest Euca-API/AMI tools respectively.

Create a new Hybrid-Cloud Account configuration using the "Add new Account" link in the Hybrid-Cloud Plugins "Actions" menu.. The following information are required:

Hybrid-Cloud account name

Account type (Amazon Cloud or Eucalyptus Cloud)

AWS Access Key ID

AWS Secret Access Key

Optional Account Description

Hint:

Please find your AWS Access Key ID and Secret Access Key at your "Security Credentials" section of your Amazon AWS Account.

Manage and automate public and private clouds

AMIs

The AMI action allows to easily add (and remove) openQRM Image Objects for a specific public or private AMI. A list of e.g. public available Ubuntu AMIs for each Amazon EC2 Region is available at <http://uec-images.ubuntu.com/>

Import - import AMIs from a public or private Cloud

To import a cloud server (the AMI of an active EC2 instance) follow the steps below:

- Select a hybrid cloud account to use for import
- Select an active public cloud instance running the AMI to import

- Select an (empty) openQRM server image (of type NFS or LVM-NFS)
- Provide the private SSH key file (Keypair) of the instance to import

This will automatically import the AMI from the selected public cloud instance into the (previously created) empty server image in openQRM.

The imported AMI now can be used through 'network-deployment' in openQRM so e.g. it can now also run on a physical system or on any other virtualization type.

Export - export openQRM Images to a public or private cloud

To export an openQRM image to a public cloud provider as an AMI follow the steps below:

- Select a hybrid cloud account to use for the export
- Select the image (of type NFS or LVM-NFS) to turn into an AMI for the export
- Provide a name for the AMI and additional parameters such as size, architecture, the public and private key file plus the EC2 user ID (AWS Account ID)

This will automatically export the selected openQRM image to the public cloud provider. It will be available as new AMI as soon as the transfer procedure is finished.

Instances

In a simple way new instances can be launched in a specific public or private cloud region via the INSTANCES action. Additional to the common instance configuration parameters such as instance type, the security group and keypair a custom configuration script (URL) can be attached to the instance.

Hint:

Custom configuration scripts to automatically pre-configure the instance can be easily uploaded to S3 with the S3 actions file-manager.

Groups

Security Groups and custom firewall permissions are managed by the GROUPS action. Security Groups are attached to Instances to automatically configure the in- and outbound IP traffic.

Keypairs

Keypairs are managed by the KEYPAIR action. Same as Security Groups a Keypair is attached to Instances to all SSH login to the running Instances via a specific private SSH key.

Volumes

The VOLUMES action allows to create and snapshot EBS Volumes which can then be used to attach (or detach) them as additional blockdevices (hardddisks) to the instance.

Snapshots

This action provides the capability to manage the snapshots of the EBS Volumes e.g. for backup or re-deployment.

S3

The S3 action provides a Web-File-Manager for S3. It manages buckets and file uploads.

Hint:

Create custom configuration scripts to automatically pre-configure specific Instances and upload them to a S3 bucket via the 'file-upload' action in the S3 file-manager. Then attach the S3 URL of a custom configuration scripts to an Instance during 'Add Instance'. During start the Instance then automatically fetches the custom script from the S3 URL and executes it.

Automated application deployment

Automate application deployment for Hybrid-Cloud Instances with Puppet

The Hybrid-Cloud plugin is fully integrated with the openQRM Puppet plugin. This allows to manage application deployment for Hybrid-Cloud Instances in the same as for the internal IT resources. Simply add custom Puppet recipes to

```
/usr/share/openqrm/plugins/puppet/web/puppet/manifests/classes/
```

and group them into the the

```
/usr/share/openqrm/plugins/puppet/web/puppet/manifests/groups/
```

Every recipe in the groups directory automatically appears as selectable Puppet deployment in the openQRM UI. Edit your servers to assign them to the Hybrid-Cloud Instances.

Automated Monitoring

Automated service monitoring for Hybrid-Cloud Instances with Nagios

Similar to the Puppet integration the Hybrid-Cloud plugin fully integrated with the openQRM Nagios plugin. This allows to manage service monitoring for Hybrid-Cloud Instances in the same as for the internal IT resources.

Automated Highavailability

Automated Highavailability for Hybrid-Cloud Instances

openQRM is capable to provide highavailability on infra-structure level for Hybrid-Cloud Instances in a fully automated way. In case of an Instance failure or a failure of a complete availability-zone openQRM's highavailability plugin automatically triggers a seamless failover of the Instance to another availability-zone in the same region.

openQRM Cloud Integration

Automate and consolidate your public and private Cloud deployments with openQRM Cloud

The Hybrid-Cloud Integration of openQRM makes it easy to 're-sell' Public Cloud resources through openQRM Private Cloud. This way openQRM Cloud consolidates and fully automates internal deployments (within the own datacenter using own IT resources) and also external provisioning to different Public or other Private Cloud providers (using the Cloud Providers IT resources). Simply adding a Hybrid-Cloud virtualization product to the openQRM Cloud product manager enables the this option.

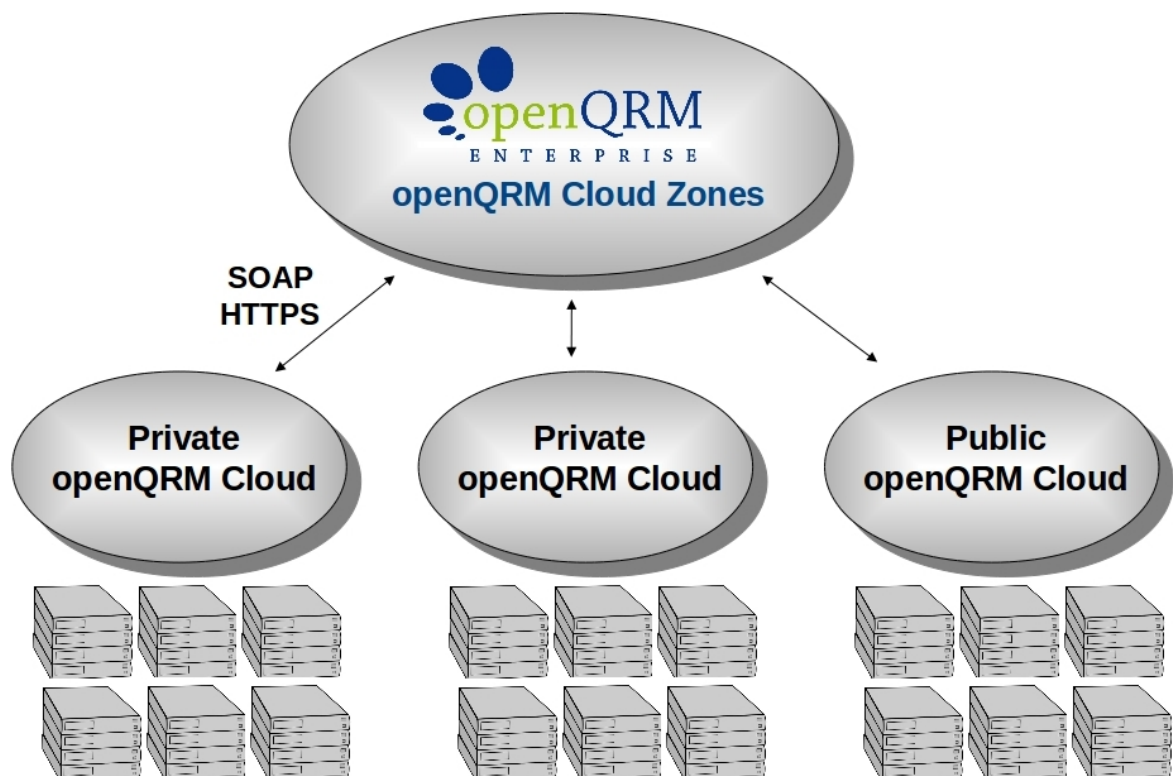
Howto

"Automated Amazon EC2 Cloud deployments with openQRM 5.2 on Debian Wheezy" at <http://www.openqrm-enterprise.com/resources/documentation-howtos/howtos/automated-amazon-ec2-cloud-deployments.html>

Enterprise Features

Cloud-Zones: Managing openQRM Clouds in multiple IT Locations

openQRM Enterprise Cloud-Zones is the layer on top of your existing openQRM Cloud infrastructure. It has the capability to partition multiple datacenter locations into logical zones and map access and permissions for each zone to e.g. your companies' business topology. openQRM Enterprise Cloud-Zones comes with a central, multilingual and intuitive Cloud portal with a plug-able Google-maps integration providing a detailed overview of where your data is located and your systems are running.



To setup openQRM Enterprise Cloud Zones:

1. Create one or more Cloud-Zones usergroups
2. Create one or more Cloud-Zones
3. Set the Cloud-Zones permission to the specific Cloud-Zones usergroup
4. Configure each openQRM Cloud which is part of openQRM Cloud-Zones as a "Cloud-Zones client" (main cloud configuration)
5. Create Cloud-Zones users within the different usergroups

Hint:

The Cloud-Zones user-management is integrated with the LDAP plugin and can use a centralized user-management provided by a LDAP (or Active Directory) server.

Cloud integration with E-Commerce systems via Cloud-Shop

The Cloud-Shop plugin connects openQRM Cloud to e-Commerce systems like e.g. Magento and Virtuemart.

- Install Magento or Virtuemart either on a dedicated system or on the openQRM server itself
- Enable the “cloud-shop” plugin
- Setup the shop database connection credentials in the Cloud-Shop plugin
- Create CCU products in the Magento/Virtuemart

Users and the amount of CCU's the (Magento/Virtuemart) users bought are synced automatically into the openQRM Cloud

Centralized user-management with LDAP

The LDAP-plugin connects openQRM and its Cloud-portal for User-Authentication to a LDAP (or Active Directory) server.

1. Install “openldap ” on the openQRM Server
2. Migrate the openQRM system accounts to the LDAP server using the LDAP migration tools
3. Configure the LDAP credentials in the LDAP plugin, save the configuration
Test the connection to the LDAP server with the saved credentials
4. Activating the LDAP Plugin configuration then takes over the user management for openQRM (Admin UI), openQRM Cloud and openQRM Enterprise Cloud-Zones.

Users in the “openqrm” user group are provided with the administration privileges in the openQRM Admin UI

Automated IP/Network/VLAN management with IP-Mgmt

The “IP-Mgmt” Plugin in openQRM provides an automatic IP-address configuration of the “public” (external) network cards (the eventual management network card is provisioned with an internal management IP-address by the “dhcpd” Plugin).

It allows to create several IP-networks (blocks of IP addresses plus DNS/Gateway/Subnet configuration) and to easily configure them on the public network interfaces of the Servers.

Please notice:

The “IP-Mgmt” Plugin on purpose provides the IP-configuration on behalf of the Server (representing your “Service”) and NOT on behalf of the resource!

e.g. in case of a high availability fail-over the resource will change (and therefore also its internal management IP-address) but the “IP-Mgmt” plugin will automatically take care to assign the same IP-address(es) to the restarted Server.

It is therefore recommended to configure the applications and services on the Server to use the “public” IP-address assigned by the “IP-Mgmt” plugin.

Hint:

The “IP-Mgmt” plugin provides hooks for the following virtualization plugins to automatically configure the VLAN tagging for the public (external) virtual machine network interfaces:

KVM (localboot), openVZ (localboot), Xen (localboot)

Automated IT-Documentation with I-do-it

The "I-do-it" plugin integrates with I-do-it to provide an automatic IT documentation (CMDB).

5. 5.1. Automatic configuration

To generate and/or update your IT documentation in I-do-it for the openQRM-network please use the "Config / AutoConfig" link in the I-do-it-plugin menu. The documentation will be created automatically and inserted/updated in I-do-it.

Please notice:

Please make sure to have an I-do-it server installed on the openQRM Server! The I-do-it plugin (for now) does not setup or preconfigure the I-do-it server.

Support for the Windows Operating System

The Windows plugin adds support for the Windows operating systems to openQRM. It consist of a small agent which runs on the Windows system as service.

The windows support has the following functionalities:

- sysprep prepare Windows Image for Cloning and Cloud Deployment
- VNC access to the VM console for Cloud Users
- pre-set administrator password
- pre-set System Hostname
- pre-set IP addresses (integrated with "ip-mgmt")
- KVM virtio support for Disk Interface (needs Windows virtio driver package)
- Puppet for Windows integration -> supports Windows App deployment (also in the Cloud)

Install the openQRM Client for Windows

Please request your openQRM client for Windows integration packages from openQRM Enterprise!

Copy the openqrm-client.exe to the Windows System at **C:** and run 'gpedit.msc' and add the the openqrm-client.exe as new Startup script

Local Computer Policy

- > *Computer Configuration*
- > *Software Settings*
- > *Windows Settings*
- > *Start*

As paramter for the openqrm-client.exe startup script please add your openQRM server ip address.

Sysprep the Windows OS

Download the openQRM sysprep configuration and copy the unattend.xml file to the Administrators Desktop on the Windows system and edit the AdministratorPassword->Value section with a hard-to-guess password.

Please notice:

Defragment the hard-drive! Before actually running syspre the hard-drive should be fresh defragmented!

Open a Windows command shell as Administrator and run

```
C:\Windows\System32\sysprep\sysprep.exe /generalize /oobe /shutdown /unattend:C:\Users\Administrator\Desktop\unattend.xml
```

After sysprepping the Windows OS shuts down and its Image is ready to be cloned/snapshotted for further regular and Cloud deployments.

Automated power-management with IPMI

The IPMI plugin automatically controls the power state of physical systems (resources) in openQRM via the IPMI technology. Via a provided "Server start hook" the "IPMI" plugin is capable to startup the Server physical resources from the "off" state. A "resource fence hook" is additional capable to "fence" a physical system in case of a fail-over via the "high availability" plugin (STONITH).

Please notice:

Please enable and configure "IPMI" in the BIOS of physical systems managed by openQRM.

Hint:

The "IPMI" Plugin is integrated with the openQRM Cloud!

It provides the capability to automatically "scale up" and "scale down" the openQRM Cloud backends virtualization hosts. If configured in the Cloud openQRM automatically shuts down virtualization hosts which are not in use (no VM running on them). If needed (if the virtualization host Limit is reached on all other available hosts) openQRM starts a powered-off virtualization host Server via the IPMI technology.

Network card bonding for network deployment

The NetBond plugin automatically configures "bonded" (consolidates) network-interfaces for "network-deployed" Servers in openQRM.

To configure the network card bonding simply edit a Server and set the "Capabilities" to e.g.:

```
BOND=bond0:eth0:eth1
```

to bond eth0 and eth1 to bond0.

The NetBond plugin also supports to configure multiple bonding interfaces. Here is an example parameter:

```
BOND=bond0:eth0:eth1,bond1:eth2:eth3
```

The specified network interfaces are automatically setup during Server start.

Please notice:

The "Netbond" plugin supports all different types of "network-deployment" in openQRM. It does not support any "local-deployment" deployment type!

Automated event mailer to forward critical events in openQRM

The Event-Mailer plugin allows to “forward” any types of openQRM events to third-party support-Tools or a human support-team member by mail.

To use the Event-Mailer Plugin please:

1. Make sure the openQRM server system is allowed to send out mails
2. Install and configure a Mail-Server on the openQRM Server (e.g. postfix)
3. Create one or more dedicated openQRM users
4. Setup the event levels to forward (mail) via the Event-Mailer plugin configuration

Users are now notified for each event type configured in the Event-Mailer plugin.

Secure, remote access for the openQRM Enterprise support team

The support plugin allows to open limited, authenticated, time-limited and encrypted access to your openQRM by the openQRM Enterprise support team. The support access can be setup and initiated (and stopped) via the support plugins.

Please notice:

Before setting up the support plugin please contact your openQRM Enterprise support team! They are looking forward to assist you!

Development

Development Plugin

The Development plugin provides developer informations like translation and rest api for base and plugins.

Template Plugin

The Template plugin provides a complete Skeleton for a new Plugin.

Debugging

Debugging openQRM actions and commands

Every action and command in openQRM can be easily debugged by the following steps :

1) check /var/log/syslog (or /var/log/messages on CentOS) when you run the action in the UI -> the openQRM command queue will tell you the full command it is running

e.g

```
...
openQRM-cmd-queue NOTICE : Running 80755de09e5e26fa78bbefa48f5f7f27 :dbclient -K 10 -y -i
/usr/share/openqrm/etc/dropbear/dropbear_rsa_host_key -p xxxxxxxx root@[ip-address] "/usr/share/openqrm/bin/openqrm-
cmd /usr/share/openqrm/plugins/vmware-esx/bin/openqrm-vmware-esx-vm post_host_statistics -i [ip-address]"
..
```

1) the command always starts with:

```
"dbclient -K 10 -y -i /usr/share/openqrm/etc/dropbear/dropbear_rsa_host_key -p xxxxxxxx root@IP"
```

-> this is the same as "ssh root@IP". openQRM does this with a private/public shared key in a secure way. So it runs a command on the remote system (or on itself)

1) then it runs:

```
"/usr/share/openqrm/bin/openqrm-cmd [full-commandline-on-the-host-it-is-running-the-command]"
```

/usr/share/openqrm/bin/openqrm-cmd is a simply bash wrapper setting paths and environment variables

[full-commandline-on-the-host-it-is-running-the-command] is the full commandline which is being executed on the remote system

4) Now run as root on the remote system (or on openQRM itself when the command is running locally)

```
bash -x [full-commandline-on-the-host-it-is-running-the-command]
```

or to send the output to a logfile

```
bash -x [full-commandline-on-the-host-it-is-running-the-command] 2>/tmp/logfile.txt
```

In the output of this "bash -x" command you will see every little micro step it is doing. In case the command fails this will exactly tell you where and why.

Appendix

Explanation of the openQRM Cloud configuration parameters:

`cloud_admin_email`

The email address for the Cloud to send status messages and events to.

`auto_provision`

If the Cloud should automatically provision systems or wait until a Cloud administrator approves the request.

`external_portal_url`

The external DNS/Domain name for the Cloud portal accessible for the Cloud users.

`request_physical_systems`

If the Cloud should also enable the automatic provisioning of physical server.

`default_clone_on_deploy`

By default the Cloud provisions a clone of the requested image (not the origin)

`max_resources_per_cr`

Global Cloud limit. Until openQRM 5.0 this is statically set to 1.

`auto_create_vms`

If the Cloud should automatically create virtual machines on available virtualization host Servers or if it should use a static pool of pre-created Vms.

`max_disk_size`

Global Cloud limit. Maximum overall disk space (in MB) used by a Cloud user.

`max_network_interfaces`

Global Cloud limit. Maximum overall network-interfaces used by a Cloud user.

`show_ha_checkbox`

If to show the high availability option for Cloud requests. Needs the high availability plugin enabled and started.

`show_puppet_groups`

If to show the automatic application deployment option (Puppet) for Cloud requests. Needs the puppet plugin enabled and started.

`auto_give_ccus`

Automatically provides some Cloud Computing Units (CCUS, the virtual currency of the Cloud) to new registered Cloud user.

`max_apps_per_user`

Global Cloud limit. Maximum overall number of active Servers used by a Cloud user.

`public_register_enabled`

If Cloud users should be able to register themselves via the public portal.

`cloud_enabled`

Use this option to set the Cloud in a maintenance mode. If set to false running systems will stay as they are but Cloud users will not be able to submit new requests.

`cloud_billing_enabled`
Enables/disables the internal billing mechanism. If disabled Cloud users will not be charged.

`show_sshterm_login`
If to enable/disable the Web-SSH login option for the Cloud users. Needs the sshterm plugin enabled and started.

`cloud_nat`
If to translate the (private) openQRM managed network to a public network. Requires to set pre/post-routing on the gateway/router to the external (public) network.

`show_collectd_graphs`
If to enable/disable System statistics for systems requested by the Cloud users. Needs the collectd plugin enabled and started.

`show_disk_resize`
If to provide the Disk-resize option to the Cloud users.

`show_private_image`
The private Image option allows to map certain Images to specific Cloud users.

`cloud_selector`
Enables/disables the Cloud Product-Manager.

`cloud_currency`
The real currency to which the virtual Cloud currency (CCUs) are mapped to.

`cloud_1000_ccus`
Defines the mapping/value of 1000 CCUs (virtual Cloud currency, CCUs) to the real currency defined in config option `cloud_currency`

`resource_pooling`
Allows mapping of virtualization host Servers to specific Cloud usergroups.

`Ip-management`
Enables/disables the automatic IP-address configuration for the external (public) network interfaces of the requested Cloud systems. Requires the ip-mgmt plugin to be enabled and started.

`Max-parallel-phase-one-actions`
Performance optimization parameter. How many actions should run in phase 1.

`max-parallel-phase-two-actions`
Performance optimization parameter. How many actions should run in phase 2.

`max-parallel-phase-three-actions`
Performance optimization parameter. How many actions should run in phase 3.

`max-parallel-phase-four-actions`
Performance optimization parameter. How many actions should run in phase 4.

`max-parallel-phase-five-actions`
Performance optimization parameter. How many actions should run in phase 5.

`max-parallel-phase-six-actions`
Performance optimization parameter. How many actions should run in phase 6.

max-parallel-phase-seven-actions

Performance optimization parameter. How many actions should run in phase 7.

Server_hostname

If to allow Cloud users to provision their own hostnames for their Cloud systems.

cloud_zones_client

If this Cloud is an openQRM Enterprise Cloud zones client.

cloud_zones_master_ip

Defines the openQRM Enterprise Cloud zones IP-address.

cloud_external_ip

Defines the public IP-address of this Cloud.

deprovision_warning

Sends a deprovision warning to the user when the configured CCU number is reached.

deprovision_pause

Pauses Servers of requests when the configured CCU number is reached.

vm_provision_delay

Delayed provisioning of virtual machines for N seconds.

vm_loadbalance_algorithm

Loadbalancing-Algorithm for virtual machines. 0 = Load, 1 = Memory, 2 = Random, 3 = First available host until host VM-Limit is reached.

allow_vnc_access

Defines if Cloud Users are allowed to connect direct to the VM console via VNC.

Contact



openQRM Enterprise GmbH
In der Kuppe 11
53175 Bonn / Germany

Telefon: +49 (0)228 / 85 09 882-0
Fax: +49 (0)228 / 85 09 882-1
Mail: info@openqrm-enterprise.com