The openQRM Datacentre Management and Cloud Computing Platform

Preface

This documentation is brought to you by OPENQRM AUSTRALIA PTY LTD

https://openqrm-enterprise.com/

Document Version: 24.03.2021

OPENQRM AUSTRALIA PTY LTD
Freedom Taxation Building
1Ransley St
Penrith NSW 2750
Australia
Telephone: +61 (0) 2 8069 9669
Email : info@openqrm-enterprise.com

Table of contents

The openQRM Datacenter Management and Cloud Computing Platform..........1
Preface..........1
Table of contents..........2
Introduction to openQRM..........6
History..........7
The why of it all..........7
Escaping the Vendor-Lock..........7
What means openQRM?..........8
Architecture, Design, Concept..........8
Services are just Files!..........8
The openQRM Boot Concept..........10
openQRM's automatic hardware detection..........11
openQRM Base..........12
    Definition of a Resource..........12
    Kernel..........12
    Image..........12
    Appliance..........12
    Storage..........13
openQRM Plugins..........14
    The Plug-in Manager..........14
    Remote Command Execution..........16
    Flow of an openQRM command..........16
Requirements..........16
Supported Platforms (Storage, Hosts, Guests)..........17
System Provisioning..........18
Generic Provisioning – openQRM's Appliance Model..........18
Virtualization Host Deployment..........19
Virtual Machine Deployment..........19
    Please notice:..........19
Storage Layer..........19
Pluggable Storage Types..........19
Storage Management..........21
Storage Authentication..........22
Virtualization Management..........22
Pluggable Virtualization Types..........22

Hypervisors are just Resource Providers! 22
Server Image Management 23
Installing from NFS 24
Transfer to NFS 24
Installing from Local 25
Transfer to Local 25
Remote Deployment 26
Local Deployment 26
P2V – Physical to Virtual Migration 26
V2P – Virtual to Physical Migration 27
V2V – Virtual (type A) to Virtual (type B) Migration 27
Service High-Available 27
HA on Resource Level 27
HA on Application Level 28
HA for the openQRM server 28
Plugin description 29
- **aws** 29
- **citrix** 30
- **cloud** 30
- **Cloud IpGroups** 33
- **Cloud Admin** 33
- **SOAP-WebService** 33
- **collectd** 33
- **dhcpd** 34
- **dns** 34
- **equallogic-storage** 34
- **highavailability** .... **Error! Bookmark not defined.**
- **image-shelf** 35
- **iscsi-storage** 36
- **kvm** 36
- **kvm-storage** 37
- **linuxcoe** 37
- **local-server** 37
- **local-storage** 38
- **lvm-storage** 39
- **nagios3** 39
- **netapp-storage** 40
- **nfs-storage** 40

sanboot-storage 41
solx86 41
sshterm 42
tftpd 42
tmpfs-storage 42
vmware-server 42
vmware-server2 43
vmware-esx 43
windows 44
xen 44
xen-storage 45
zfs-storage 45
Getting Started 46
Setting up the server 46
Installing openQRM 47
Debian/Ubuntu 47
Updating openQRM 48
Configuring a Basic Setup 49
Pre-setup the Hypervisor: 49
Pre-setup LVM 49
Activate DHCP, TFTP, Storage Plugins 50
Creating Images 50
Image Locations 51
Logical Images 51
Using a pre-made Image from Image Shelves 52
Adding Resources 52
Adding Kernels 53
Creating Appliances 53
Common Use Cases 54
Scalable High-Availability Setup 54
Integration into existing IT infrastructures 56
Windows deployment 56
kvm-storage 56
xen-storage 58
sanboot-storage 59
Installing the Windows openQRM-Client 59
Automatic Configuration Management 60
Plugin Boot Services 60

Automated configuration management via Puppet 61
Monitoring 61
Basic Monitoring 61
Nagios 61
collectd 62
Zabbix 62
The openQRM Cloud 62
What is Cloud Computing? 63
Comparison of major Cloud Competitors 64
openQRM Cloud 64
Development 65
The openQRM Build System 65
Plugin Development, Third-Party Integration, API 67
Image Hooks 71
SOAP Web Services 72
Coding Guidelines 72
Security Considerations 72
openQRM Security Concept 72
SSL encryption for the UI access 72
Storage authentication 73
Securing the Network 73
Securing your Services 73
Getting Support 74
Thanks 74

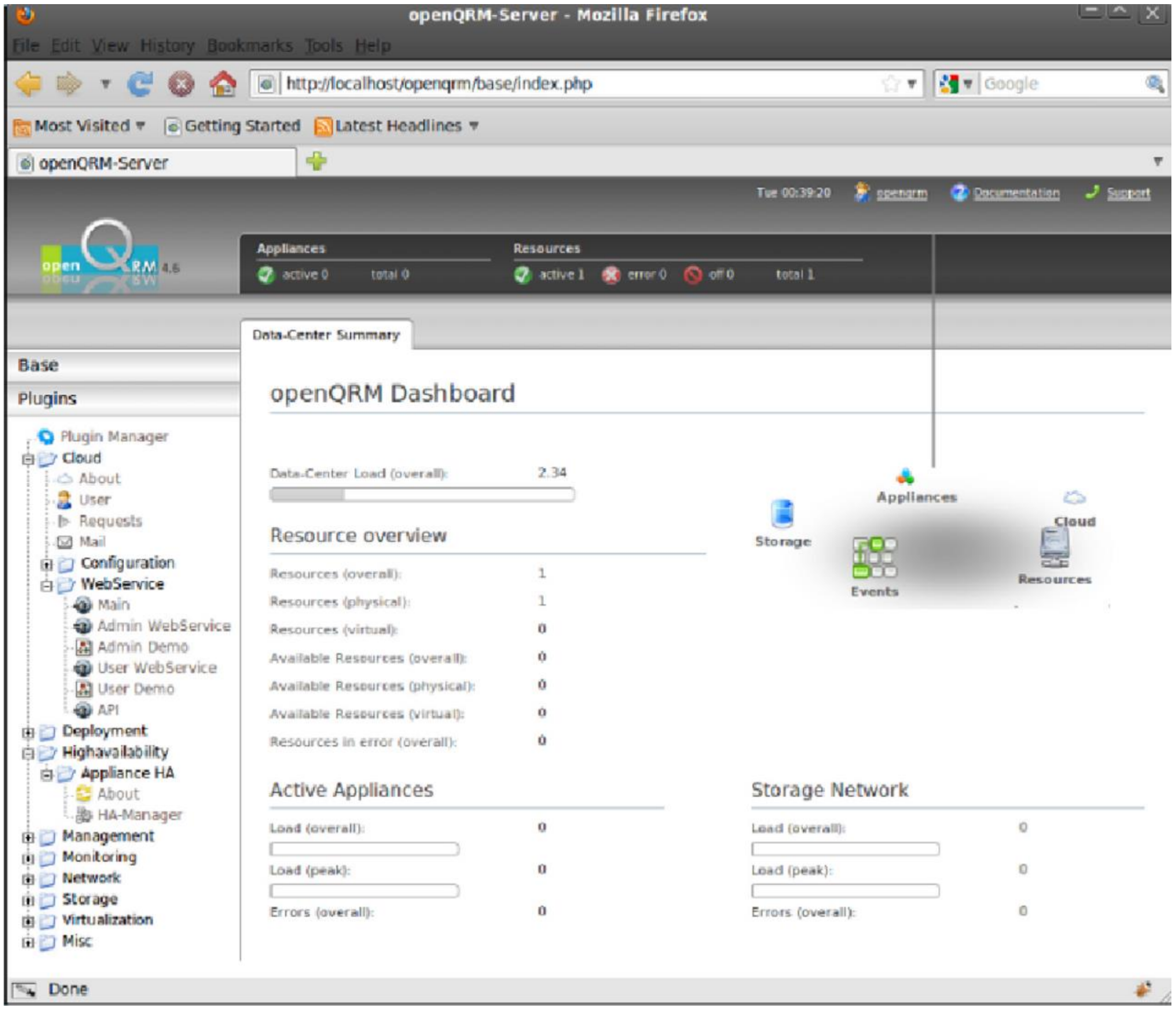
Introduction to openQRM

openQRM is the next generation, Open- S o u r c e Data Center Management and Cloud Computing Platform designedto fully automate data centers and manage them in a scalable way. Among its many characteristics is a unique architecture that unifies physical and virtual machine deployment within a single management console. openQRM integrates with all mainstream virtualization technologies and supports transparent P-to-V, V-to-P and also V-to-V migrations.

openQRM's storage integration uses snapshots to clone servers for rapid deployment, backup/restore, server versioning, disk re-sizing and persistent cloud storage. openQRM also provides “N-to-1” failover allowing groups of servers to use a single “standby”. It also has the ability to fail over from physical to virtual machines, and an open APIto integrate with existing business processes and other data center related tools.

With its concept of “plug-ability” openQRM combines proven open-source products and commercial third-party components for data center administration, system- and service-monitoring, high-availability and automated provisioning within a single management console.

Here a screenshot of openQRM's Datacenter Dashboard:



openQRM Dashboard

Data-Center Load (overall): 2.34

Resource overview

Resources (overall):	1
Resources (physical):	1
Resources (virtual):	0
Available Resources (overall):	0
Available Resources (physical):	0
Available Resources (virtual):	0
Resources in error (overall):	0

Active Appliances

Load (overall):	0
Load (peak):	0
Errors (overall):	0

Storage Network

Load (overall):	0
Load (peak):	0
Errors (overall):	0

History

The initial version of openQRM was developed by the Qlusters company founded in 2001. While Qlusters first concentrates on High-Performance-Computing (HPC) it changed its business focus to Data-Center Management in 2003. The first openQRM version was based on the Java programming language und further developed until version
3.1.4 when the company closed its doors beginning of 2008. Amazing that 2005 openQRM already included a fully automated provisioning system and the support for Virtual Machine deployment very much similar than today’s concept of “Cloud Computing”, just it was named “Utility Computing”. Caused by a long development history the last versions of openQRM 3.x grew very big and complex. Because of that development progress slowed down very much in the last years of Qlusters. Luckily, they decided to change the initial commercial license for openQRM to an open-source one (MPL) in 2006. Matthias Rechenburg was working for Qlusters since day one and from the start of the open-source release Project Manager of the openQRM project. When Qlusters shut down in 2008 he decided to continue openQRM as a community-driven open-source project.

At this point, starting with the 4.0 release version, openQRM got re-written from the scratch, its features and mechanisms were ported to the much lighter PHP language and its license was updated to GPL Version 2. Within just 2 years of re-design, development and QA the openQRM team succeeded in outperforming the functionalities of the “old”, Java-based openQRM and got the new version enterprise-ready.

The why of it all

Datacenter are always custom and complex environments. It is taking a lot of effort and hassle to maintain them. The complexity originates from the number of involved subsystems and from the complexity of each subsystem. In a modern Datacenter there are always quite a few subsystems involved like physical servers, virtual machines, different operating systems, network components, network configuration and network services like DNS and dhcpd plus a system- and service monitoring, backup/restore, out-of-band management and so on. Preferred would be to have all those different aspects managed centralized within a single administration console which is exactly the goal of the openQRM framework.

Fitting this kind of complexity into a single application sounds impossible?

openQRM's main concept is to “break” Datacenters into manageable subsystems. In openQRM each subsystem is separately implemented via a plugin which provides the functionality to manage it. Then openQRM creates automated and generic interfaces between the different components via its completely plug-able software architecture. Actually, the base server of openQRM is designed to just have a single function, to manage plugins. That way new features like additional deployment-, storage- and virtualization types can be added to openQRM without changing a single line of code in the base server. Not only that this concept keeps the base server always small, static and robust but it also allows several developers working on different plugins in parallel without interfering with each other’s changes.

Escaping the Vendor-Lock

One of the main principals of openQRM is that it generates generic interfaces between different subsystem resultingin a standardized, flexible and dynamic scalable IT environment while avoiding not necessary dependencies. The specific subsystems are implemented by either Open-Source or commercial third-party components by fitting them into openQRM's Datacenter abstraction layer. To provide variety and specific sysadmin preferences openQRM always tries to give several options regarding the implementation of each subsystem. An example for that are the automated monitoring solutions supported and integrated in openQRM such as Nagios3, Zabbix and Collectd plus openQRM own basic monitoring service.

You should use what you like and prefer!

The different Virtualization technologies and approaches to unify them almost all try to solve the same problem of

specific, sometimes even closed format of the virtual Hard disks. The fact that every Virtualization technology is using their own virtual Hard disk format makes migrating Systems, running in those Virtual Machines, to another Virtualization technology or even back to physical system.

To avoid any locking to specific Virtualization Vendors openQRM therefore provides a unified virtualization layer whichsits on-top of each Virtualization technology and conforms them within the openQRM server. In openQRM the (server-) images are directly connected from the storage, through the network, to the virtual Machines from any type. Via this unified virtualization layer openQRM supports seamless migration from physical systems to virtual machines (P2V), from virtual machines back to physical systems (V2P) and also the migration from Virtualization technology A to Virtualization technology B (V2V).

openQRM also avoids Vendor-Locks on the storage subsystems by integrating with various storage providers such as NFS, iSCSI, AOE (Coraid), EqualLogic, Netapp and ZFS.

Sysadmins should be able to choose and select the technology fitting best for their business

What means openQRM?

The name openQRM means “open Qlusters Resource Manager” and is based on its commercial history. To avoid confusion and to keep the already big openQRM community the openQRM team decided to continue with this name after the previous main sponsor Qlusters shut down their company. The openQRM Project is now a fully community driven, Open-Source project backed up again by its new main Sponsor openQRM Enterprise.

Architecture, Design, Concept

Services are just Files

An important question about Datacenters is:

Are we looking at “Services” or “Servers”?

Is it important for us that specific physical (or virtual) hardware continues to work or is it more important to keep the services provided by the whole Datacenter up and running?

Since everything which moves (like CPU ventilators and Hard disks) will break at some point it seems kind of seriousto avoid binding Datacenter services to physical hardware. But even virtual machines are running on their Hypervisors physical hardware.

But then, what is the best place to store our most important services? The answer is: Modern StorageSystems

Modern Storage appliances coming with out-of-the-box support for High-Availability, Scalability and Data-security through enhanced RAID Hard disk arrays allowing to hot-swap failed disk without interruption. All serious files and data within a Datacenter should be stored on those kind of storage systems to ensure data-availability, its integrity and to have a single place for backup and restore. At the end it will also help to make your sysadmin sleep better …

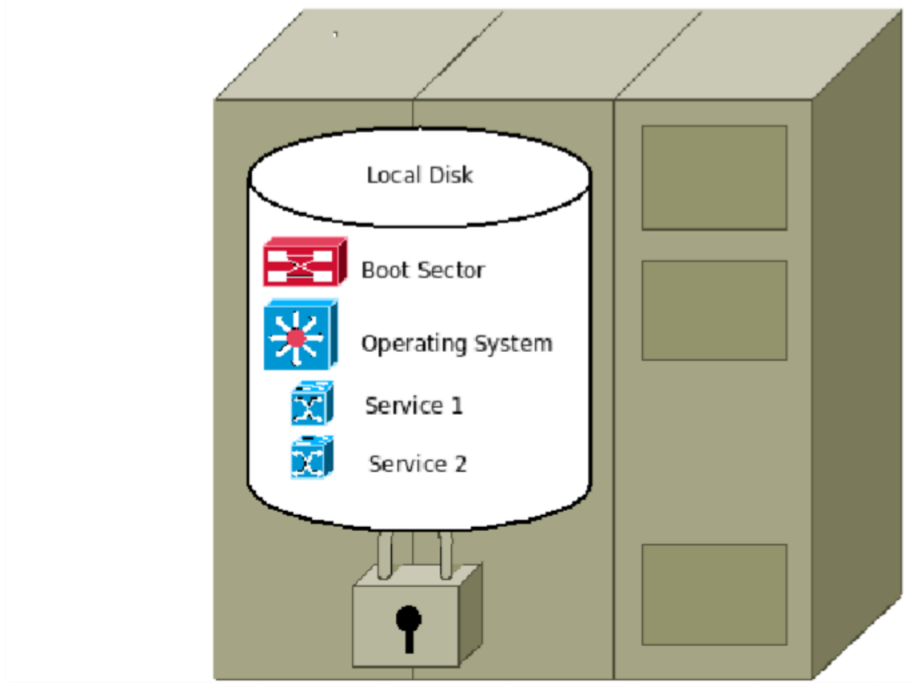
openQRM's best-practice recommendation is to avoid using local Hard disks because they are the first thing which fails in physical servers.

And how we usually install server?

- Server gets ordered and placed in the datacenter
- It gets connected to the network
- The operating system is being installed on the local Hard disks
- The OS gets its special configuration
- Application are being installed on it
- The Applications gets their final configuration and data

- The system is being added to the Backup
- Monitoring is being setup

Operating System plus appliaction services tied to physical Hardware



When knowing that the Hard disks are the weakest part of our physical (or virtual) hardware why we still bind the most important parts, the services, applications and servers running on the OS, to it?

With its unique architecture openQRM provides a generic Datacenter-Abstraction layer which completely separates the “services” from physical servers or virtual machines by storing and using them directly from a robust and high-available centralized storage.

From the point of view that those Datacenter services are the most important part the openQRM team especially looked at Linux server and asked themselves:

What is a running server system?

A running (Linux-) server consist of the following components:

- a Linux Kernel, which is a file …
- some kernel modules, which are files …
- an initial Ram disk (initrd), which is a file …
- a root-filesystem containing the applications and servers, which are all just files …

When a running Linux system, providing all the important Datacenter services we would like to keep up and running all the time, is just a bunch of files then we should start treating them as “files”. First thing is to store them in a save place, on a modern Storage system (there are several more advantages of using modern Storage systems explained later).

Next special observation was about the initrd-stage, when Linux systems are using a small, initial ramdisk to pre-setup the init procedure by running /sbin/init. The only purpose and responsibility the initrd has in Linux is to find and mountthe root-filesystem, normally located on a local Hard disk. openQRM uses this generic Linux mechanism in an enhanced way by making the process of finding and mounting the root-filesystem completely plug-able. In openQRM the specific storage-plugin tells a booting system within the initrd stage from where and how it should mount its root-filesystem.

Since storage-types are also plug-able in openQRM any kind of external, remote storage devices can be used!

One “funny” example is a possible “gmailfs-storage” plugin which would support booting up systems from Google-Mail account storage. …. sure, just a funny example, would be slow as dry bread … but on the other hand Google would care about your backups.

Please find out more about unique boot concept in the next chapter “The openQRM Boot Concept”

The openQRM Boot Concept

Regular boot from Hard disk

- System is being turned on
- Bios reads boot-sector from first Hard disk
- Boot-manager is being executed from the Disk
- Boot-manager loads Operating System from local Hard disk
- Operating System is being executed and loads its components from the local Hard disk
- Operating System starts applications and services

Now at some point the Hard disk will break and loading the boot-manager and/or the operating system will fail.Here how openQRM overcomes this situation with a centralized, network boot-manager:

The openQRM way

- System is being turned on
- Bios is configured to do a network-boot (PXE)
- System sends a PXE request and asks for an automated network-configuration via dhcp
- openQRM's dhcpd-server answers the request and provides an ip-address
- System activates its network-configuration and reads its PXE-configuration from the openQRM server
- System downloads its operating system kernel and initial ramdisk (initrd)
- System executes the Operating System kernel and loads the initial ramdisk
- Within the ramdisk network-hardware is being automatically detected and initialized
- Having full network connection, the system now downloads its full set of kernel modules
- Additional hardware-detection with all available kernel modules
- System gets configuration parameters from the openQRM server

At this stage all available hardware is detected and the network is fully initialized. Here the system can continue in two different ways:

1. As idle, free (meaning available) server resource
2. As an active, assigned resource acting as a service provider

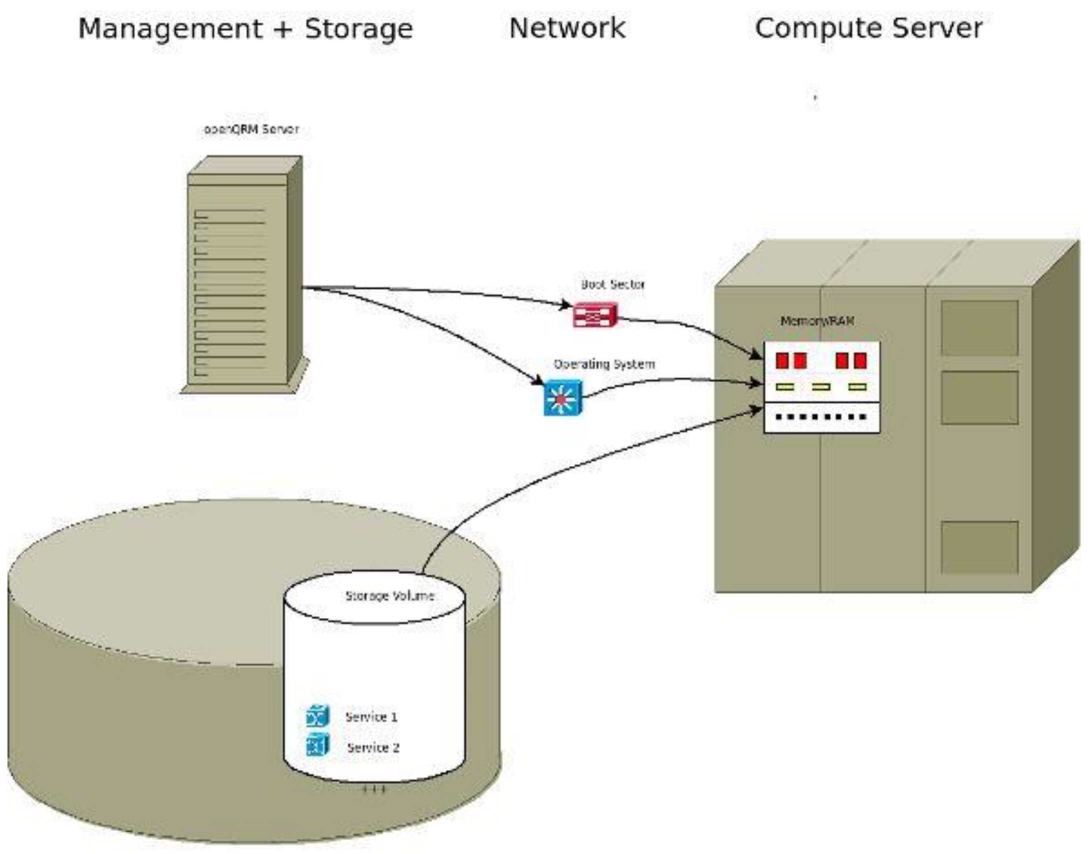
If the system is not assigned yet (1) it will simply stay within its initial, minimal ramdisk waiting for further actions from

openQRM. In case the system is active assigned to a started appliance (2) here the further steps it will execute:

- System checks the deployment type and method of the server-image assigned to its appliance
- According to the deployment type the system downloads image-deployment hook provided by specificstorage-

plugin

- System executes “mount_root” function provided in the image-deployment hook
- The image-deployment hook mounts the server-image-location from the remote storage read-writeable
- Kernel- and kernel-modules files are being transferred to the mounted root-filesystem
- The openQRM-client is being installed on to the mounted root-filesystem
- The image-deployment hook re-mounts the server-image-location read-only
- System continues with regular init (running /sbin/init on the root-filesystem)
- During further init procedure of the System the openQRM-client is started
- According to the activated plugins in openQRM the System now starts further plugin services



Operation System and application services separated from hardware

And what if the network fails?

…. then the provided network services of the Datacenter will not work anyway and one should fix the network asap.

openQRM’s automatic hardware detection

For the past openQRM used a combination of “hwsetup”, “kudzu-knoppix” and “hwdata” to automatically detect hardware during boot up. Adapted from Knoppix this worked great for a long time. While this method meanwhile is not very well maintained any more it recently creates troubles especially on CentOS Kernels as reported.

To overcome this situation we researched a new, much more efficient way for openQRM's hardware detection using “pcimodules”. The “pcimodules” command is available as a patch for the “pcituils” package and simply lists all needed kernel modules according the pci ids of the detected hardware. Using this tool openQRM's hardware detection now could be reduced to 3 lines of code:

```
for module in $(pcimodules) ; do
 modprobe -s -k "$module"
```

While testing this new “pcimodules” hardware detection method on physical and virtual systems our QA reported that much more hardware was detected compared to the previous “hwsetup” utility. Even USB devices are detected flawlessly.

openQRM Base

Definition of a Resource

A resource in openQRM is “everything which has a CPU and some memory”. Resources in openQRM have different types such as Physical Systems, Virtualization Host or Virtual Machine. According its type openQRM interfaces and communicates with the specific resource.

Since the openQRM platform is recommending (not forcing) to avoid using local Hard disks a resource in openQRM isjust CPU and memory. The resource's local Hard disk is in general NOT part of the resource itself but can be used for e.g., swap space or even for local-deployment. In general, openQRM main concept is based on rapid, image-based deployment and the server-images of the current running services are located on remote storage devices.

The openQRM team recommends to use the Local Hard disks available in resources (e.g., physical servers) as the local swap devices for virtual machines running on this system. Makes sense to have the swap space local.

Other than that, the local disks still can be used as storage for applications or customer data. One just need to keep in mind that this again binds the service currently running on the specific resource to its local disk. In this case it is ofcourse recommended to have a separated backup procedure for the local disk data.

Kernel

Kernels in openQRM are Linux Operating System kernels which can be assigned to resources. This happens automatically through the appliance model through openQRM's integrated, centralized network-boot-manager PXELINUX from the Syslinux project [http://syslinux.zytor.com].

Image

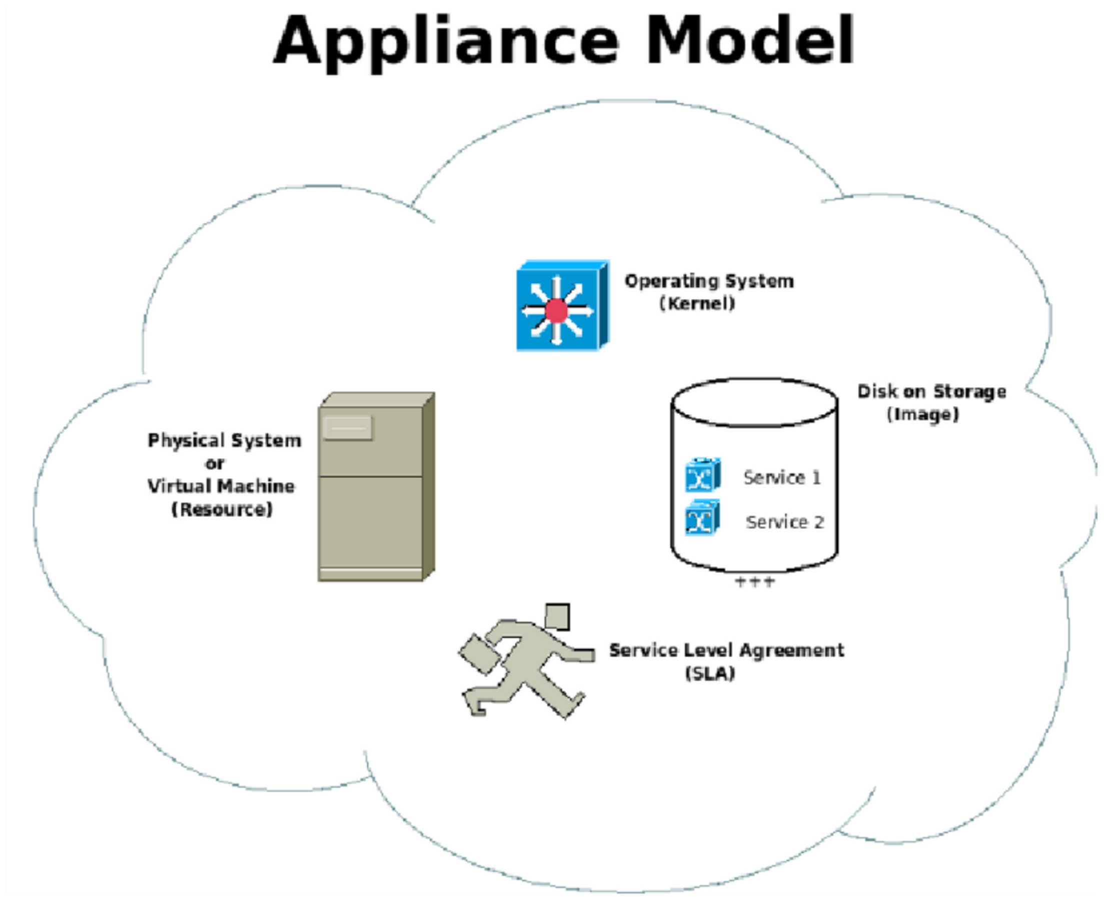
The definition of an image (server-image) in openQRM is that it is located on a network-attached storage device (NASor SAN) and contains a root-filesystem of an Operating system supported by openQRM. The image is completely self-contained and, via the appliance model it and in combination with a kernel can be started on any available (idle) resource.

The server-image root-filesystem may be a minimal Operating system installation which is then further leveraged (e.g.,via the Puppet integration) or it also can be a full installed completely pre-configured set of applications. An image can even be a snapshot of an existing server or a clone or copy of an existing server-image.

Appliance

Appliances in openQRM representing one (or more) of the actual services which should be provided by the Datacenter. An appliance is the combination of a kernel, an (server-) image, a resource and service requirements plus service-level-agreements (SLA). With those information’s openQRM then fully automates the management of the specific services running on the appliance's server-image.

Appliances are just like TVs; it takes a single button to start or stop them and they will always provide the desired service.


Appliance Model
Operating System (Kernel)
Disk on Storage (Image)
Service 1
Service 2
+++
Physical System or Virtual Machine (Resource)
Service Level Agreement (SLA)

Please check the chapter about High-Availability to see how openQRM automatically keeps Datacenter services running.

Storage

A storage component in openQRM consists of an integrated resource containing some kind of network-attachable storage (NAS or SAN). Storages in openQRM are providing the image-locations, meaning the place where server- images are stored and directly attached to resource as required. By creating a storage server openQRM then exactly knows how to interface with its specific storage technology and further allows automated management of the available storage space and volumes.

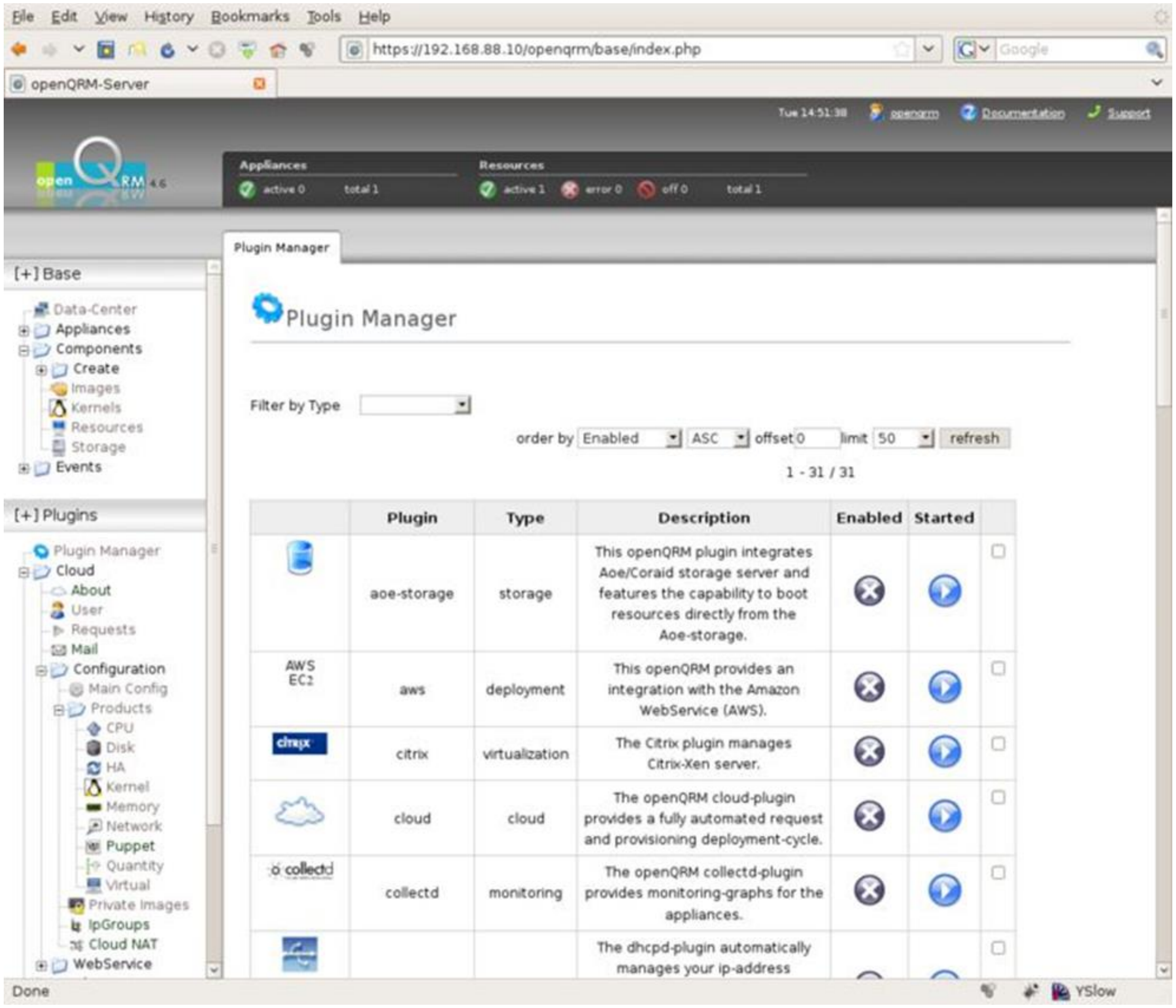
openQRM supports a whole bunch of different storage server types such as NFS, iSCSI, AOE, EqualLogic, Netapp and ZFS but it also provides own custom storage types such as the “lvm-storage” plugin which is based on Logical

Volume Management (LVM). As virtualization types storage types are fully plug-able in openQRM so that new customstorage devices can be integrated easily.

Just like appliances the storage resources are also running the openQRM-client service which allows remote-management of the storage-subsystem by openQRM.

openQRM Plugins

Here a screenshot of openQRM's Plugin-Manager:



Plugin Manager

Filter by Type

order by Enabled ASC offset 0 limit 50 refresh

1 - 31 / 31

	Plugin	Type	Description	Enabled	Started	
	aoe-storage	storage	This openQRM plugin integrates Aoe/Coraid storage server and features the capability to boot resources directly from the Aoe-storage.			
AWS EC2	aws	deployment	This openQRM provides an integration with the Amazon WebService (AWS).			
citrix	citrix	virtualization	The Citrix plugin manages Citrix-Xen server.			
	cloud	cloud	The openQRM cloud-plugin provides a fully automated request and provisioning deployment-cycle.			
collectd	collectd	monitoring	The openQRM collectd-plugin provides monitoring-graphs for the appliances.			
			The dhcpd-plugin automatically manages your ip-address			

The Plug-in Manager

Since its Plug-ins are providing all the features of the openQRM Server the Plug-in Manager is the central point to enable (or disable) additional functionalities for the openQRM managed environment. By default, it presents a list of all available Plug-ins plus their state. Plug-ins in openQRM can have 3 different states:

- Disabled (Stopped)
- Enabled and Stopped
- Enabled and Started

The Plug-in state can be changed either through the “action” icons in the specific Plug-in row or via the group actions button at the end of the list.

Detailed Plug-in states description

- Disabled (Stopped)

The specific Plug-in functionality and its menu entry are disabled

- Enabled and Stopped

The specific Plug-in functionality is initialized and its menu entry is enabled. Eventual Plug-in services are not started.

- Enabled and Started

The specific Plug-in functionality is initialized, its menu entry is enabled and eventual Plug-in services are started.

Enabling, Disabling, Starting or Stopping a Plug-in in openQRM results in 4 different Plug-in actions:

- Plug-in Init
- Plug-in Start
- Plug-in Stop
- Plug-in Uninstall

For each action openQRM submits a Plug-in command to its internal command queue which then runs the commands through the “Remote command execution system” (next Chapter).

Detailed description of the Plug-in commands

- Plug-in Init

For the Plug-in Init action openQRM runs:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/PLUGIN_NAME/etc/init.d/openqrm-plugin-PLUGIN_NAME  init  ADMIN_USER  AD
```

- Plug-in Start

For the Plug-in Start action openQRM runs:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/PLUGIN_NAME/etc/init.d/openqrm-plugin-PLUGIN_NAME   start
```

- Plug-in Stop

For the Plug-in Stop action openQRM runs:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/PLUGIN_NAME/etc/init.d/openqrm-plugin-PLUGIN_NAME   stop
```

- Plug-in Uninstall

For the Plug-in Uninstall action openQRM runs:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/PLUGIN_NAME/etc/init.d/openqrm-plugin-PLUGIN_NAME  uninstall  ADMIN_US
```

For the “Init” and “Uninstall” actions a valid openQRM administrator user and password are required. Thoseparameters are used by the Plug-ins to create (or drop) additional Plug-in tables in the openQRM database.

Manual running Plug-in actions

By running the above listed Plug-in action commands Plug-ins states can also be changed from the command-line.

Plugin Hooks

openQRM offers a great selection of “Plug-in Hooks” allowing each Plug-in to run commands when e.g., resource or appliance state changed. This mechanism is described in detail in the later “Plug-in Development” section.

Remote Command Execution

openQRM's Framework for (remote) command execution

As the central management tool of complete IT environment openQRM needs to be able to run administrative commands on the openQRM server itself but also on other integrated, remote systems within the openQRM managed network. Therefore, openQRM provides an SSL-secured remote command execution subsystem consisting of 3 major components:

- A “dropbear” Server automatically deployed and started on every resource managed by openQRM
- The public-shared SSL key of the openQRM Server also automatically deployed to every resource
- The “dbclient” util executed by the openQRM command queue

“Dropbear” is a minimal SSH-Server and Client which openQRM uses to create a secure transmission channel between the openQRM server and its managed resources. Via a public-shared SSL key mechanism openQRM gains password-less, secure access to all systems within the managed network. Also commands which are being run on the openQRM server itself are using this safe mechanism.

Flow of an openQRM command

1. PHP resource class sends a command through its “send_command” method.
2. Via the openqrm-exec util (OPENQRM_SERVER_BASE_DIR/openqrm/sbin/openqrm-exec) this method puts the new command plus its parameters and unique token in the openQRM command queue (OPENQRM_SERVER_BASE_DIR/openqrm/var/spool/) using the token for the command filename.
3. From there the command is being picked up by the openqrm-cmd-queue (OPENQRM_SERVER_BASE_DIR/openqrm/sbin/openqrm-cmd-queue), a service started by the openQRM Server init script.
4. The openQRM command queue sequentially runs the commands and executes them on the specified (remote) hosts through the “dbclient” util.

All commands in openQRM are fully logged to “syslog”. Failed commands are recognized by the openQRM command queue. In case a command fails in the queue it is re-scheduled to run 3 times. After a command failed 3 times openQRM generates an Error-Event which shows up in the Central Event log. In the event log sysadmin get a brief overview of the command which failed, its error message and the opportunity to Re-Run it.

Requirements

openQRM is designed with an infinite linear scalable architecture providing flexible solutions for custom Datacenter. Depending on the requirements of the IT environment managed by openQRM therefore it offers nearly infinite waysto set it up. Following 3 example Use-cases and their system requirements:

Simple Proof-of-Concept Setup

- 1 Physical System dedicated for the openQRM Server
- VT CPU Extension (full Virtualization Support)
- A free partition dedicated for the server-image store (at least 20 GB)
- 1 GB Memory (the more the better)

It is no problem at all to run openQRM, the Storage- and Virtualization-part on a single system for e.g., a POC setup.

Basic Setup

- 3 Physical Systems (one for the openQRM Server, one for Storage and one as a Virtualization Host)
- VT CPU Extension (full Virtualization Support for the system dedicated as Virtualization Host)
- A free partition dedicated for the server-image store (at least 100 GB) on the system dedicated for the Storage

The basic setup basically starts to distribute the Storage- and Virtualization components to external servers and therefore provides a better performance and scalability.

Production Setup

- 4+N Physical Systems (two for openQRM Server HA, N Storage and N Virtualization Hosts)
- VT CPU Extension (full Virtualization Support for the systems dedicated as Virtualization Hosts)

For a production setup it is recommended to take care of the high-availability of openQRM. This is archived by an active-passive setup for the openQRM Server (e.g., using Linux-ha). For perfect scalability openQRM supports an unlimited number of Storage- and Virtualization resources. New resources (Storage and/or Virtualization Hosts) canbe added to the openQRM managed network at any time.

Scaling up

Since openQRM fully supports dynamic IT environments it is possible and easy to scale from a simple setup to a basic one or even to a production setup at any time.

Supported Platforms (Storage, Hosts, Guests)

openQRM distinguishes between different types of resources in the managed network. There are Storage resources, Virtualization Hosts resources, Virtual Machine resources, Systems dedicated for rapid deployment and the openQRM Server itself. According to the resource type openQRM includes support for various different Operating Systems, Storage- and Virtualization appliances.

Supported Operating Systems for the openQRM Server:

- Ubuntu
- Debian
- CentOS

It is recommended to always use the latest version of the above listed Linux Distributions.Supported Operating

Systems for rapid deployment:

- Ubuntu
- Debian
- CentOS
- Fedora
- Suse/openSUSE Windows
- (XP/Windows 7)
- Solaris/open Solaris

Also, for the Systems dedicated for rapid deployment it is recommended to always use the latest version of the above listed Linux Distributions.

Supported Storage Appliances:

- Dell EqualLogic

- NetApp Filer
- Solaris/open Solaris ZFS Store (e.g., Nexentastore)

Additional to those third-party Storage providers openQRM comes with a whole set of own, custom storage types which are turning a simple Linux Box with an LVM partition into a rapid cloning, robust server-image store for openQRM.

- Supported Virtualization Vendors

- VMWare Server 1
- VMWare Server 2
- VMWare ESX / VMWare Vsphere
- Citrix XenServer
- Xen
- KVM

System Provisioning

openQRM comes with a complete generic deployment abstraction layer and appliance model which conforms provisioning of physical system and virtual machines from any type. First thing to do to deploy a server-image to a physical server is to integrate its resource into the openQRM environment. Adding new physical resource to openQRM is way easy. Turning on "network-boot" (PXE) in the servers BIOS and powering them on is enough for openQRM to auto-discover and add new servers. As explained in detail in the section "openQRM Boot concept" the started server will automatically boot via the network and appear as new, "idle" resource in openQRM's resource list.

Please note: **To initialize and start openQRM's network-boot environment the "dhcpd" and "tftpd" plugin needs to be enabled and started!**

Through openQRM's Virtualization Plug-ins virtual Machines are created and added to openQRM in the same way. Virtual Machines from different types are similarly created via the Virtualization type specific VM-Manager which allows to set various VM parameters such as the VM name, amount of memory, number of CPUs and its virtual network connection. The Virtual Machines boot sequence is then set to "netboot" (PXE) allowing them to get seamlessly added to openQRM in the same way as physical systems. They will appear in openQRM resource list as new, "idle" resource from the specific Virtual Machine type.

Generic Provisioning – openQRM's Appliance Model

All provisioning and deployment is controlled through openQRM's "Base" Menu section using an "appliance model". The appliance model turns the complex workflows of deploying a new system according a bunch of various configuration parameters, requirements and SLAs into one, single mouse click. Following the basic rules of an appliance in openQRM:

1) Appliances are providing the Datacenter services

2) An appliance consist of

- An Operating System Kernel
- A Server-Image containing a root-filesystem
- A Resource, acting as the Service container
- Additional configuration parameters defining service requirements and service level agreements

3) Every component of an appliance can be transparently exchanged/replaced

4) Each component can be managed separately without any further dependencies

4) Appliances are started or stopped by a single action

5) Plug-in Hooks in openQRM taking care to setup and configure all involved appliance subsystems

This appliance model is a central concept in openQRM's framework and all integrated subsystems are unexceptionally following it.

Virtualization Host Deployment

Virtualization Hosts (Hypervisors) consisting of physical systems which are being parted into several autonomous virtual machines. openQRM deploys Virtualization Hosts via its generic provision concept using the appliance model. To enable management of a Virtualization, Host the Hosts-appliance resource-type needs to be set to the specific Virtualization technology Host type.

Virtual Machine Deployment

Virtual Machines in openQRM are network-booted, deployed and managed in the same way as all other resources.

Please notice:
There are several Virtualization plugins available which bypass the network-boot and support local-deployment and local-boot for Virtual Machines. Examples for those kinds of Plug-ins are the “kvm-storage” and the “xen-storage” Plug-in which are emulating the “idle” state of its virtual resources and using openQRM's hook for virtual-resource- commands. The virtual-resource-command hook provides a method to map directly executed resource commands such as reboot to its virtual complement which is then executed on the virtual resources host (more details in the development section).

Those Plug-ins adding support for deployment and management of Operating Systems which do not support rapid deployment via network-boot such as Windows and Solaris/open Solaris within Virtual Machines.

Storage Layer

Since openQRM's rapid deployment methods are based on centralized Storage Systems they are a key componentin the openQRM management network. The Storage layer in openQRM is providing the remote, (server-) “image location”. Depending on the Storage type this “image-location” can be an NFS-export, an iSCSI Lun, an AOE Volume, any kind of remote block-device or anything else which contains a valid root-filesystem content.

Similar to Appliances, Storages in openQRM consist of a resource which is already integrated and available in openQRM. Therefore, the first thing to do to create a new Storage in openQRM is to add its resource. This can be done by deploying a storage server-image to an existing “idle” resource. In cases an already existing Storage Servershould be used for openQRM deployment its resource easily can be integrated through the “local-server” Plug-in.

Some of the supported commercial Storage Vendors offering kind of Storage Appliances which are closed system not allowing to be directly integrated as monitored resource within openQRM. Examples for that kind of Storage Providers are e.g., NetApp and EqualLogic. For those Plug-ins the specific Storage resource should be manually created to the “resource → new” Web form.

Please notice:

All resources manual added through the “resource → new” Web form will be automatically excluded from openQRM basic monitoring. They will always appear in “unknown” state (yellow icon) in openQRM since it is not possible (or not supported) to run additional third-party components, like the openQRM-Client, on those storage appliances.

Pluggable Storage Types

To enable support for infinite different Storage Technologies in openQRM the Storage types are Plug-able. New Storage types are added to openQRM via its Storage Plug-ins which also provide the defined interface for managingthe specific Storage Vendor.

Here a list of different Storage Plug-ins in openQRM:

- **aoe-storage**

This Plug-in provides an AOE-Storage- and deployment type supporting too boot resources directly from an AOE Storage-Server. It also provides the Storage-Server part which turns a simply Linux box + AOE- and Vblade-Tools installed into an AOE Storage-Server fully managed by openQRM.

- **equallogic-storage**

The netapp-storage Plug-in interfaces with NetApp Filer Appliances allowing resources to directly boot from the NetApp iSCSI Luns. Same as for all Storage Plug-ins it includes an embedded management for the NetApp Filer.

- **nfs-storage**

The nfs-storage Plug-in is almost the simplest Storage Type supported by openQRM. An exported directory includinga server's root-file-system content is enough for This Plug-in integrates Dell's Equallogic Storage Appliance into the openQRM framework. It supports seamless management of Equallogic Storage Systems and the capability to boot resources directly from the EqualLogic's iSCSI Luns.

- **netapp-storage**

openQRM to enable resource booting directly from a NFS Storage- Server. Server-images on NFS (NFSROOT) providing a generic transfer layer so they are a very good candidate for server-image-templates which are then automatically transferred to e.g., iSCSI Luns, AOE Volumes or even to local block-devices.

- **local-storage**

The local-storage is an exceptional Storage Plug-in in openQRM. It consist of a NFS Storage-Server with an underlaying Logical Volume Management (LVM) taking care of the supports for rapid cloning. Its deployment mechanism provides a "grab", "deploy" and "un-deploy" life-cycle for resources. Its "grab" stage transfers the content of all attached Hard disks of a resource to the remote NFS Storage location using a binary image format (dd). The "deploy" stage of the local-storage Plug-in re-transmits those binary server-images back to the local Hard disks of the same or other resources. With its "un-deploy" stage which automatically updates the server-image on the Storage in case the involved appliance is stopped it keeps a 1:1 relation between the specific server-image on the remote Storage and the local Hard disk of a specific resource deployed via an appliance.

Please notice:

The local-storage Plug-ins life-cycle will re-set the involved boot-sequence after the "deploy" stage to "local-boot" instead of the regular "netboot". After un-deployment it automatically set the boot-sequence to "netboot" again.

- **iscsi-storage**

This Plug-in integrates with the Enterprise iSCSI Target, an Open-Source implementation which is freely available in many modern Linux Distributions. It automatically manages the iSCSI Targets configuration and exported block- devices plus it supports to boot resources directly from those iSCSI Luns.

- **lvm-storage**

The lvm-storage Plug-in is a combination of the aoe-storage, the iscsi-storage and the nfs-storage Plug-in featuring an underlaying Logical Volume Management (LVM) to support rapid cloning and snapshots. It includes an embedded management console for the LVM volumes within the openQRM UI.

- **xen-storage and kvm-storage**

Those Plug-ins providing a Virtualization- (Xen, KVM) and Storage-part (LVM) within a single Plug-in. Its Virtualization part is using the local (local to the Virtualization Host) LVM volumes to attach them as local (local to the Virtual Machine) Hard disk. Its Storage part includes full administration capabilities for the Storage Servers LVM volumes. The Plug-ins also following strictly using openQRM's appliance model but, other than the other Storage types basedon direct network-boot deployment, they are booting their Virtual Machines from their local attached, virtual Hard disks. Independent from openQRM usual network-deployment those Plug-ins providing the support for Non-Linux Operating Systems such as Windows and Solaris/open Solaris in openQRM.

Please notice:

This results in a dependency to “local-disk” devices on the Storage Host. That means that VMs from this type must be located on the same Storage host where the logical volume (the VMs root-disk) is located

- **tmpfs-storage**

The tmpfs Storage Plug-in provides the possibility to deploy (Linux-) System “in-Memory”. Depending on the image- configuration it creates a tmpfs-root-mountpoint which is being populated via “install-from-nfs”.

- **sanboot-storage**

Sanboot-storage integrates GPXE (etherboot.org [http://etherboot.org]) into openQRM as a second, optional network- bootloader. It supports an image-based Windows-deployment. Using Sanboot Windows systems are directly from an iSCSI Target or AOE-Storage.

- **zfs-storage**

The ZFS Storage Plug-in allows to benefit from Sun's new robust ZFS file-system provided by Solaris, open Solaris or FreeBSD systems. The ZFS file-system is known for its awesome features such as high storage capacities, integration of the concepts of filesystem and volume management, snapshots and copy-on-write clones, continuous integrity checking and automatic repair, RAID-Z and native NFSv4 ACLs. The ZFS Storage Plug-in in openQRM depends on the “solx86” Plug-in which integrates Solaris, open Solaris and FreeBSD systems seamlessly into the openQRM framework. It supports booting resources directly from the remote ZFS file-system through the iSCSI protocol and also provides the storage management capabilities embedded in the openQRM UI.

Storage Management

openQRM unifies and automates the administration of the different Storage Providers by its integrated Storage Management. Therefore, during the design phase of openQRM, the openQRM team especially analyzed the frequent and common storage actions required in the openQRM environment. Those are:

- Create a new Volume with a given name and Size
- Remove a Volume with a given name
- Clone a Volume with a given name (creates a 1:1 copy)
- Authenticate a Volume against a resource

If supported by the Storage Technology there are also the following Storage action possible:

- Snapshot a Volume with a given snapshot name and snapshot size
- Resize a Volume with a given name and a new size

All those Storage actions are implemented by their specific Storage Plug-in and exposed through a Plug-in specific Volume manager. While the Create, Remove, Clone, Snapshot and Resize Storage actions are executed by the sysadmin the authenticate action is handled fully automatically by openQRM via a Storage-Auth-Hook as describedin the following chapter.

Storage actions are also integrated within the openQRM Cloud to automate the Storage management for rapid deployment through the Cloud Portal.

Storage Authentication

openQRM's rapid deployment is all about centralized server-management through network-booting and directly attached remote root-file-system storage. To secure the storage- and management-network and to ensure only the resource dedicated for a specific appliance is allowed to access and mount a remote (server-) image-location from an external Storage Server openQRM automatically takes care to authenticate the resource against its image-location. This happens through a Storage-Auth-Hook provided by the Storage Plug-ins. According to the appliance image- deployment type this hook automatically gets executed in the openQRM Server engine when an appliance is starting or stopping. The hook includes all required parameters and information’s about the appliance which is then used by the specific Storage Plugin to enable or disable access to an image-location on the Storage Server.

A detailed functions description of the Storge-Auth-hook is available in the Plug-in Development section.

Virtualization Management

In openQRM Virtualization Hosts are managed through the appliance model. The specific resource-type in the appliance configuration tells openQRM which Virtualization interface type to use. For this reason, the Virtualization Host needs to be integrated and available in openQRM. This can be done by deploying a Virtualization Host server-image to an existing “idle” resource. In cases an already existing Virtualization Host should be used its resource easily can be integrated through the “local-server” Plug-in.

Pluggable Virtualization Types

Not only the storage types but also the virtualization types are fully plug-able in openQRM. Via its open plugin API openQRM integrates with VMware-ESX, Vmware-Server 2, Vmware-server (1), Xen, KVM and Citrix XenServer. Adding support for further virtualization technologies like Virtualbox, openVZ and others is on the future road map. To be able to seamlessly handle all those different kinds of virtual machines openQRM puts a layer on top of the virtualization methods to unify their management. In openQRM virtual machines are simply net-booting into the openQRM management environment in the same way as physical systems.

Hypervisors are just Resource Providers

Continuing with the full separation between hard- and software, meaning on one side physical- and also virtual machines (because the Vms are running on the Hypervisor which is running on the bare metal) and on the other sidethe software layer, the server-images located on a safe storage device, in an openQRM environment a Hypervisor becomes “just” a resource provider, just being responsible to host the virtual compute resource of the user’s choice. That way the appliance running on a virtual machine also gets fully independent from its Hypervisor Host and can be transparently (live-) migrated to another Hypervisors of the same or different virtualization technology or even from physical systems to virtual machines and the other way around. OpenQRM supports P2V, V2P, V2V, P2P migration without any changes on the actual server-images itself.
OpenQRM cannot only manage different types of Hypervisor technologies but it can also deploy them via the regular generic deployment mechanism of its framework. That offers scalability for the complete IT infrastructure because the data-center can grow (and shrink) as demanded by just adding (or removing) more Hypervisors

Which Virtualization Technology fits best (for my application)?

Since each Virtualization Technology has different advantages (and eventual disadvantages) it should be selected depending on which application services should be virtualized. That means for different purposes one should choose the Virtualization type fitting best to the applications. It also means that in a “perfect world” system administrator ending up with managing lots of different Virtual Machine Types and Technologies plus the physical system acting as Virtual Machine hosts. By abstracting physical and virtual machines as Data center “resources” openQRM is the perfect tool for the “perfect world” scenario since it separates the actual service (OS + applications) from the physical or virtual machine (the “resource”) it is running on and therefore allows the administrator to easily and dynamically adapt the “resource type” (the Virtualization type) of services according to the service demands.

This unique abstraction of the different Virtualization Technologies in openQRM also has another huge advantage:

openQRM avoids and prevents Vendor locking!

With openQRM switching from one Technology to another is absolute transparent and does not affect the actualservice (the server-image) at all.

Server Image Management

Mostly all rapid deployment methods in openQRM taking advantages of modern Storage Server and focusing on an “image-based” deployment. That means that systems are not installed by openQRM, which of course can be done with openQRM in a fully automated way using the LinuxCOE Plug-in integration, but systems are “just” connected through the network to ready-to-run server-images on remote Storage server. Here a few of the great advantages of the image-based deployment compared to regular provisioning via (manual) local-installation.

- Server-images are based on well-known-to-work server-templates, working and pre-configured out-of-the-box
- Server-images saves the installation step and time. Deployment time is equal Boot time.
- Server-images can be efficiently snapshotted. Snapshots just store the changes between the origin image
- Server-images can be cloned-on-demand. Even snapshots can be deployed seamlessly
- Server-images guarantee a reproducible IT-environment
- Server-images can easily be backed-up and restored
- Server-images provides a single-place for updates
- Server-images can be re-visioned. Think of it like “subversion for your servers”
- Server-images are just files. Treat them like files.
- and much more

Since openQRM supports a whole bunch of different Storage types it provides a logical abstraction which makes itreally easy to use the different server-images types.

An image in openQRM consist of different configuration parameters such as its Storage Server, its root-device identifier and several other deployment parameters. The two most important parameters are on which Storage Server the image is located plus a storage-type dependent root-device identifier providing the exact information where the image is located on the Storage.

On NFS-based Storage Server this root-device identifier is the exported path to the image root-file-system, on iSCSI-based Storages it is the Lun ID and on AOE/Coraid based Storage devices it is the shelf and slot number of the exposed volume. Since the root-device identifier it is also a plug-able component provided by the Storage Plug-ins.
When creating a new image openQRM first ask on which Storage server its image-location is located. After selecting the Storage Server openQRM contacts the Storage and triggers it to send its list of current available root-device identifiers. This list is then used to populate the root-identifier select-box in the “image → new” form.

Please find a detailed description on how the root-identifier hooks is used in the development section.

Rapid deployment in openQRM is based on network-boot and resources are always passing the initrd stage taking care to mount the server-image's root-file-system content from the (remote) Storage Server. Within this stage the resource enjoys full network connection plus the capability to download additional tools provided by Plug-ins. The initrd stage also provides some unique, great ways to transfer root-file-system content from one Storage provider to another.

One thing which all transfer methods have in common is that they require an NFS-based Storage system either as source or destination. Therefore, it is an advantage to have at least one NFS-based Storage Server acting as a store for “golden images” (server-image templates).

Please notice:

All of the below listed server-image transfer methods are configured as “single shot” meaning the transfer- configuration is being resetted after the first deploy.

Following a description of the available transfer methods.

Installing from NFS

The “install from NFS” transfer method is configured through the “image → new” or “image → edit” form via the “install- from-nfs” select box. This box provides a list of all NFS-based server-images available in openQRM.

A typical Use-case is populating an empty iSCSI Lun which works as explained by the following step-by-step list:

- Create a new (empty) iSCSI Lun on an available Storage Server
- Create a new image using the (still empty) iSCSI Lun as its root-device identifier
- In the new Image configuration select an available NFS-based server-image from the “install-from-nfs” selectbox
- Create a new appliance using an available resource, a kernel and the just created image
- Start the appliance

The following happens within the initrd stage of the starting appliance:

- The resource discovers the iSCSI Target (the Storage Server of the image)
- The resource connects the (still empty) iSCSI Luns as its root-file-system
- The resource mounts the “install-from-nfs” image-location in a temporary location
- Then the resource copies the root-file-system content from the temporary mount point (the NFS-based server-image template selected by the “install-from-nfs” select box) to its root-device (the iSCSI Lun) using rsync.
- When the transfer completed the resource umounts the temporary location and continues regular boot-up and init from the, now populated iSCSI Lun.

Transfer to NFS

The “transfer to NFS” transfer method is configured through the “image → new” or “image → edit” form via the“transfer-to-nfs” select box. This box provides a list of all NFS-based server-images available in openQRM.

A typical Use-case is creating a new server-image template from an existing server-image located on an Lun. Thisworks as explained by the following step-by-step list:

- Create a new (empty) NFS export on an available NFS-based Storage Server
- Create a new image using the (still empty) NFS export as its root-device identifier
- In the new Image configuration of an existing iSCSI-based image select the just created (still empty) NFS-based server-image from the “transfer-to-nfs” select box
- Create a new appliance using an available resource, a kernel and the iSCSI-based image
- Start the appliance

The following happens within the initrd stage of the starting appliance:

- The resource discovers the iSCSI Target (the Storage Server of the image)
- The resource connects the iSCSI Luns as its root-file-system

- The resource mounts the “transfer-to-nfs” image-location in a temporary location
- Then the resource copies the root-file-system content from the mounted iSCSI LUN to the temporary mountpoint (the NFS-based server-image template selected by the “transfer-to-nfs” select box) by rsync.
- When the transfer completed the resource umounts the temporary location and continues regular boot-up andinit from the iSCSI Lun.
- Result is a new, fully populated NFS-based server-image containing the root-file-system content of theexisting iSCSI-based image.

Installing from Local

The “install from Local” transfer method is configured through the “image → new” or “image → edit” form via the “install-from-local” select box. This box provides a list of common local root-devices on resources.

A typical Use-case is populating an empty iSCSI Lun which works as explained by the following step-by-step list:

- Create a new (empty) iSCSI Lun on an available Storage Server
- Create a new image using the (still empty) iSCSI Lun as its root-device identifier
- In the new Image configuration select a root-device from the “install-from-nfs” select box
- Create a new appliance using an available resource (one with an existing root-file-system content on theroot-device as configured by the install-from-local select box), a kernel and the just created image
- Start the appliance

The following happens within the initrd stage of the starting appliance:

- The resource discovers the iSCSI Target (the Storage Server of the image)
- The resource connects the (still empty) iSCSI Luns as its root-file-system
- The resource mounts its local root-device (defined by the “install-from-local” definition) i n a temporary location
- Then the resource copies the root-file-system content from the temporary mount point (the local device) to its root-device (the iSCSI Lun) using rsync.
- When the transfer completed the resource umounts the temporary location and continues regular boot-up andinit from the, now populated iSCSI Lun.

Transfer to Local

The “transfer to Local” transfer method is configured through the “image → new” or “image → edit” form via the “transfer-to-nfs” select box. This box provides a list of common local root-devices.

A typical Use-case is deploying a network-booted appliance with its root-file-system located on a partition on the local Hard disk. This works as explained by the following step-by-step list:

- Edit an existing (fully populated) NFS-based server-image and set its “transfer-to-local” select box to a localroot-device
- Deploy this image on a resource via an appliance

The following happens during the initrd stage:

- The resource mounts it root-file-system by NFS
- The resource mounts the local device configured by the “transfer-to-local” image configuration in a temporary location
- Then the resource copies the root-file-system content (the NFS root-file-system content) to the temporarymount point (the local device selected by the “transfer-to-local” select box) using rsync.
- When the transfer completed the resource umounts the temporary location and continues regular boot-up andinit

from its NFSROOT root-file-system

To start this kind of server-image from its resource local Hard disk partition follow the steps below:

- Create a Storage from the type “local installed server” using the openQRM server as the resource
- Create a new image, select the “local installed server” Storage as its storage provider
- Fill in the custom root-device from which a resource should boot (the one populated by “transfer-to-local)
- Create a new appliance using an available resource, a kernel and the just created “local installed server”image
- Start the appliance

The resource will reboot, network boot kernel and initrd and startup init on its local root-device as defined in the image definition.

Remote Deployment

All rapid deployment in openQRM is done via network-boot (PXE) using a special openQRM initrd. As described in detail in the section about “openQRM boot sequence” the openQRM initrds are plug-able and allowing a deployment type (created by a Storage Plug-in) to tell the booting system from “where” and “how” it should mount its root-file- system from.

For all remote deployment the usage of local Hard disks of the resources is avoided expect for Swap-Space. Swap-Space available on local attached Hard disks is automatically detected and used by the resources.

Local Deployment

Local deployment in openQRM is also done via network-boot (PXE) using a special openQRM initrd and comes in three flavors:

- Local-Storage Deployment
- Local-Server Deployment
- Automatic Installation

Local-Storage Deployment

(described in the Storage section)

Local-Server Deployment

openQRM also supports a local deployment method which allows to centralized, network-boot the resource and let it mount and use a local Hard disk partition for further boot-up. Basically, it allows to define the local boot-device via a network-bootloader.

HINT: The local Hard disk partition defined for boot-up in the image-configuration can be empty if used in combinationwith “install- from-nfs” to automatically populate the partition with root-file-system content.

Automatic Installation

The third flavor of local deployment in openQRM is “automatic profile-based installation” via the integration with LinuxCOE [http://linuxcoe.sourceforge.net/]. This mechanism automatically installs systems on their local disk. Those installation then can be used for either direct deployment through “local-server deployment” or as server-templates for server-image creation.

P2V – Physical to Virtual Migration

Since resources are decoupled from their root-files-systems migration appliances from physical server to virtual machines is complete transparent and easy. Follow the steps below to exchange an appliance physical resource with a virtual one:

- stop the appliance

- edit the appliance
    - change the “resource-type” from “Physical System” to a “Virtualization-type VM”
    - select a new resource from the type “Virtualization-type”
    - save the appliance
- start the appliance

The appliance will now start using the new, virtual resource as defined in the appliance configuration.

V2P – Virtual to Physical Migration

Similar to the P2V migration here the steps to migrate an appliance from a virtual machine to a physical system:

- stop the appliance
- edit the appliance
    - change the “resource-type” to “Physical System”
    - select a new resource from the type “Physical System”
    - save the appliance
- start the appliance

The appliance now runs on a physical system.

V2V – Virtual (type A) to Virtual (type B) Migration

Using the same method appliances can be moved from one Virtualization type to another. Here the steps necessary for the migration:

- stop the appliance (Virtualization type A)
- edit the appliance
    - change the appliance resource type from “Virtualization-VM” (type A) to “Virtualization-VM” (type B)
    - select a new resource from “Virtualization-VM” (type B)
    - save the appliance
- start the appliance

The appliance will now start using the new, virtual resource (Virtualization type B) as defined in the appliance configuration.

Service High-Available

HA on Resource Level

When talking about “Green IT” the current approach is to use Virtualization to consolidate “many” physical servers to run virtualized on “one” (or more) Hypervisor Hosts. While this method is good to save the overall power consumption it is often forgotten that the in the new, virtualized situation in the case the “one” Hypervisor Hosts breaks those “many” servers running in virtual machines on this Hosts will also get unavailable. This means that in the modern, virtualized Worlds we need to especially take care of the high availability.

The usual method to keep system High available (10 custom servers):

- Get additional 10 servers preferred of the same manufacture consisting of the same parts
- Configure syncing of the disks between the 10 pairs of servers

- Implement a Fail-over solution for the service running on the 10 clusters

As a result, this method requires 20 physical systems to keep 10 servers high available.

HA in openQRM:

- Deploy the 10 custom servers via openQRM
- Add 1 server as Hot-Standby

In the case one of the 10 custom servers break openQRM will use the one available system to restart it via its rapid deployment methods. As the result 10 (or more) servers can made high available with just a single Hot-Standby system (since resource types are not linked to the actual server-image in openQRM physical servers even could fail-over to virtual machines).

This saves the power consumption of 9 servers!

HA on Application Level

Application HA can be archived in openQRM by adding custom Nagios checks to monitor the service state of a specific application. In case this check fails it can do various things to re-activate the service again:

- Restarting the service
- Forcing a reboot
- Forcing a fail-over to a passive application-standby

HA on Application level is not yet automated in openQRM but must be setup manual.

HA for the openQRM server

To avoid a “single point of failure” (SPOF) and to keep the openQRM Server high-available it is recommended for a production setup to run openQRM in an “active/passive” HA-configuration. openQRM's architecture of keeping everything within a single-base-directory makes it really easy to install openQRM in High-available mode.

The openQRM Team recommends to use Linux-HA [http://www.linux-ha.org/] for the openQRM HA setup. **Requirements for openQRM HA**

- 2 or more systems for the active and passive openQRM Cluster nodes
- Shared-Storage providing the openQRM Server file-data
- External (remote) Database

HINT: openQRM does not matter if installed on physical system or on a virtual machine.**Steps to install openQRM in HA-mode**

- Install OS on the systems dedicated for the openQRM Cluster nodes
- Mount the shared storage in openQRM's base-dir (normally /usr/share/openqrm) on all Cluster nodes
- Setup Linux-HA with a Cluster ip-address
- Install openQRM on ONE system only!
    - During setup use the External (remote) Database and the Network-interface with the Clusterip-address
- On the rest of the system
    - link /usr/share/openqrm/web/ to DOCUMENT_ROOT/openqrm
    - link /usr/share/openqrm/etc/init.d/openqrm to /etc/init.d/openqrm

- Wrap /etc/init.d/openqrm into a Linux-HA init script

After that Linux-HA will take care to always keep one system running as the active Cluster node using the global Cluster ip-address.

Plugin description

aoe-storage

The Aoe-storage plugin integrates Aoe/Coraid Storage into openQRM. It adds a new storage-type 'aoe-storage' anda new deployment-type 'aoe-root' to the openQRM-server during initialization.

Aoe-storage type: A linux-box (resource) with 'vblade' installed should be used to create a new Storage-server through the openQRM-GUI. The Aoe-storage system can be either deployed via openQRM or integrated into openQRM with the 'local-server' plugin. openQRM then automatically manages the vblade disks on the Aoe-storage server.

Aoe-deployment type: The Aoe-deployment type supports to boot servers/resources from the Aoe-storage server. Server images created with the 'aoe-root' deployment type are stored on Storage-server from the storage-server type 'aoe-storage'. During startup of an appliance, they are directly attached to the resource as its rootfs via the aoe-protocol.

How to use:

- Create an Aoe-storage server via the 'Storage-Admin' (Storage menu)
- Create a Disk-shelf on the Aoe-storage using the 'Shelfs' link (Aoe-plugin menu)
- Create an (Aoe-) Image ('Add Image' in the Image-overview). Then select the Aoe-storage server and selectan Aoe-device name as the image root-device.
- Create an Appliance using one of the available kernels and the Aoe-Image created in the previous steps.Start
- the Appliance

---

aws

The aws-plugin provides an integration with the Amazon WebService (AWS) and provides a seamless migration-path “from” and “to” AWS.

Configure AWS Account Create a new AWS Account configuration using the “AWS Accounts” menu item. Thefollowing information’s are required:

- AWS Account NameJava Home Dir
- EC2 Home Dir
- AWS Private key fileAWS Cert file
- SSH key file used for the AMIAWS Region
-

Import Servers from AWS To import an AWS Server (→ the AMI of an active EC2 Instance) follow the steps below:

- Select an AWS Account to use for the import
- Select an active AWS EC2 Instance running the AMI to import
- Select an (empty) openQRM Server image (from type NFS- or LVM-NFS)

This will automatically import the AMI from the selected AWS EC2 Instance into the (previously created) empty Server Image in openQRM.

The imported AMI now can be used with all existing “resource-types” in openQRM so e.g., it can now also run on a physical system or on any other virtualization type.

citrix

The Citrix plugin is tested with Citrix XenServer 5.5.0How to use:

- install Citrix-XenServer on a server system
- also install the second cd containing the support for Linux Vms
- login to the Citrix XenServer via ssh and scp the /usr/sbin/xe util to the openQRM servers /usr/sbin direnable the
- openQRM Citrix plugin via the plugin manager
- manually add a resource via “Base → Resources → New”, provide the Citrix servers mac- and ip-address create
- a storage type “Local-installed server” via “Base → Storage → New”, select the Citrix server resourceand provide a name
- create an image via “Base → Image → New”, provide a name
- create an appliance via “Base → Appliances → New”, select the Citrix servers’ resource, the default kernel and the previously created image
- set the appliance “Resource type” to “Citrix Host” and save
- go to “Plugins → Virtualization → Citrix → Citrix VM Manager”, select the Citrix Host appliancenow click on “auth”
- and provide the authentication details to login to the Citrix Host
- create a new vm via the ”+VM” button
- the new vm will boot-up via the network and in a short time appear in the resource overview as a new idle resource in the data-center

cloud

The openQRM cloud-plugin provides a fully automated request and provisioning deployment-cycle. External data-center users can submit their Cloud requests for systems via a second web-portal on the openQRM-server. After either manually or automatic approval of the Cloud requests openQRM handles the provisioning and deployment fully automatically.

How to use:

To setup automatic deployment with the cloud-plugin first the openQRM environment needs to be populated with available resources, kernels and server-images. The combination of those objects will be the base of the cloud- requests later.

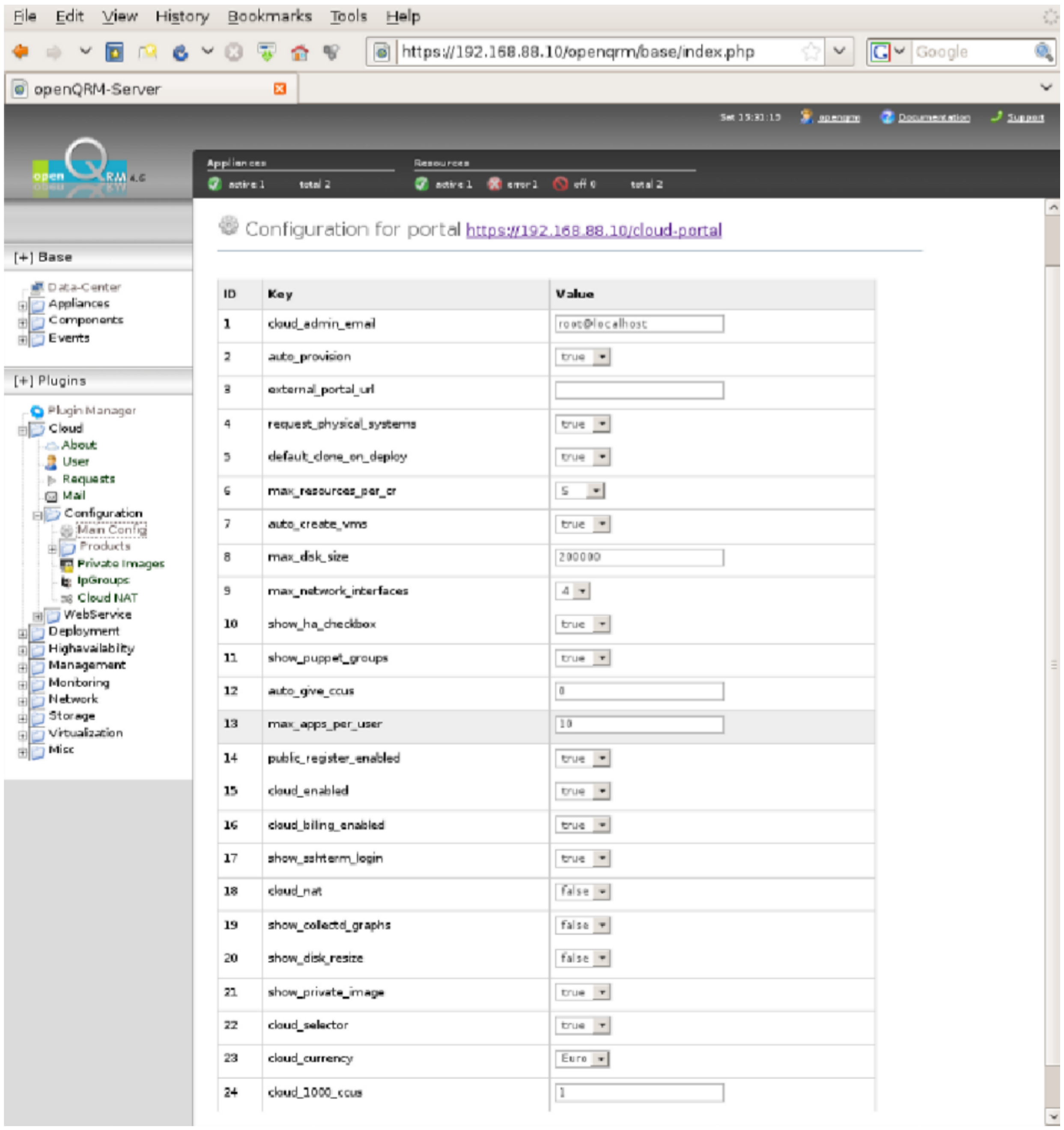
- Start some resources (physical and/or virtual)Create one (or more) storage-server
- Create one (or more) server-image on the storage-serversCloud-Users can be created in 2 different ways:
- User can go to http://openqrm-server-ip/cloud-portal [http://openqrm-server-ip/cloud-portal] and registerthemselves
- Administrators of openQRM can create Users within the Cloud-plugin UI

Cloud-Requests can be submitted to the openQRM Cloud either via the external Cloud-portal by a logged in user or on behalf of an existing user in the Cloud-Request manager in the openQRM UI.

- start time - When the requested systems should be available
- stop time - When the requested systems are not needed any moreKernel - Selects the kernel for the requested
- system
- Image - Selects the server-image for the requested system
- Resource Type - What kind of system should be deployed (physical or virtual)Memory - How much memory the
- requested system should have

- CPUs - How many CPUs the requested system should have
- Disk - In case of Clone-on-deploy how much disk space should be reserved for the userNetwork Cards - How
- many network-cards (and ip-addresses) should be available High available - Sets if the requested system should be
- high-available
- Clone-on-deploy - If selected openQRM creates a clone of the selected server-image before deployment

Cloud Configuration



Configuration for portal https://192.168.88.10/cloud-portal

ID	Key	Value
1	cloud_admin_email	root@localhost
2	auto_provision	true
3	external_portal_url	
4	request_physical_systems	true
5	default_clone_on_deploy	true
6	max_resources_per_cr	5
7	auto_create_vms	true
8	max_disk_size	200000
9	max_network_interfaces	4
10	show_ha_checkbox	true
11	show_puppet_groups	true
12	auto_give_ccus	0
13	max_apps_per_user	10
14	public_register_enabled	true
15	cloud_enabled	true
16	cloud_billing_enabled	true
17	show_sshterm_login	true
18	cloud_nat	false
19	show_collectd_graphs	false
20	show_disk_resize	false
21	show_private_image	true
22	cloud_selector	true
23	cloud_currency	Euro
24	cloud_1000_ccus	1

Explanation of the configuration parameters:

1. cloud_admin_email - The email address of the Cloud-Administrator
2. auto_provision - Can be set to true or false. If set to false requests needs manual approval.
3. external_portal_url - Can be set to the external Url of the Cloud-portal
4. request_physical_systems - If the Cloud should provide also physical system to the users
5. default_clone_on_deploy - Set to true (default) the Cloud deploys “clones” of the selected server-image
6. max_resources_per_cr - Global-Cloud-Limit, sets the max. number of resources per request
7. auto_create_vms - If the Cloud should automatically create new virtual machines
8. max_disk_size - Global-Cloud-Limit, sets the max. disk usage on the Cloud
9. max_network_interfaces - Global-Cloud-Limit, sets the max. number of network-interfaces

10. show_ha_checkbox - If to show the High-availability options to the users
11. show_puppet_groups - If to show the Puppet application-classes to the users
12. auto_give_ccus - Number of CCUs automatically given to new users
13. max_apps_per_user - Global-Cloud-Limit, sets the max. number active appliances per user
14. public_register_enabled - If the public registration to the Cloud is enabled or disabled.
15. cloud_enabled - Global switch to enable or disable the Cloud. Sets the Cloud in maintenance if set to false
16. cloud_billing_enabled - Switch to enable or disable the automatic Cloud billing
17. show_sshterm_login - If to show the ssh-term login option to the users
18. cloud_nat - If the Cloud is using Natural Address Translation (NAT)
19. show_collectd_graphs - If to show the Collectd System graphs to the users
20. show_disk_resize - If to show the Disk-resize option to the users
21. show_private_image - Enables or disables the Private-Image feature
22. cloud_selector - Enables or disables the Cloud-Selector feature
23. cloud_currency - The currency used by the Cloud (e.g., Euro, US, etc.)
24. cloud_1000_ccus - Defines the Mapping between the real currency (Euro, US, etc.) and virtual one (CCU)

Cloud Ip Groups

The openQRM cloud-plugin provides automatically network-configuration for the external interfaces of the deployed systems. To create and populate a Cloud Ip Group please follow the steps below:

- Select the Cloud Ip Group link from the cloud-plugin menu
- Click on 'Create new Cloud Ip Group' link and fill out the network parameters for the new Ip Groupin the Ip Group
- overview now select the new created Ip Group and click on the 'load-ips' button Now put a block of ip-addresses
- for this Ip Group into the textarea and submit.

Cloud Admin
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 API documentation of the openQRM Cloud SOAP WebService can be found at :

http://www.openqrm-ng.net/downloads/plugins/cloud/openqrm-soap-api/ [http://www.openqrm-ng.net/downloads/plugins /cloud/openqrm-soap-api/]

Cloud Lockfile

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

collectd

The Collectd plugin provides automated monitoring and system graphs for appliances in openQRM. 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 appliance to collect enough data to create the graphs.

dhcpd

The dhcpd-plugin automatically manages your ip-address assignment and network-boot environment for the rapid- deployment features of openQRM. Since 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 this plugin.

How to use:

No manual configuration is needed for the dhcpd-plugin. It automatically configures a dhcpd.conf file during initialization. To manual add resources for static ip-assignment please find the dhcpd.conf used by the plugin at:

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

---

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-sever during start/stop ofan appliance. The hostnames are automatically set to the appliance name with the ip address of the appliance- resource.

How to use:

No manual configuration is needed for the dns-plugin. It automatically configures the dns-name server during initialization with the domain name configured in:

```
 /usr/share/openqrm/plugins/dns/etc/openqrm-plugin-dns.conf
```

---

equallogic-storage

The Equallogic-storage plugin integrates Equallogic iSCSI Storage hardware into openQRM. It adds a new storage-type 'equallogic-storage' and a new deployment-type 'equallogic-root' to the openQRM-server during initialization.

Equallogic-storage type: An Equallogic-Server, added manually as a new resource with its ip-address set to the group IP, should be used to create a new Storage-server through the openQRM-GUI. openQRM then automatically manages the Equallogic-disks (Luns) on the Equallogic-storage server.

Equallogic-deployment type: The Equallogic-deployment type supports booting servers/resources from the Equallogic-storage server. Server images created with the 'equallogic-root' deployment type are stored on Storage- server from the storage-server type 'equallogic-storage'. During startup of an appliance, they are directly attached to the resource as its rootfs via the iSCSI-protocol.

Usage philosophy: Volumes on the Equallogic hardware are used as a block device over iSCSI, on which OpenQRM will create a filesystem (no partitions). This filesystem is mounted through the rootfs deployment scripts. It is recommended to use a separate network or VLAN purely for iSCSI, to which the Equallogic network interfaces are connected. When booting directly from the Equallogic volumes, the current implementation also requires a DHCP server on this network for IP allocation on the second (storage) network interface of (cloud) appliances; during boot, the rootfs hook will use udhcpd to set an IP on the secondary interface.

Current limitations:

- Snapshotting is not implemented; e.g., no support for using a volume as “master” image and using snapshotsof that as rootfs volumes, as with lvm-iscsi-storage.
- Clone-on-deploy in the cloud is not actually cloning volumes; instead, new volumes are made on the storagewhich will be formatted when using install-from-nfs deployment. Private images does use cloning.
- The clone function in the Equallogic storage manager is exactly that; it runs the “clone” command on the hardware.
- Only alphanumeric characters, colon, dot and dash are allowed in volume names.

How to use:

- Enable SSH access on your Equallogic storage group
- Create an Equallogic-storage server via the 'Storage-Admin' (Storage menu)
- Create a volume on the Equallogic-storage using the 'Volume Admin' link (Equallogic-plugin menu)
- Create an (Equallogic-) Image ('Add Image' in the Image-overview). Then select the Equallogic-storage serverand select an Equallogic-device name as the images root-device.
- Create an Appliance using one of the available kernels and the Equallogic-Image created in the previoussteps.
- Start the Appliance

Equallogic emulator: For pre-production and without-hardware testing, a wrapper script called eqemu-scst has been written. This script can be used to turn any linux server with the SCST + SCST-iSCSI initiator stack to emulate the behavior of an Equallogic group. It will need to be adapted to your environment and is only included for testing and development purposes. Can be found in the OpenQRM source at plugins/equallogic-storage/bin/eqemu-scst

---

high availability

The highavailability-plugin automatically provides high-availability for the appliances managed by openQRM.How

to use:

Simply use the HA-Manager to select the appliances which should be high-available. In case of an error openQRMwill try to find a new resource fitting to the appliance profile and re-start/re-deploy the appliance.

---

image-shelf

The image-shelf-plugin provides ready-made Server-Images templates for various purposes. Those Server-Image templates are transparently transferred to 'empty' Images located on Storage-Servers managed by openQRM. After that they can be directly used for rapid-deployment. This is the easiest method to get started.

Please notice that the Image-Shelfs are providing NFS-deployment Server-Image templates which then can be transferred to e.g., Iscsi- or Aoe-deployment Images via the INSTALL_FROM deployment parameters.

- How to use:
- Enable the 'nfs-storage' plugin
- Create a Storage-server from the type 'NFS-Storage' or 'Lvm Storage Server (Nfs)'

(You can use the openQRM-server itself as resource)

- Create a new export on the NFS-Storage server via the 'nfs-storage' or 'lvm-storage' pluginCreate a
- new Image on the NFS-Storage server and select the previously created

NFS export (the storage-location for the image) as the Image's 'root-device'

- Now Click on the Image-shelf
- Select an Image-Shelf from the list
- Select a Server-Template from the list
- Select the just created (empty) NFS-Image
- Check the Event-list for the progress of the Image creation

---

iscsi-storage

The Iscsi-storage plugin integrates Iscsi-Target Storage into openQRM. It adds a new storage-type 'iscsi-storage' anda new deployment-type 'iscsi-root' to the openQRM-server during initialization.

Iscsi-storage type:

A linux-box (resource) with the Enterprise Iscsi-target installed should be used to create a new Storage-server through the openQRM-GUI. The Iscsi-storage system can be either deployed via openQRM or integrated into openQRM with the 'local-server' plugin. openQRM then automatically manages the Iscsi-disks (Luns) on the Iscsi-storage server.

Iscsi-deployment type:

The Iscsi-deployment type supports to boot servers/resources from the Iscsi-stoage server. Server images created with the 'iscsi-root' deployment type are stored on Storage-server from the storage-server type 'iscsi-storage'. During startup of an appliance, they are directly attached to the resource as its rootfs via the iscsi-protocol.

How to use:

- Create an Iscsi-storage server via the 'Storage-Admin' (Storage menu)
- Create a Disk-shelf on the Iscsi-storage using the 'Luns' link (Iscsi-plugin menu)
- Create an (Iscsi-) Image ('Add Image' in the Image-overview). Then select the Iscsi-storage server and selectan Iscsi-device name as the images root-device.
- Create an Appliance using one of the available kernels and the Iscsi-Image created in the previous steps.
- Start the Appliance

---

kvm

This plugin is tested with KVM kvm-62 To benefit from KVM's 'virtio' feature at least kvm-84 is needed

The KVM plugin adds support for KVM-Virtualization to openQRM. Appliances with the resource-type 'KVM Host' are listed in the KVM-Manager and can be managed via the openQRM GUI. Additional to the regular partition commandslike create/start/stop/remove the KVM-plugin provides a configuration form per vm to re-configure the partition as needed (e.g., adding a virtual network card or hard disks).

Hint: The openQRM-server itself can be used as a resource for an KVM-Host appliance. In this case network-bridging should be setup on openQRM-server system before installing openQRM. At least an “internal” bridge for the openQRM management network is needed. The name for this bridge can be configured in the KVM plugin configuration file via the parameter OPENQRM_PLUGIN_KVM_INTERNAL_BRIDGE.

Additionally, an external bridge (e.g., pointing to the internet) can be setup and configured via the OPENQRM_PLUGIN_KVM_EXTERNAL_BRIDGE parameter in the KVM plugin-configuration file. openQRM then will create every first (virtual) network-card for the KVM Vms on the internal bridge and every other on the external one. With this 2-bridge setup every vm will then have its first nic pointing to the openQRM management network (doing the pxe-boot) and every other nic will point e.g., to the internet.

After having a network-bridge configured openQRM should be installed on the internal bridge-interface (by default br0). This can be done by setting the openQRM management network-interface in /usr/share/openqrm/etc/openqrm-server.conf to br0 before initializing openQRM.

How to use:

- Create an appliance and set its resource-type to 'KVM Host'
- Use the 'VM Manager' in the Kvm-plugin menu to create a new Kvm-server virtual-machines on the Host
- The created Kvm-server vm is then booting into openQRM as regular resources

kvm-storage

The kvm-storage plug-in is further described in the section about “Windows deployment”

linuxcoe

The linuxcoe-plugin provides automatic installation for the systems managed by openQRM via LinuxCOE. It embeds the LinuxCOE web-interface and supports assignment of custom installation profiles to systems managed by openQRM. The automatic installed system then can be used to create server-templates.

More about LinuxCOE at: http://linuxcoe.sourceforge.net [http://linuxcoe.sourceforge.net/]

local-server

The local-server-plugin provides an integration for already existing, local-installed systems in openQRM. After integrating an existing, local-installed server it can be used 'grab' the systems root-fs and transform it to an openQRM server-image. It also allows to dynamically deploy network-booted server images while still being able to restore/restart the existing server-system located on the local-harddisk.

How to use:

- Copy (scp) the 'openqrm-local-server' util 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
```

- Execute the 'openqrm-local-server' util on the remote system via ssh e.g.:

```
ssh [ip-address-of-existing-server] /tmp/openqrm-local-server integrate -u openqrm -p openqrm -q [openqrm
```

- 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
                                                                                    d
* After setting the system to 'network-boot' in its bios it also can be used to deploy server-images from
```

- To remove a system from openQRM integrated via the local-server plugin run the 'openqrm-local-server' utilagain. e.g. :

```
ssh [ip-address-of-existing-server] /tmp/openqrm-local-server remove -u openqrm -p openqrm -q [openqrm-se
```

---

local-storage

The Local-storage plugin adds support to deploy server-images to local-harddisk on the resources to openQRM. It provides mechanism to 'grab' server-images from local hard disks in existing, local-installed systems in the data-center. Those 'local-storage' server-images then can be dynamically deployed to any available resources in openQRM. The deployment function then 'dumps' the server-image 'grabbed' in the previous step to the harddisk of a resource and starts it from the local-disk. The 'local-storage' server-images are stored on a storage-server from the type 'local-storage' which exports the images via NFS.

Local-storage storage type: A linux-box (resource) which has NFS-server installed and a logical volume group available (lvm2) should be used to create a new Storage-server through the openQRM-GUI. The Local-storage system can be either deployed via openQRM or integrated into openQRM with the 'local-server' plugin.

Local-storage deployment type: The Local-deployment type supports to create server-images from existing systems and deploy those images to other available servers/resources.

How to use:

Grabbing a server-image from an existing system

- Create a Local-storage server via the 'Storage-Admin' (Storage menu)
- Create a 'local-storage' storage location (nfs-export on a lvol) using the

Local-storage plugins local-storage manager

- Create a 'local-storage' image ('Add Image' in the Image-overview)

Set the root-device and root-device-type to the storage location created in the previous step

- Go to 'local-storage' → 'Grab'
- In the first step select an idle resource to grab
- In the second step select the 'local-storage' image created in the previous step
- The content of the idle resources harddisk are now transferred (grabbed) to the

'local-storage' image location on the storage server.

Deploying a 'local' server-image to an available resource

- Create a 'local' image ('Add Image' in the Image-overview)
- Select a storage-server from the type 'local-storage'
- Set the root-device and root-device-type according to the 'local-storage' image to deploy
- create a new Appliance using a kernel and the just created 'local' server-image

Select an available resource

- Start the Appliance

This will reboot the resource and 'dump' the server-image from the 'local-storage' storage server the resource local-harddisk and starts it after the deployment finished.

When stopping the appliance, the disk content of the resource will be 'grabbed' again to update the server-image onthe 'local-storage' server.

---

lvm-storage

The 'lvm-storage' plugin transforms a standard Linux-box into a rapid-fast-cloning storage-server supporting snap-shotting for NFS-, Aoe-, and Iscsi-filesystem-images. The snapshots (clones from a 'golden server image') are immediately available for deployment and saving space on the storage-subsystem because just the delta of the server image is being stored. It adds a new storage-type 'lvm-storage' and three new deployment-types 'lvm-nfs', 'lvm-aoe' and 'lvm-iscsi' to the openQRM-server during initialization and basically combines the functionality of the 'nfs-storage', the 'aoe-storage' and the 'iscsi-storage' plugins.

Lvm-storage type: A linux-box (resource) with the Enterprise Iscsi-target, NFS-server and vblade (aoetools) installed should be used to create a new Storage-server through the openQRM-GUI. The Lvm-storage system can be either deployed via openQRM or integrated into openQRM with the 'local-server' plugin. openQRM then automatically manages the Aoe/Iscsi-disks and NFS-exports on the Lvm-storage server.

Lvm-deployment type: The three Lvm-deployment types ('lvm-nfs', 'lvm-aoe' and 'lvm-iscsi') supporting to boot servers/resources from the Lvm-storage server via NFS, Iscsi or the Aoe-protocol. Server images created with the 'lvm-nfs/iscsi/aoe' deployment type are stored on Storage-server from the storage-server types 'lvm-storage'. During startup of an appliance, they are directly attached to the resource as its rootfs.

How to use: The Lvm-storage server supports three different storage technologies, NFS, Aoe and Iscsi. The functionality and usage is conforming to the corresponding 'nfs-storage', 'aoe-storage' and 'iscsi-storage' plugins with the great benefit of the underlaying logical volume manager. This adds rapid-cloning capabilities through snapshotting and supports to create new server-images from 'golden-images' (server-templates) within seconds.

Please check the 'nfs/aoe/iscsi-storage' plugin for detailed usage information.

---

nagios3

The nagios3-plugin automatically monitors the systems and services managed by the openQRM-server.

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

---

netapp-storage

The NetApp-storage plugin integrates NetApp-Filer Storage systems into openQRM. It adds a new storage-type 'netapp-storage' and a new deployment-type 'netapp-iscsi' to the openQRM-server during initialization.

NetApp-storage type: A NetApp-Filer Storage system can be easily integrated into openQRM by adding a new

resource with the mac- and ip-address of the NetApp server. openQRM then manages the Volumes and Iscsi-Luns on the NetApp-Filer automatically.

NetApp-deployment type: The NetApp-deployment type supports to boot servers/resources directly from the NetApp-stoage server via the Iscsi-protocol. Server images created with the 'netapp-iscsi' deployment types are stored on Storage-server from the storage-server type 'netapp-storage'. During startup of an appliance, they are directly attached to the resource as its rootfs either through iscsi.

How to use:

- Create a new resource with the ip and mac-address of the NetApp-storage server (Resource menu)
- Create a NetApp-storage server via the 'Storage-Admin' (Storage menu)
- Create an (NetApp) Image ('Add Image' in the Image-overview). Then select the NetApp-storage server deployment-type ('NetApp Iscsi-root'). Select a NetApp storage device as the image root-device.
- Create an Appliance using one of the available kernels and the NetApp-Image created in the previous
- steps.Start the Appliance

---

nfs-storage

The Nfs-storage plugin integrates Nfs Storage-servers into openQRM. It adds a new storage-type 'nfs-storage' and a new deployment-type 'nfs-root' to the openQRM-server during initialization.

Nfs-storage type: A linux-box (resource) with 'nfs-server' installed should be used to create a new Storage-server through the openQRM-GUI. The Nfs-storage system can be either deployed via openQRM or integrated into openQRM with the 'local-server' plugin. openQRM then automatically manages the exports on the Nfs-storage server.

Nfs-deployment type: The Nfs-deployment type supports to boot servers/resources from the Nfs-stoage server. Server images created with the 'nfs-root' deployment type are stored on Storage-server from the storage-server type 'nfs-storage'. During startup of an appliance, they are directly attached to the resource as its rootfs via the nfs-protocol.

How to use:

- Create an Nfs-storage server via the 'Storage-Admin' (Storage menu)
- Create a new nfs-export on the Nfs-storage using the 'Exports' link (Nfs-plugin menu)
- Create an (Nfs-) Image ('Add Image' in the Image-overview). Then select the Nfs-storage server and chooseone of the Nfs-storage-devices as the 'root-device'.
- Create an Appliance using one of the available kernels and the Nfs-Image created in the previous
- steps.Start the Appliance

puppet

The Puppet plugin provides automated configuration-management for appliances in openQRM. It seamlessly integrates Puppet within the openQRM GUI and assists to put specific appliances 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. This 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/  .
```

Commits this repository will automatically the puppet configuration at /etc/puppet/* The puppet-configuration is organized in 'classes', 'groups' and 'appliances'. Own custom classes should be added to the class-directory. Classes should be then combined in 'groups' which will be automatically displayed in the puppet-manager. The 'appliances' section is fully managed via the puppet-manager user-interface.

The default puppet-plugin configuration comes with a set of pre-made puppet-classes and groups. The available groups are:

- basic-server
- webserver
- database-server
- lamp

Those pre-defined groups can of course be adapted and enhanced via the puppet svn repository.

Please notice that the puppet-plugin depends on the dns-plugin! Make sure to have the dns-plugin enabled and started before the puppet-plugin.

How to use:

- Go to the 'puppet-manager' in the puppet-plugin menu
- Select an appliance to automatic configure via puppet
- Select the puppet-groups the appliance should belong to

Within short time the puppet-server will distribute the new configuration to the appliance automatically.

---

sanboot-storage

The sanboot-storage plug-in is further described in the section about “Windows deployment”

---

solx86

The solx86-plugin provides an integration for already existing, local-installed open Solaris/Solaris X86 systems in openQRM. After integrating an existing, local-installed server it can be used as a ZFS-Storage server.

How to use:

To integrate a Solaris/open Solaris X86 system please login to the Solaris/open Solaris system as root and run thefollowing commands:

```
wget https://[openqrm-server-ip-address]/openqrm/boot-service/openqrm-solx86
chmod +x openqrm-solx86

./openqrm-solx86 integrate -u [openqrm-admin] -p [openqrm-admin-password] -q [openqrm-server-ip-address]
```

To remove the openQRM integration from your Solaris/open Solaris system please run :

```
./openqrm-solx86 remove -u [openqrm-admin] -p [openqrm-admin-password] -q [openqrm-server-ip-address]
```

sshterm

The sshterm-plugin integrates 'ajaxterm' into openQRM and provides a secure login to the openQRM-server and themanaged resources through the Web-interface.

No manual configuration is needed for the sshterm-plugin.

tftpd

The tftpd-plugin automatically manages to upload kernels to the resources (servers) managed by openQRM. Since the dynamic deployment methods in openQRM are based on network-booting (PXE) a tftpd-server is a fundamental service to server the operation-system files via the network. An automatic configured Tftpd-server is provided by this plugin.

How to use:

No manual configuration is needed for the tftpd-plugin. It automatically starts up the tftpd-service during start-up of the plugin.

tmpfs-storage

The tmpfs-storage plugin supports an “In-Memory” deployment method. It works in combination with the “install-from-nfs” hook and allows to run complete server within RAM.

Apart from the notable performance boost this non-permanent deployment method also takes advantages of being completely independent from local hard-disks since the whole system content is stored in memory. This makes it perfect for kiosk-mode scenarios such as internet cafes, IT training rooms but also for lean Hypervisor provisioning and High-Performance Computing (HPC). Another benefit for those setups is that since no local disks are used thereis not really much left which can break in the physical hardware; even then the hardware is replace-able at any time and no service is tied to it.

For the administrators of those kiosk-mode scenarios this mean less maintenance, a complete centralized management and the security that if users break their systems (the software installation) it will be fixed automatically by a reboot which causes a fresh deployment of a static server-template directly into RAM.

vmware-server

The vmware-server plugin is tested with VMware-server 1

VMware Server version 1 is known to be a great choice for applications which require a full-virtualization technology. VMware-server Virtualization hosts can be easily provisioned via openQRM by enabling this plugin. It also enables the administrator to create, start, stop and deploy the 'Vms' seamlessly through the web-interface. The virtual VMware-server-resources (Vms) are then transparently managed by openQRM in the same way as physical systems.

How to use:

- Enable and start the “local-server” plugin
- Integrate a VMware Server system via “local-server” (please check the local-server “about” page)
- Set the appliance (automatically created by “local-server” integration) resource-type to 'VMware-Server Host'
- Use the 'VMware Server Manager' in the VMware Server plugin menu to create a new VMware virtual-machineon the Host
- The created VMware Vms are then booting into openQRM as new resources

---

vmware-server2

The vmware-server2 plugin is tested with VMware-server-2.0.1-156745.i386

VMware Server 2 is known to be a great choice for applications which require a full-virtualization technology. VMware-server Virtualization hosts can be easily provisioned via openQRM by enabling this plugin. It also enables the administrator to create, start, stop and deploy the 'Vms' seamlessly through the web-interface. The virtual VMware-server-resources (Vms) are then transparently managed by openQRM in the same way as physical systems.

How to use:

- Enable and start the “local-server” plugin
- Integrate the VMware Server 2 system via “local-server” (please check the local-server “about” page)
- Set the appliance (automatically created by “local-server” integration) resource-type to 'VMware-Server2 Host'
- Use the 'VMware Server 2 Manager' in the VMware Server 2 plugin menu to create a new VMware virtual-machine on the Host
- The created VMware Vms are then booting into openQRM as new resources

---

vmware-esx

The vmware-esx plugin is tested with VMware ESX 3.5 - ESXi 4.0

This plugin integrates VMware ESX server as another virtual resource provider for openQRM. Since VMWare ESX does not provide an API for the linux operation-system yet the integration is currently done via 'password-less ssh' to the ESX server (from the openQRM-server).

How to get ssh enabled and 'password-less' login to the ESX server running is well documented in the internet.

Please notice that this mode is unsupported by VMware ! … still, we would like to be able to manage ESX.

Requirements : - An existing 'Datastore' (Storage) on the ESX server. Datastores in VMware ESX are the location where the virtual machine files are being saved. For the openQRM VMware-ESX plugin the default datastore can be configured in the plugin’s configuration file. By default, openQRM will try to gather the first available datastore (e.g., “datastore1” on a fresh installed ESXi 4.0) and use it for storing the virtual machines.

- password-less ssh access (as user root) from the openQRM server to the ESX server (as mentioned before). Hint: make sure to set /.ssh/authorized_keys to mode 0600 on the ESX host (dir and file)

How to use :

- How to integrate a VMware ESX server into openQRM :
- First make sure to enabled 'password-less ssh login' on the ESX server

To check you can run as root on the openQRM-server :

```
    ssh [ip-address-of-the-esx-server] ls
```

```
    This should give you a directory listing.

  * Now integrate the ESX server by running the following command :
```

```
    /usr/share/openqrm/plugins/vmware-esx/bin/openqrm-vmware-esx  init  -i  [ip-address-of-the-esx-server]
```

```
    This procedure will ask for a valid openQRM username and password.

  * The above procedure will integrate the ESX server within openQRM fully automatically.
    It will create the following components :

    - a resource (the ESX server)
    - a local storage placeholder for the ESX server resource
    - a local image placeholder for the ESX server resource
    - a local kernel placeholder for the ESX server resource
    - and a local appliance (the ESX server appliance)
  * Go to the 'ESX-Manager' within the VMware-ESX plugin menu. Select the ESX-appliance.
  * In the next screen you can now create/start/stop/remove/delete virtual machines on the ESX server.
    Created virtual machines will automatically start into openQRM and appear as new idle resources, ready
    for use.
```

---

windows

The windows plug-in is further described in the section about “Windows deployment”

---

xen

The xen plugin is tested with Xen 3.2 and higher and requires full virtualization (HVM via VT)

Xen Virtualization hosts can be easily provisioned via openQRM by enabling this plugin. It also enables the administrator to create, start, stop and deploy the 'Vms' seamlessly through the web-interface. The virtual Xen-resources (Vms) are then transparently managed by openQRM in the same way as physical systems.

Hint: The openQRM-server itself can be used as a resource for an XEN-Host appliance. In this case network-bridging should be setup on openQRM-server system before installing openQRM. At least an “internal” bridge for the openQRM management network is needed. The name for this bridge can be configured in the XEN plugin-configuration file via the parameter OPENQRM_PLUGIN_XEN_INTERNAL_BRIDGE.

Additionally, an external bridge (e.g., pointing to the internet) can be setup and configured via the OPENQRM_PLUGIN_XEN_EXTERNAL_BRIDGE parameter in the XEN plugin-configuration file. openQRM then will create every first (virtual) network-card for the XEN Vms on the internal bridge and every other on the external one. With this 2-bridge setup every vm will then have its first nic pointing to the openQRM management network (doing the pxe-boot) and every other nic will point e.g., to the internet.

After having a network-bridge configured openQRM should be installed on the internal bridge-interface (by default eth0). This can be done by setting the openQRM management network-interface in /usr/lib/openqrm/etc/openqrm-server.conf to br0 before initializing openQRM.

How to use:

- Create an appliance and set its resource-type to 'Xen Host'
- Use the 'Xen Manager' in the Xen menu to create a new Xen virtual-machines on the Host
- The created Xen vm is then booting into openQRM as regular resources

xen-storage

The xen-storage plug-in is further described in the section about “Windows deployment”

ZABBIX

zabbix

The zabbix-plugin automatically monitors the systems and services managed by the openQRM-server.How to

use:

After enabling and starting the Zabbix plugin you can login to Zabbix as 'Admin' with an empty password. Pleasemake sure to set password for the 'Admin' account at first login !

All managed systems by openQRM will be automatically discovered and monitored by Zabbix. You can now in detailconfigure the system and service checks via the intuitive Zabbix UI.

zfs-storage

The ZFS-storage plugin integrates ZFS-Target Storage into openQRM. It adds a new storage-type 'zfs-storage' and a new deployment-type 'iscsi-root' to the openQRM-server during initialization.

ZFS-storage type: A linux-box (resource) with the ZFS-target installed should be used to create a new Storage-server through the openQRM-GUI. The ZFS-storage system can be either deployed via openQRM or integrated into openQRM with the 'local-server' plugin. openQRM then automatically manages the ZFS-disks (Luns) on the ZFS-storage server.

ZFS-deployment type: The ZFS-deployment type supports to boot servers/resources from the ZFS-stoage server. Server images created with the 'iscsi-root' deployment type are stored on Storage-server from the storage-server type 'zfs-storage'. During startup of an appliance, they are directly attached to the resource as its rootfs via the iscsi-protocol.

How to use:

- Create an ZFS-storage server via the 'Storage-Admin' (Storage menu)
- Create a Disk-shelf on the ZFS-storage using the 'Luns' link (ZFS-plugin menu)
- Create an (ZFS-) Image ('Add Image' in the Image-overview). Then select the ZFS-storage server and select an ZFS-device name as the images root-device.
- Create an Appliance using one of the available kernels and the ZFS-Image created in the previous steps.
- Start the Appliance

Getting Started

Preparing the network

To take the most advantages of openQRM's management and deployment functionality it is recommended to separate the network into a dedicated “management- and storage-network” and a “public-network”. The “management-network” is completely managed by openQRM and separates the network-traffic from resources to storages from the public network-traffic. It is recommended to use a high-speed network especially for the “management- and storage-network”.

For the Virtualization Hosts in the openQRM environment it is recommended to adapt their bridging setup to the separated management- and public-network. The Virtualization plug-ins in openQRM supporting a custom bridge-definition in their specific virtualization plug-in's configuration file. Here e.g., the “xen-plugin” configuration options regarding the bridge-setup of the Hypervisor Hosts:

```
# required parameter to set the internal Xen bridge

# by default, openQRM will use the internal bridge eth0 for the virtual

# network-interface routed to the management/openQRM network on the Xen Vms.
```

```
# required parameter to set the external Xen bridge

# by default, openQRM will use external bridge eth0 for the virtual

# network-interface(s) routed to the public network (internet) on the Xen Vms
```

The first parameter OPENQRM_PLUGIN_XEN_INTERNAL_BRIDGE defines the bridge to use for the “management- and storage-network”. The second parameter OPENQRM_PLUGIN_XEN_EXTERNAL_BRIDGE defines the bridge to the “public-network”. Each parameter needs to be configured on the integrated Hypervisor host (after it downloaded the xen-plugin-boot-service) or on the openQRM Server itself (by unpacking the xen-plugin boot-service, adapting the configuration and packing it again). Other virtualization plug-ins in openQRM are working corresponding.

By default, the first virtual network-interface of Virtual Machines is automatically connected to the “management- and storage-network” while all further virtual NICs are connected to the “public-network”.

Setting up the server

Setting up the Operating System for the openQRM Server system is easy. Preferred Linux Distributions for the openQRM Server are latest Ubuntu or latest Debian. Just installing the base Operating System is enough since openQRM automatically fetches further package dependencies during setup. This is true for the installation by pre-build openQRM packages or via the installation from the sources.

Installing openQRM

From Packages

Pre-built openQRM packages for Debian, Ubuntu and CentOS are available on the openQRM Project sites files section at http://sourceforge.net/projects/openqrm/ [http://sourceforge.net/projects/openqrm/].

Debian/Ubuntu

Debian and Ubuntu packages can be installed via the dpkg util:

```
dpkg -i openqrm-[version]-debian1_[architecture].deb
```

To fetch all dependencies and initialize openQRM please run :

```
apt-get install -f
```

After that please continue in the Web-based configuration for the openQRM Server.

Another option is to use the package repositories of the openQRM Project at http://packages.openqrm.com/
[http://packages.openqrm.com/]. To use the package repositories please add the following line to /etc/apt/sources.list

```
deb http://packages.openqrm.com/lenny/ ./
```

Then run :

```
apt-get update

apt-get install openqrm
```

Now please continue in the Web-based configuration for the openQRM Server.

CentOS

On CentOS (or other Redhat-based Linux distributions) openQRM can be installed using the rpm tool:

```
rpm -iHv  openqrm-[version]-centos[version]_[architecture].rpm
```

To use the package repository of the openQRM Project please create a file /etc/yum.repos.d/openQRM.repo with the following content:

```
[openqrm]

name = Red Hat Enterprise $releasever - openQRM
baseurl  =  http://packages.linuxfellaz.net/centos5
enabled = 1

protect = 0
```

After that please run:

```
yum install openqrm
```

Then please continue in the Web-based configuration for the openQRM Server.

From Sources

Since openQRM's advanced build-system (details in the Development section) it is really easy to build openQRM from the sources. It just requires to have “make” and “svn” (subversion client) installed. The following commands automatically compile, install and start openQRM:

```
svn  co  https://openqrm.svn.sourceforge.net/svnroot/openqrm  openqrm
cd openqrm/trunk/src

make

make install
```

Further make-targets are explained in the Development section.

Updating openQRM

From Packages

In case the package repositories of the openQRM Project are used the update-manager of the Linux distribution openQRM is installed on will notify when new pre-built packages are available.

It is recommended to create a new state-backup of the openQRM Server state before applying the update!

It is also saved to uninstall and re-install the openQRM packages. The uninstall phase automatically creates a state-backup. During re-installation then please check the “restore from last backup” checkbox in the Web-configurator.

From Sources

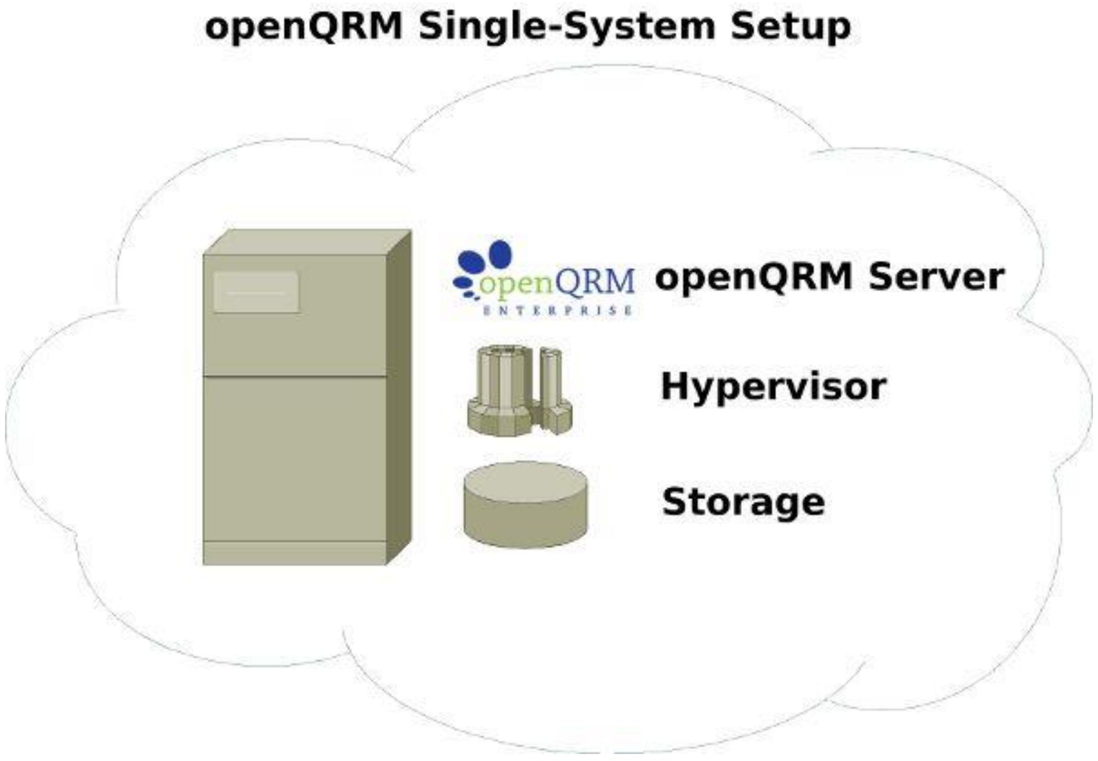
openQRM also supports direct updates from the sources. Since every base- and plug-in directory contains a separated makefile even partly updates are easy to apply. If updated source code is available change to the base-directory containing the updates and run:

```
make && make install
```

This will apply the updates to the installed openQRM Server. Also, plug-ins can be updated this way. Depending on the plug-in it may be needed to disable, re-enable and re-start the plug-in via the plugin-manager.

Configuring a Basic Setup

This basic-setup section describes a "single-system openQRM setup" which is perfect to start with openQRM. This setup can be scaled up with additional physical systems, virtual machines, external storage servers and more functionality via additional plug-ins at any time later.



When using a "single-system openQRM setup" the storage-component, a Hypervisor and openQRM Server itself are installed on a single physical system. In this setup the openQRM system acts as the storage server and virtual machine provider. Therefore, here some special configuration recommendations and requirements especially for the single-system setup:

- 1 physical system
    - VT CPU extension (full Virtualization support)
    - 1 GB Memory (or more)
    - 10 GB free disk space
    - 1 dedicated storage partition, 50 GB (or more)

Pre-setup the Hypervisor:

Install and configure the Hypervisor (e.g., KVM or Xen) on the system. Please make sure to configure the Hypervisors
bridge-setup according the management- and public-network (refer to "Preparing the Network").

Pre-**setup LVM**

For the basic-setup it is recommended to use the "lvm-storage" plug-in and its provided deployment types. To pre-configure the system to be used as an lvm-storage server make sure to have the following packages installed:

- lvm2 (logical Volume management)
- nfs-kernel-server (NFS Server)
- iscsi-target (Enterprise iSCSI Target)

Then initialize the dedicated storage partition to be used with lvm (in this example /dev/sdb1 is the dedicated freepartition) :

```
fdisk /dev/sdb
```

Now set the first partition (/dev/sdb1 in this example) to the partition type "8e" (Linux LVM) and save. The partition is now ready to be used for LVM. Please run the following commands to initialize the LVM partition and to create a volume group (named "vol" in this example) on it:

```
pvcreate /dev/sdb1
vgcreate vol /dev/sdb1
```

The storage part is now configured to be managed by openQRM.

Activate DHCP, TFTP, Storage Plugins

After setting up the openQRM Server system and pre-configuring the Hypervisor- and storage-components the initialopenQRM environment now can be started by enabling and starting the following plug-ins:

- dhcpd - automatically provides and manages the ip-addresses in the management- and storage-network
- tftpd - provides the network-boot-manager for a centralized kernel- and boot-management
- lvm-storage - provides 3 different storage- and deployment-types (lvm-nfs, lvm-iscsi and lvm-aoe)

Also please enable the Virtualization plug-in for the selected Hypervisor Technology (e.g., xen or kvm for this examplesetup)

- xen - integrates with Xen-Hypervisors and provides 2 new resource types : "Xen Host" and "Xen VM"
- kvm - integrates with KVM (kernel virtual machine) and provides 2 new resource types : "KVM Host" and "KVMVM"

Creating Images

Images in openQRM are logical objects containing parameters which are defining e.g., the image storage server, theexact image location on the storage server, the image name, version and more.

Please notice that Images in openQRM are always located on a storage server!

That means before actually creating a (server-) image in openQRM a storage server object needs to be created. In this basic-setup example we use one of the simplest storage type "lvm-nfs". To create a new "lvm-nfs" storage server in openQRM go to:

```
Base -> Components -> Create -> Storage
```

Select the "Lvm Storage Server (Nfs)" storage type and choose the openQRM Server resource (resource 0) as the storage resource. Provide a name for the storage server object and save. The new storage server is now available in openQRM at:

```
Base -> Components -> Storage
```

To create a new “image location” (a logical volume exported via NFS) click on the “Mgmt” button.

Image Locations

An “Image location” in openQRM is the physical storage location of a (server-) images root file-system content. It is the place on the storage server where the actual root-file-system content is stored. The image location is defined by the storage plug-ins “root-device identifier hook” (please check the Development section for more details).

After clicking the “Mgmt” button in the storage overview the next page presents a list of all available volume groups on the system. In our example it will show a single line for the “vol” volume group created in the previous step “Pre-setup LVM”. Select it and click on “select”.

The next form displays a list of existing logical volumes within the selected volume group (“vol” in our example). Use the “Add new logical volume to Volume group vol” section to add a new logical volume to the lvm-nfs storage server.For this basic-setup example name the new logical volume “myserver”. After clicking “add” the new created volume is listed in the volume overview.

What we have just created now is a new “image location” based on a logical volume of the given size. openQRM also created a file-system on this logical volume, mounted it and exported it via NFS (this is specific for the lvm-nfs storage type). The image location mount point is:

```
/[volume-group-name]/[logical-volume-name]/
```

In our example it is:

```
/vol/myserver
```

Of course, this new created volume is completely empty for now. How to create an openQRM image out of this image location and how to populate it with a root files-system content in the next section.

Logical Images

We have created a storage server and a (yet empty) image location on the storage server in the previous steps. Nowwe create a new, logical openQRM image object from those components. To create a new image, go to:

```
Base -> Components -> Create -> Image
```

Select the storage server on which the image is located (in our example the lvm-nfs storage server). In the next formplease provide a name for the image (we will use “myimage” as image name for this example).

The most important and required image configuration option is the “Root-device” select box. It presents the “image locations” available on the selected storage server. In our example the select box has one entry named “myserver”.

Optional fill out another image parameter such as version, root-password etc. Then save the image.

What we have done until now:

We have now created a new openQRM image (object). It is located on the (lvm-nfs) storage server and consist of theimage location created in the previous steps.

But the image location is still empty!?

How to automatically populate this empty image location with root file-system content is explained in the next section.

Using a pre-made Image from Image Shelves

openQRM provides the "image-shelf" plug-in which automatically populates NFS-based image locations from "imageshelfs". This "image shelfs" can be either local directories, ftp- and/or http locations providing zipped server-templates.

The openQRM Project offers a public available image-shelf which is automatically pre-configured in the openQRMsetup.

To populate the still empty image location in our example basic-setup please go to:

```
Plugins -> Deployment -> Image-shelf -> Import
```

The page presents a list of available image shelfs. For openQRM version 4.6 (and higher) please use the 1. image shelf provided by "openqrm-enterprise". For "older" openQRM setups (4.5 and lower) please select the image shelf from "www.openqrm-ng.net [http://www.openqrm-ng.net]".

On the following page a list of available server-templates on this image shelf is displayed. For this example, we pick the "Ubuntu-x86_64-9.10" server-template. Click on "get".

Now openQRM presents a list of NFS-based server-images available to "put" this server-template to. Please select the previous created "myimage" image.

openQRM now mounts the "myimage" image location, downloads and unpacks the root file-system content on the mounted directory and finally creates a new kernel from the just populated image location fitting to the selected server-template (in our example a 64bit Ubuntu kernel).

Please check the event list for progress on the download and image population phase.What we have so far:

We now have a ready-to-run, fully populated server-image which can be deployed via an appliance.

Adding Resources

Resources are added to openQRM in a fully automated way by simply net-booting them via PXE. openQRM auto-discovers new system and boots them into the "idle" mode, meaning free and available for provisioning. This is true for physical systems and for virtual machines. Virtual resources are created through the specific virtualization plugins "VM-Manager" and also net-booting into openQRM transparently.

The "idle" mode for resources is a quite important layer especially for resource-planning, dynamic scalability and high-availability. The pool of "idle" resources allows a detailed capacity planning and provides an exact overview about available systems for fail-over situations (hot-standbys). In the "idle" phase resources are ready to be picked up by an appliance which specifically contains the deployment information's e.g., which kernel and image to use for boot-up.

After assignment of a resource to a (started) appliance the resource reboots from "idle" to "assigned" state and starts

the defined server-image including its services. When stopping an appliance its resource are freed and released tothe "idle" resource pool.

Adding Kernels

The initialization phase of openQRM creates a “default” kernel from the kernel running on the openQRM server system itself. This “default” kernel is used to automatically discover and boot resources into the “idle” state.

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

```
OPENQRM_SERVER_BASE_DIR/openqrm/bin/openqrm kernel add -n name -v version -u username -p password [-l locati
```

- **Name** can be any identifier as long as it has no spaces or other special characters; it is used as part of thefilename.
- **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 openQRMbase dir under etc/templates.

Example:

```
OPENQRM_SERVER_BASE_DIR/openqrm/bin/openqrm kernel add -n openqrm-kernel-1 -v 2.6.29 -u openqrm -p openqrm
```

To specify a new version of the “default” kernel go to:

```
Base -> Components -> Kernels
```

Then select the new kernel which should replace the previous “default” kernel and click on “set-default”.

Creating Appliances

Appliances in openQRM act as the “service-container” and containing the detailed information’s and parameter how to deploy (and un-deploy) the service in the Data center. As explained in the Appliance model of openQRM appliances consist of the abstracted Data center components plus Service Level Agreements (SLA) defining e.g., CPU- and Memory requirements, machine type, High-availability and more. As the abstraction of the service-layer in openQRM appliances simplify the mostly very storage-, resource- and virtualization-type dependent deployment mechanisms since for starting and stopping an appliance knowledge about the custom underlaying subsystems is not needed.

Appliances in openQRM are created from the following components:

- A resource (physical system or virtual machine)
- A kernel (the Operating System kernel to boot)
- A server-image (logical object holding the storage-server and image-location details)
- Additional Service Level Agreements (SLA)

Appliances in openQRM are also used to define their special purpose as Virtualization-Host. Virtualization management for appliances is enabled by setting the “resource-type” to one of the “Virtualization-Host” system types.

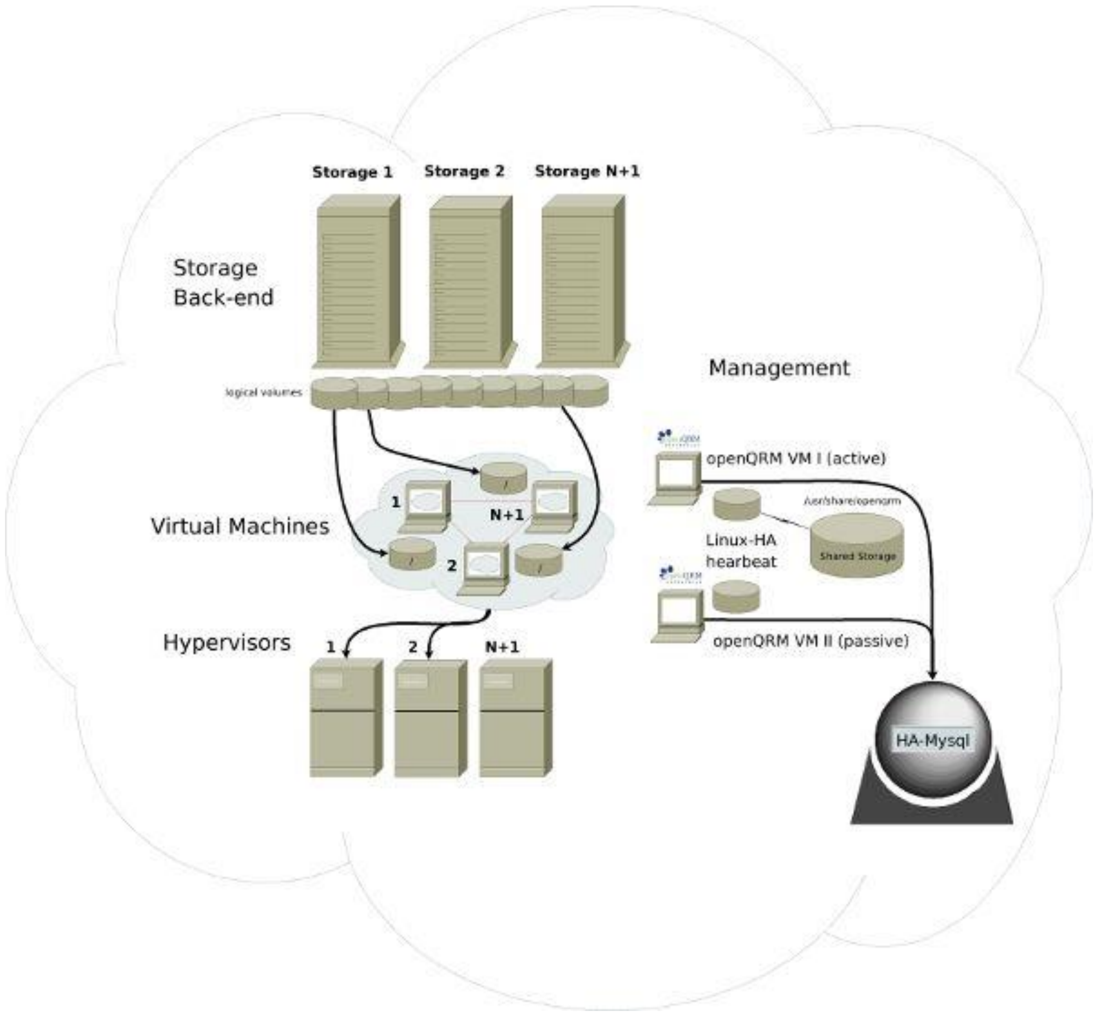
Common Use Cases

openQRM is to be known as the swiss-army-knife when it comes to Datacenter Management by integrating all involved Datacenter subsystems within a single management console. Here a list of its most common use cases:

- ISP customer provisioning
- Private- and Public Cloud Computing
- Developer Teams
- QA Teams
- Server consolidation / Virtual machine management

The basic-setup described in the previous section is a good starting point for a scalable openQRM environment. Fromthe "single-system openQRM setup" the next step is to distribute the storage- and Hypervisor-components to remote systems (physical resources) and to let openQRM "just manage" the distributed environment by sending commands to the specific resources.

Scalable High-Availability Setup



For production setups it is recommended to use a distributed openQRM environment and to make sure the openQRM Server itself is installed in a High-availability mode. This section explains how to configure HA for the openQRM Server.

Requirements for the HA setup:

- 4 (or more) physical systems
  - 1 (or more) system(s) dedicated to be Virtualization Hosts
  - 1 (or more) system(s) dedicated to be the Storage servers
  - 2 (or more) systems dedicated for openQRM Server (system A) plus Hot-Standby systems (system B)for fail-over

The explanation focus on the 2 systems dedicated for the openQRM Server plus its Hot-Standby system.

- Install the 2 systems with a Linux Distribution supported by openQRM (A and B)
- Connect both systems to the network and configure their ip-addresses (A and B)
- Install and setup openQRM Server on system A, use a remote and High-available Database and an aliasnetwork-interface (only A)
- Create a shared storage on (one of) the Storage server
- Stop openQRM Server on system A (only A)

- Move the openQRM Server base-directory to a temporary directory (only A)
- Mount the shared storage location from the Storage server at the openQRM Server base-directory (A and B)
- Move the openQRM Server content from the temporary directory into the now (mounted) base-directory (onlyA)
- Install and setup Linux-HA on both systems
- Create a wrapper-init script responsible for starting openQRM Server on the active Cluster node
- Add the wrapper-init script as a Linux-HA service using a global Cluster ip-address

Now Linux-HA automatically takes care to activate the openQRM Server only on its active Cluster node and to fail-over the service plus its ip-address in case of a system failure.

Integration into existing IT infrastructures

The migration of an existing IT infrastructure to an openQRM managed environment is seamless and can be done in small, self-defined and conceivable steps. Basically, it is all about transferring existing server (their root file-system content) to a storage managed by openQRM. To migrate an existing server system into openQRM please follow the steps below:

- Create an empty image location on one of the storage servers in openQRM
- Sync the complete root file-system content of the existing server to the empty image location on the storage e.g., via rsync
  - This step can be done during run-time of the existing system. If it is a virtual machine it is recommended to stop it and start the syncing procedure when it fully shut down.
  - Make sure to exclude /proc and /sys from the syncing procedure and create them in the imagelocation as empty directories.
- Shut down the existing server, set its BIOS to network-boot (PXE) and reboot it.
  - It will be automatically added to openQRM as new, idle resource
- Create an image using the previous created image location which is now populated with the root file-systemcontent of the existing server

- Create an appliance using the existing systems resource, the new created image and a kernel.
- Start the appliance

The existing server is now fully migrated and managed by openQRM. Its root-device is located on the storage server (the image location) and its resource is now fully flexible and can be replaced at any time. The appliance can now also move to a virtual machine or to a different physical system at any time since its root file-system content is not tiedto its local hard-disk any more.

Windows deployment

kvm-storage

The 'kvm-storage' plugin is a combination of the “lvm-storage” and the “kvm” plugin. It provides a new storage type “kvm-storage” based on lvm which is used a local-disk device for KVM virtual machines on the same system.

Please note: This plugin provides support to manage KVM virtual machines in the “common” way. That *means that the KVM Vms are using local-disks which are logical volumes on the KVM-Storage Host. This results in a dependency to “local-disk” devices on the KVM-Storage Host. ⇒ the KVM VMs depends on the logical volumes provides by KVM-Storage Host ⇒ VMs must be on the same KVM-Storage host where the logical volume (the VMs root-disk) is located*

Requirements:

- A server for the KVM-Storage Host

(this can be a remote system integrated into openQRM e.g., via the “local-server” plugin or the openQRM server itself)

- The server needs VT (Virtualization Technology) Support in its CPU (requirement for KVM)
- lvm2 tools installed
- One (or more) lvm volume group(s) with free space dedicated for the KVM VM storage
- KVM installed
- One or more bridges enabled for the KVM virtual machines

1. KVM Storage Management :

- Create a new storage from type “kvm-storage”
- Create a new logical volume on this storage
- Use the “local-storage” plugin to populate the new logical volume
- Now use the “linuxcoe-plugin” to automatically install a Linux distribution on it.
- Another option is to connect to the VMs VNC console and install an OS in the regular way.
- Create an Image using the new created logical volume as root-device

Result is an openQRM Image (server-template) which can be deployed to a KVM-Storage VM (on the same system)via an Appliance.

2. KVM (Storage) VM Management :

- Create a new appliance and set its resource type to “KVM-Storage Host”
- Create and manage KVM virtual machines via the KVM-Storage VM Manager

This results in new (idle) resources in openQRM which can be deployed with KVM-Storage volumes (on the same system) via an Appliance.

3. KVM Storage Deployment :

- Create a new appliance
- Select an idle resource with the type “KVM-Storage VM”
- Select an “KVM-Storage” Image (on the same system as the idle resource)
- Set the resource type of the appliance to “KVM-Storage VM”
- Start the appliance

This step will “assign” the logical volume on the KVM-Storage Host as the local-disk and boot device to the KVM-Storage VM (on the same system). The VM now boots up locally from the logical volume specified by the Image.

xen-storage

The 'xen-storage' plugin is a combination of the “lvm-storage” and the “xen” plugin. It provides a new storage type “xen-storage” based on lvm which is used a local-disk device for Xen virtual machines on the same system.

Please note: This plugin provides support to manage Xen virtual machines in the “common” way. That *means that the Xen Vms are using local-disks which are logical volumes on the Xen-Storage Host. This results in a dependency to “local-disk” devices on the Xen-Storage Host. ⇒ the Xen VMs depends on the logical volumes provides by Xen-Storage Host ⇒ VMs must be on the same Xen-Storage host where the logical volume (the VMs root-disk) is located*

Requirements:

- A server for the Xen-Storage Host

(this can be a remote system integrated into openQRM e.g., via the “local-server” plugin or the openQRM server itself)

- The server needs VT (Virtualization Technology) Support in its CPU (requirement for Xen)
- lvm2 tools installed
- One (or more) lvm volume group(s) with free space dedicated for the Xen VM storage
- Xen installed
- One or more bridges enabled for the Xen virtual machines

1. Xen Storage Management :

- Create a new storage from type “xen-storage”
- Create a new logical volume on this storage
- Use the “local-storage” plugin to populate the new logical volume
- Now use the “linuxcoe-plugin” to automatically install a Linux distribution on it.
- Another option is to connect to the VMs VNC console and install an OS in the regular way.
- Create an Image using the new created logical volume as root-device

Result is an openQRM Image (server-template) which can be deployed to a Xen-Storage VM (on the same system)via an Appliance.

2. Xen (Storage) VM Management :

- create a new appliance and set its resource type to “Xen-Storage Host”
- Create and manage Xen virtual machines via the Xen-Storage VM Manager

This results in new (idle) resources in openQRM which can be deployed with Xen-Storage volumes (on the same system) via an Appliance.

3. Xen Storage Deployment :

- Create a new appliance
- Select an idle resource with the type “Xen-Storage VM”
- Select an “Xen-Storage” Image (on the same system as the idle resource)
- Set the resource type of the appliance to “Xen-Storage VM”
- Start the appliance

This step will “assign” the logical volume on the Xen-Storage Host as the local-disk and boot device to the Xen-

Storage VM (on the same system). The VM now boots up locally from the logical volume specified by the Image.

sanboot-storage

The “sanboot-storage” plugin provides a rapid, image-based deployment method especially for the Windows Operating system. It integrates with GPXE [http://etherboot.org/] and supports to seamlessly connect iSCSI Luns and AOE/Coraid Storage devices directly in the bootloader stage, before the actual Operating System is loaded. This makes it extremely useful specifically for Windows systems because Windows will still “think” it is using a local hard-disk.

Requirements for a sanboot-storage server:

- one or more dedicated LVM volume group(s) with free space available
- Enterprise iSCSI Target installed
- Software AOE Shelf “vblade” installed

How to deploy Windows systems booted directly from SAN

- Enable and start the “dhcpd”, “tftpd”, “sanboot-storage” and “windows” plugin
- Start a physical system with the boot-sequence set to: first network-boot, second CDROM
- The system now boots from the network and is automatically added to openQRM as new, idle resource Create a new storage from type “sanboot-storage”
- Create a new volume on the sanboot storage Create a new image out of the just created volume
- Create a new appliance using the idle resource, the default kernel and the previous created image.
- Start the appliance

The system will now reboot assigned and configured to boot directly from the iSCSI Lun (or AOE Shelf). Since the Lun is still empty it will fail after the network-boot connected the iSCSI Lun (or AOE Shelf) as local-disk and then jumpto the next configured boot-device, the CDROM. At this point Windows (7) can be installed normally.

Please notice that “Windows XP” needs to be first installed on a local hard-disk and then the hard-disk imageis transferred to the iSCSI Lun or AOE Shelf.

This step is not needed for Windows 7 which can be directly installed on the iSCSI Lun or AOE Shelf.

For the detailed documentation how to transfer Windows XP installations to a SAN please refer to:

http://etherboot.org/wiki/sanboot [http://etherboot.org/wiki/sanboot] and http://etherboot.org/wiki/sanboot/iscsi_install

[http://etherboot.org/wiki/sanboot/iscsi_install]

Installing the Windows openQRM-Client

After successful installation the Windows system should be integrated into openQRM by installing the Windows openQRM-Client on it. The Windows Plugin adds support for the Windows Operating Systems to openQRM. It consistof a basic monitoring agent and a remote-execution subsystem which run on the Windows systems as Windows services after integrating them with a simple setup program. The openQRM-Client setup program is available for download on the “About” page of the “windows” plugin.

Please notice: Before you run the setup program for the Windows openQRM-Client please create a new user“root” on the windows system!

Please notice: After running the Windows openQRM-Client installer please make sure to have TCP port 22(ssh) enabled in the Windows firewall!

Hint for Windows XP: Please run “gpedit.msc” and add the Permission to remote shutdown the system to user“root”

openQRM is now able to manage and monitor the Windows system (-image). Commands on the Windows systems are executed via the “virtual resource command hook” which transparently translates e.g., a “reboot” action into “shutdown.exe”.

The Windows server-image is now ready and fully functional using the openQRM appliance model.

It is recommended to create snapshots from the original Windows server-image after it is finally setup andthen just to deploy the snapshots of the origin!

That way one can keep a different Windows system revision without losing the origin state. Just re-creating the snapshot will cause a fresh Windows installation.

Automatic Configuration Management

Image-based deployment basically comes in 2 flavours:

- Ready-setup and pre-configured Server images
- Minimal base Server images which are then leveraged during deployment time

This section describes automated methods to leverage server images according users’ requirements during deployment time.

Plugin Boot Services

Plug-in boot-services can be used to e.g.:

- install additional package dependencies
- pre-setup applications
- start additional services

The basic openQRM Client boot-service provides a handy shell function “openqrm_install_os_dependency()” to install additional package dependencies. The function is located in

```
OPENQRM_SERVER_BASE_DIR/openqrm/include/openqrm-package-functions
```

```
openqrm_install_os_dependency [package-name]
```

Sourcing this function file from a plug-in boot-service init script makes it available. It requires the package name toinstall as first argument and can be used like:

Depending on the Linux distribution it automatically decides on the right package manager (dpkg or rpm) and triggers and installation of the named package. The function can be used in “force” mode by exporting the following shell parameter before running it:

```
FORCE_INSTALL="true"
```

Additional to automated package installation a plug-in boot-service init script can of course be used to pre-configure several custom application configurations and to start additional service on the resource it is running on.

Automated configuration management via Puppet

The Puppet integration in openQRM allows further to fully automate the deployment process of the application stack on the requested Cloud Appliances.

Puppet was selected by the openQRM team as the number one choice when it comes to automated configuration management because with it allows such a fine- g r a i n e d development of large IT infrastructures. Compared to other tools Puppet allows to "code" the data-center environment via the Puppet language. This programmable configuration layer allows system-administrators, QA and developers to "develop" their IT infra-structure via the well-defined puppet-language. The data-center then becomes "re-makeable".

The Puppet features in openQRM are also integrated and available in the openQRM Cloud.

Monitoring

The first step to make sure all systems and services in a data centers are running well is to monitor them. openQRM comes with several different Datacenter monitoring options further described in this section.

Basic Monitoring

openQRM comes with a Client, automatically installed on all managed systems in the openQRM network through the basic openQRM Client boot-service, which includes a basic monitoring utility. This "openqrm-monitord" consist of a shell script sending statistics to the openQRM Server via the https protocol. The statistical data includes uptime, load, CPU model + count, network-interface count, memory, swap etc. openQRM gathers those statistics and collects them in its database.

In case a resource does not send statistics for 4 minutes openQRM sets its state to "error" and runs the "high-availability hook" (further described in the Development section).

The openQRM Client is also available for the following non-Linux Operating Systems:

- Windows - described in the "Windows deployment" section
- Solaris/open Solaris - described in the "solx85 plug-in" section

Nagios

A well-known, proven and widely used monitoring tool is Nagios which is available for openQRM in the flavor of an additional plugin. The second, also essential, step is the automatic handling of errors, what openQRM is famous for. The combination of the enhanced monitoring utility Nagios and the automated error-handling, high-availability and fail-over features of the openQRM data center management platform creates a powerful and dynamic environment which reduces down-time of systems and services in a modern data center to the minimum.

Nagios is a widely adopted open-source host, service and network monitoring program based on a client/server concept. The Nagios-server gets monitoring information’s by active- or passive-checks. That means it either actively tests the availability of a system or service from the Nagios-server itself or it passively receives information’s about tests running on the remote system. The passive checks are initiated by the Nagios-client (nrpe) running on the systems monitored by the Nagios-server. The client part of Nagios is designed in a plug-able way. New service checks can be added easily by creating new Nagios-plugins interfacing with the plugin-architecture of the base Nagios- server.

The Nagios-server runs within an Apache web-server and consists of perl and shell scripts mixed with binary tools executed via the CGI-interface. In configurable intervals it checks various services like SMTP, POP3, HTTP, NNTP, PING, etc. and provides the gathered data center information’s in a nice web-interface. It also monitors system resources like CPU-load, memory and disk usage, running processes, logs, etc. and environmental factors such as CPU-temperature.

collectd

collectd is a daemon which collects system performance statistics periodically and provides mechanisms to store the values in RRD files. The integration of collectd as an openQRM plug-in provides an automated setup of the collectd server and client on all managed appliances within the openQRM network. The collectd Clients send their statistics information’s to the main collectd server running on the openQRM Server as a plug-in service. Per appliance statisticsare stored in RRD Database files which are used as the data source to frequently create well-arranged system statistics graphs out of them. The system graphs are embedded into the openQRM UI and also available for openQRM Cloud users.

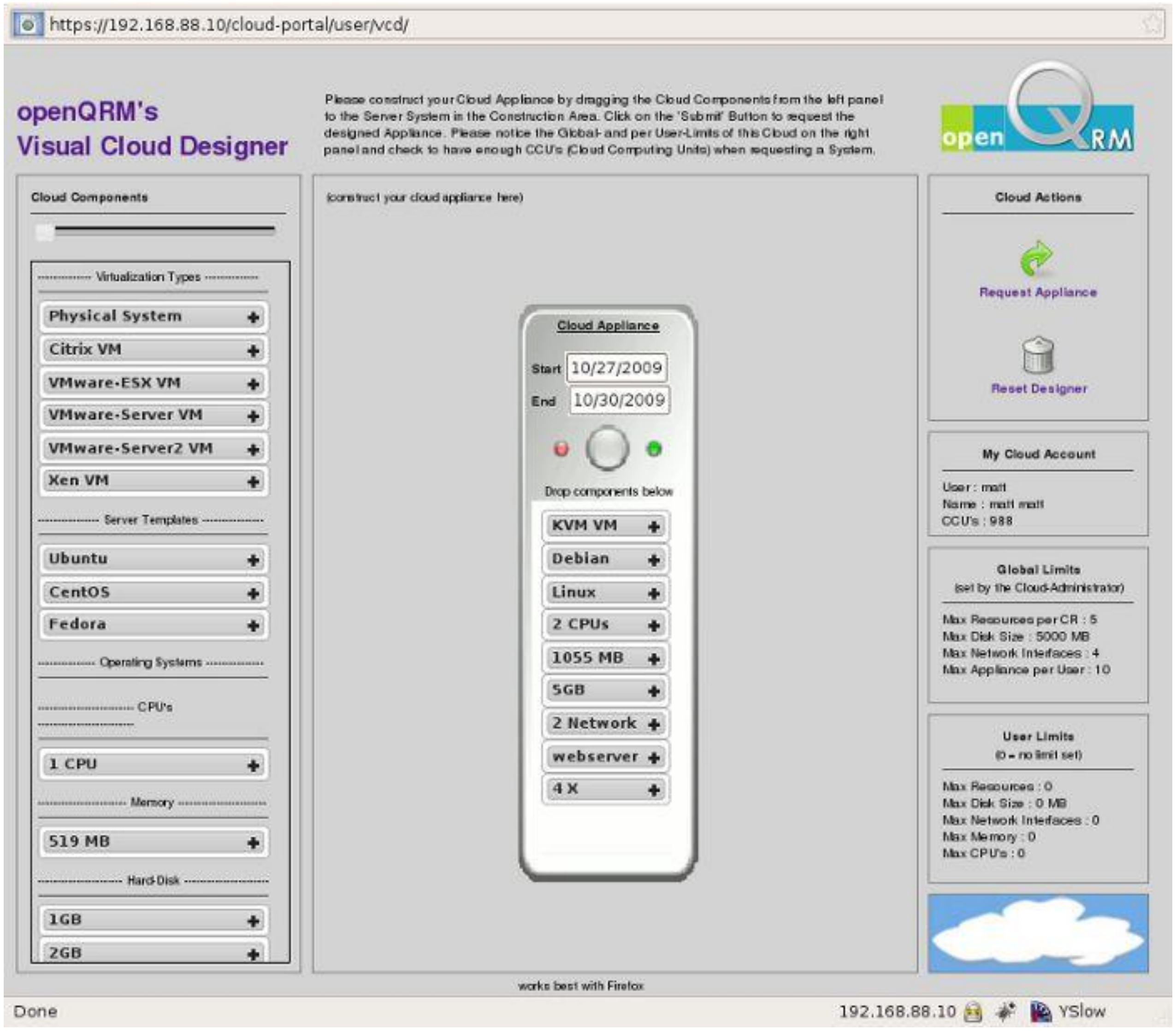
Zabbix

Zabbix is a quite new System- and Service monitoring tool which is known for its great scalability. It is an enterprise-class open-source distributed monitoring solution also available in openQRM as additional plug-in.

The Zabbix plug-in provides an automated Zabbix Server and Client setup for the appliances managed by openQRM. Zabbix Clients are automatically discovered by the Zabbix Server and through the embedded UI custom system-, service- and network-checks can be defined.

The openQRM Cloud

Here a screenshot of openQRM's Visual Cloud Designer:


https://192.168.88.10/cloud-portal/user/vcd/

openQRM's Visual Cloud Designer

Please construct your Cloud Appliance by dragging the Cloud Components from the left panel to the Server System in the Construction Area. Click on the 'Submit' Button to request the designed Appliance. Please notice the Global- and per User-Limits of this Cloud on the right panel and check to have enough CCU's (Cloud Computing Units) when requesting a System.

Cloud Components

Virtualization Types: Physical System, Citrix VM, VMware-ESX VM, VMware-Server VM, VMware-Server2 VM, Xen VM

Server Templates: Ubuntu, CentOS, Fedora

Operating Systems

CPU's: 1 CPU

Memory: 519 MB

Hard-Disk: 1GB, 2GB

(construct your cloud appliance here)

Cloud Appliance

Start 10/27/2009

End 10/30/2009

Drop components below: KVM VM, Debian, Linux, 2 CPUs, 1055 MB, 5GB, 2 Network, webserver, 4 X

Cloud Actions: Request Appliance, Reset Designer

My Cloud Account

User : matt
Name : matt matt
CCU's : 988

Global Limits
(set by the Cloud-Administrator)

Max Resources per CR : 5
Max Disk Size : 5000 MB
Max Network Interfaces : 4
Max Appliance per User : 10

User Limits
(0 = no limit set)

Max Resources : 0
Max Disk Size : 0 MB
Max Network Interfaces : 0
Max Memory : 0
Max CPU's : 0

works best with Firefox

Done 192.168.88.10 YSlow

What is Cloud Computing?

Approximately since beginning of 2008 the term “Cloud Computing” came up as new hype in the IT-World directly after “Virtualization”. Since then, this term is used for different new and also well-known technologies for managing large data-centers and IT infrastructures. Some use it as a synonym for “Software as a service” (SAAS) coming from the Web 2.0 evolution, some other connect the term “Cloud Computing” with Automated Provisioning ofa large number of virtual machines and “Servers on demand”. In general, the definition of the term “Cloud Computing” leaves it open but defines in more generic way:

Cloud computing is Internet (“cloud”) based development and use of computer technology (“computing”) [1][2][3]. It isa business information management style of computing in which typically real-time scalable [4] resources are provided “as a service” [5] over the Internet [6] to users who need not have knowledge of, expertise in, or control over the technology infrastructure (“in the cloud”) that supports them [7]. (Wikipedia → Cloud Computing)

Even more, both aspects of Cloud Computing, SAAS and Automated Provisioning, are co-existing in a kind of symbioses relationship. SAAS means to provide a Service via a network (Inter- or Intranet) to its customers on demand. Most common for this case is that the applications and services are deployed to a separated virtual machine. This isolation of the application in its own VM results in better flexibility and security for the SAAS provider.

The requirement to deploy a huge number of VMs (one per application instance) needs automatism which is solved through Automated Provisioning.

Cloud Computing is not a new technology but a new name for a combination of two or more already known technologies like HPC- and HA Clustering, Grid-Computing, Utility-Computing, Distributed-Computing etc. They are all there to manage large IT infrastructures used for different purposes in an automated way.

The role of the open-source community especially for Cloud Computing

Modern Clouds mostly consists of already existing, well-known and often open-source components. For every aspectof a Cloud environment a different set of utilities is used e.g., Puppet for automated configuration management, Nagios for system- and service monitoring, one or more virtualization technologies, Linux-HA for high-availability etc. Those utilities are already used in large production and accepted by system administrators and the IT-management.It would make no sense to re-write all those tools from the scratch just because of a new name (Cloud Computing) forknown methods of deploying and managing large server environments. For that reasons Clouds are mainly a set of “loosely connected” tools. The challenge for modern Cloud Computing is the integration of those “loosely connected tools” into a single management User Interface. It can only be archived via a plug-ability which combines all the separated utilities to benefit from their cooperation. This is exactly the goal of the openQRM data-center management platform.

Types of Cloud Solutions

Public Clouds [http://en.wikipedia.org/wiki/Cloud_computing#Public_cloud] Private Clouds

[http://en.wikipedia.org/wiki/Cloud_computing#Private_cloud] Hybrid Clouds

[http://en.wikipedia.org/wiki/Cloud_computing#Hybrid_cloud]

Currently most Clouds are specialized to “only” automated Virtual Machine management but not to manage completeIT infrastructures using the concept of “Cloud Computing” and “resources on demand”.

Public vs. Private Cloud Computing

Clouds can be divided into “Public Clouds” and “Private Clouds”. While within a private cooperated data-center environment security aspect may depend on the system administrator and on their actual usage. For an internally QA network, protected by the company’s firewall, security may not be that critical compared to the load balancer cluster of the main database application. In any way for Public Cloud Computing an exhaustive security concept is neededto ensure the integrity of the users’ data and provide continuously reliability for the Cloud service. OpenQRM's thoroughly designed security concept consist of automated authentication of the server-images, the separation of the hard- and software layer to ensure the data-integrity and enable service high availability via rapid re-deployment plusthe separation between the actual Cloud management system and the end users Cloud portal. To gain additional security on top of the Cloud-security virtual private networking (VPN) and file system encryption can (and should) be additionally used in the application layer.

Comparison of major Cloud Competitors

The Cloud Comparison Matrix [http://www.openqrm-enterprise.com/fileadmin/DATA/Whitepapers/cloud_computing_compare-20091215.pdf]

Created by participants of Devopsdays 09 in Ghent

openQRM Cloud

On top of this straight and generic openQRM framework sits the Cloud plugin as a "simple UI" for the end user. It provides a fully automated provisioning cycle from physical systems or virtual machines deployment through automated application and configuration management according the users requests up to automated de-provisioning to free up the users compute resources. Via an additional external Cloud web-portal user registered to the Cloud can login to manage their own Cloud environment by requesting new resources, de-provisioning existing ones or managing their active Cloud requests.

For the system administrator the Cloud provides an internal interface plugged into the openQRM server. It provides a fine- g r a i n e d overview about the current Cloud activities and several configuration options to tune the Cloud behavior
e.g., the Cloud can be set to automatically or manual approve new Cloud request, automatically create new virtual machines if not enough existing one are available, enable or disable the clone-on-deploy features etc. It also consist of a Cloud Ip-manager which is used to automatically configure the external network-interfaces of the provisioned machines.

The default behavior of the openQRM Cloud is to use the "clone-on-deploy" mechanism to provision resources requested by the users. This means that for every Cloud request the integrated storage management will execute a clone command on the storage server hosting the "golden-image" (server-image template) and then use the snapshotted server-image to deploy it for the user. This method has the huge advantage that the snapshots on the storage server (e.g., via the LVM snapshot storage feature) normally do not consume any space because they are only read-only copying from the original logical volume (server-image). This means only the changes the users makesto its server are saved to the storage and no additional space for the actual root file system plus its applications.

For automated leveraging the application layer of the provisioned systems according to the users request openQRM integrated with the Puppet configuration management system. Puppet takes care to setup and pre-configure the users' machines via pre-made and known-to-work recipes. The integration of Puppet as an additional plugin and the cooperation of the Puppet-plugin and the Cloud add-on provides a selection of "out-of-the-box" application servers and gives added value by automatism for the users and administrators.

The openQRM Cloud comes with an integrated billing system for the compute resources consumed by the end users.Based on "Cloud Computing Units" (CCU's), the virtual currency in the openQRM Cloud, system administrators and users can plan and keep track of their used compute power and costs. OpenQRM simply calculates the amount of CCU's per request and subtract it from the Users account. That way the Cloud manager is completely open how to sell the computing power to the market end users (e.g., via eBay).

More details about the Cloud configuration in the section "Plugin description".

Development

openQRM plug-able architecture is designed to fully support continuous integration. With its static base framework which basically just manages the different plug-ins openQRM allows several developer (-teams) to work on several plug-ins in parallel without interfering each other. openQRM comes with several special features making the life of its developers easier e.g., its build- and packaging system described in the next chapter.

The openQRM Build System

The sources of openQRM are organized in the same way as in the later installation. Its directory structure is adapted to a standard Unix filesystem layout plus the source (or base-dir) pre-fix. openQRM tools are located in src/bin, configuration files in src/etc, daemons in src/sbin and plugins in src/plugins. The main makefile runs a loop for all directories in src/. In each directory it executes the corresponding make target. In the plugins dir it runs a separated loop for all plug-in directories.

The main makefile contains the following make targets:

- check - checks runtime requirements
- configure - pre-configures the sources (empty by default)
- compile - compile openQRM and its components
- install - install openQRM from the previous compiled components (needs root)
- start - starts the openQRM Server (needs root)
- uninstall - uninstalls the openQRM Server (needs root)
- clean - cleans the source tree
- realclean - cleans the source tree and build cache
- reinstall - stops, uninstalls, cleans, re-compiles, installs and starts openQRM Server (needs root)
- rpm - builds rpm packages (needs root)
- deb - builds Debian packages (needs root)
- debsource - builds Debian source package
- initrd - builds an initrd-template
- buildrequirements - checks the build-requirements
- update - updates sources from the svn repo

build-cache

The openQRM build-system uses its own caching for downloaded components and already compiled binaries. This build-cache creates a directory src/../buildtmp (one level above src/) which keeps files once they are downloaded and/or compiled once. This method dramatically speeds up the development of openQRM since frequent re-compilations just takes a few seconds. Please notice that the first build will always take "some time" but all components are being cached, each further build just takes a few moments.

build-functions

openQRM comes with 2 especially useful build functions contained in src/include/openqrm-build-functions.

- **openqrm_cache_or_download**

Makes sure a component is available in the build-cache, if not trigger download

- **openqrm_compile_from_source**

Compiles and sets up components in the build-cache

Those functions takes a "component.conf" build-configuration as the first argument. Further action is fully automated. The components build-configuration file accepts the following parameters:

```
OPENQRM_SOURCE_VERSION="[components-version]"

OPENQRM_SOURCE_DOWNLOAD="http://[components-download-location]/[component]-$OPENQRM_SOURCE_VERSION.tar.gz"
OPENQRM_SOURCE_BINARY_RESULT="[resulting-files]"
```

Additionally, the following shell exports controlling the exact compile phase:

```
OPENQRM_MAKE_TARGET="[the-make-target-to-be-executed]"
```

```
OPENQRM_PRE_MAKE="[commands-to-be-executed-before-the-make-phase]"
```

```
OPENQRM_POST_MAKE="[commands-to-be-executed-after-the-make-phase]"
```

To run those functions directly from within makefiles a “make-assistant” util is available in the main source dir.

automated packaging

The packaging-functions file (src/include/openqrm-package-functions) contains functions for fully automated packaging:

- openqrm_rpm_package - Builds RPM-packages of openQRM using rpmbuild and .spec file templates fromsrc/rpm/ (executed by “make rpm”)
- openqrm_create_deb_source - Creates a Debian source-package of openQRM (executed by “make debsource”)
- openqrm_pbuilder - Builds Debian-packages via the Debian “pbuilder” packaging util (executed by “makedeb”)

Plugin Development, Third-Party Integration, API

Plugin Template

To make it easy starting to develop a new openQRM Plug-in an openqrm-create-plugin script is provided which automatically creates a new Plug-in skeleton based on an existing one. Here how to use it:

```
OPENQRM_SERVER_BASE_DIR/openqrm/bin/openqrm-create-plugin  OLD_PLUGIN_NAME  NEW_PLUGIN_NAME
```

Basically, this script renames all OLD_PLUGIN_NAME into NEW_PLUGIN_NAME.

provided Plugins Hooks

- **appliance hook (PHP)**

During add/remove/start/stop appliance actions openQRM checks for each plugin if it provides an appliance-hook fileat

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/web/openqrm-[plugin-name]-appliance-hook.php
```

This hook should implement the following function :

```
openqrm_[plugin-name]_appliance($cmd,  $appliance_fields)
```

openQRM then runs this function with the following parameters:

```
$cmd - add, remove, start, stop
```

```
$appliance_fields - An array containing all appliance parameters
```

Example:

The dns-plugin is using the appliance hook to automatically manage DNS resolving of the appliance name duringstart and stop actions.

- **virtual resource command hook (PHP)**

For the reboot and power off resource actions openQRM looks for

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/web/openqrm-[plugin-name]-resource-virtual-command-hoo
```

If existing openQRM runs :

```
openqrm_kvm_storage_resource_virtual_command($cmd,  $resource_fields)
```

Parameters in this hook function are :

```
$cmd - reboot, halt
```

```
$cmd - reboot, halt
```

Example:

The “kvm-storage” plug-in is using this virtual-resource-command hook to map e.g., the “reboot” resource action to thecorresponding “VM restart” command.

- **kernel hook (Shell)**

When a new kernel is created via the OPENQRM_BASE_DIR/openqrm/bin/openqrm util openQRM checks if plug-insproviding an

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/include/openqrm-plugin-[plugin-name]-kernel-hook
```

shell function file implementing the “xen_kernel_hook” functions. This function is feed with the following parameters:

```
KERNEL_NAME=$1
KERNEL_VERSION=$2
KERNEL_LOCATION=$3
KERNEL_TYPE=$4
```

Example:

The xen plug-in is using this kernel hook to additionally copy the Hypervisor file to the tftpboot directory.

- **assign hook (Shell)**

During assignment (start) of an appliance openQRM checks if plug-ins providing an

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/include/openqrm-plugin-[plugin-name]-assign-hook
```

shell function file implementing the ”[plugin]_assign_hook” functions. This function is feed with the following parameters:

```
OPENQRM_RESOURCE_ID=$1
OPENQRM_KERNEL_NAME=$2
OPENQRM_RESOURCE_PXELINUXCFG_FILE=$3
```

Example:

The xen plug-in is using this assign hook to re-write the resource PXE configuration.

- **resource hook (PHP)**

The openQRM resource hook is executed if plug-ins providing the following resource-hook file:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/web/openqrm-[plugin-name]-resource-hook.php
```

If existing openQRM runs:

```
openqrm_[plugin-name]_resource($cmd, $resource_fields)
```

Parameters in this hook function are:

```
$cmd - add, remove
```

```
$resource_fields - An array containing all resource parameters
```

Example:

The “dhcpd” plug-in is using this resource-hook add (and remove) resource to (and from) its dhcpd.conf configurationfile.

- **image-auth Storage hook (PHP)**

To automatically authenticate image-locations on storage servers openQRM checks if plug-ins providing the image-auth hook file at:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/web/openqrm-[deployment-type]-auth-hook.php
```

The different deployments types are defined by the specific storage plug-in. openQRM then runs:

```
storage_auth_function($cmd, $appliance_id)
```

Parameters in this hook function are:

```
$cmd - start, stop
```

```
$appliance_id - the appliance id
```

This function then needs to make sure to trigger an authentication update of the image specified in the appliance onthe storage server (where the image is located). For the “start” action the function makes sure to only allow the appliance resource to mount the image location from the storage. During “stop” it de-authenticates the image-location again.

- **root-mount hook (Shell)**

The root-mount hook is a shell file contains the “mount_root” function which is responsible to mount the root files-system from the storage server within the initrd-stage of a booting resource. This function is purely storage type dependent and defines "how" to mount the root-fs. The filename of this hook is:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/web/root-mount.[deployment-type]
```

It should be linked to openQRM's "boot-service" directory within the openQRM web.

Resources assigned to an active appliance download this file and run the "mount_root" function to mount their root file-system during boot-up. This hook can also define additional optional function to e.g., install-from-nfs or transfer- to-local.

Example:

The "iscsi" plug-in is using the root-mount hook to define how resources are booted directly from the iSCSI Luns ofthe target.

- **root-device identifier hook (PHP)**

To abstract images and make them selectable e.g., in the appliance configuration form storage plug-ins needs to provide an "root-device identifier" hook via the file:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/web/image.[deployment-type].php
```

The file implements the following function:

```
get_image_rootdevice_identifier($storage-id)
```

```
$storage-id - The openQRM storage id
```

The function should return an array of "[image-volume-identifier-on-the-storage], [image-name]"

The hook file can also implement "get_image_default_rootfs()" which then fills the "root-device type" input box in theimage-new form.

- **Plug-in monitor hook for reiterating commands (PHP)**

For reiterating commands openQRM provides the "plug-in monitor" hook which is executed every 10 seconds. openQRM looks for each plug-in if it provides a file named:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/web/openqrm-[plugin-name]-monitor-hook.php
```

If such file exist openQRM runs the function:

```
openqrm_[plugin-name]_monitor()
```

Example:

The "cloud" plug-in is using the openQRM monitor hook to frequently check for Cloud-events.

- **Cloud-Billing hook (PHP)**

To allow custom billing for the openQRM Cloud it provides a billing hook located at:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/cloud/web/openqrm_custom_cloud_billing($cr_id,  $cu_id,  $cu_ccunits)
```

Insert custom billing functions there.

- **High-Availability hook(PHP)**

In case of a resource error openQRM checks if plug-ins providing a high-availability hook. The hook file is named:

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/[plugin-name]/web/openqrm-[plugin-name]-ha-hook.php
```

If a plug-in provides this HA hook openQRM runs the following function implemented by the hook:

```
openqrm_highavailability_ha_hook($resource_id)
```

The function then is responsible to re-act in the “right way” to make sure no service interruption occurs. It is only executed if a resource is assigned to an active appliance.

Example:

The “highavailability” plug-in implements this hook to replace failed resources in active appliances.

Image Hooks

The mount-point configuration file /etc/fstab on server-images is created automatically according the images storage-location during the initrd-stage of the boot-up phase. For adding additional static mount-points create a file /etc/fstab-static on the server-image containing the additional mount-points. This file is automatically added to /etc/fstab during start-up.

Plug-in Boot-services

Each Plug-in can provide a Boot-service for the managed resources. This Boot-service consist of a packed Plug-in specific tools file which is being linked into the openQRM webspace at Plug-in Init. By the start phase of the openQRM-client this file is then being downloaded and unpacked by starting resources. In case this package includes a Plug-in init script the openQRM-client will also start the Plug-in on the resource. The Plug-in init script for managed resources should be located at

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/PLUGIN_NAME/etc/init.d/PLUGIN_NAME
```

within the Boot-service package. It should support the “start” and “stop” parameter.

Plug-in Services

Plug-in's also have the capability to start (and stop) additional services on the openQRM Server itself. The openQRMServer init script looks for Plug-in init scripts at

```
OPENQRM_SERVER_BASE_DIR/openqrm/plugins/PLUGIN_NAME/etc/init.d/openqrm-plugin-PLUGIN_NAME
```

on the openQRM Server. If this file is provided by the Plug-in openQRM will run it with the start/stop parameter during openQRM Server start/stop.

SOAP Web Services

Every Cloud Computing Environment needs to be integrated into a customer-management framework. Especially the external Cloud-Portal in openQRM plus its generic billing mechanism via Cloud Computing Units (CCUs) makes this integration easy. Still for a seamless integration with an existing customer-management application a WebService including a well-defined API was missing. The openQRM Team understood this need and implemented a SOAP-WebService for the openQRM Cloud. Its SOAP-Server, working in WDSL-mode, is separated into an “Cloud- Admin” and “Cloud-User” part. It is exposing the automatic provisioning/de-provisioning mechanisms to SOAP-Clients and can be used as a full “remote-control” for the openQRM Cloud.

A detailed documentation and huge examples for the “Cloud-Admin” and “Cloud-User” SOAP-Clients are included in the Cloud-plugin. The openQRM Cloud SOAP-WebService plus its documented API makes it very easy to integrate the openQRM Cloud with any third-party application.

The openQRM Cloud API-Documentation for the Cloud SOAP WebService can be easily generated using phpDocumentor [http://www.phpdoc.org/]. An online version of the openQRM Cloud API is available at: http://www.openqrm-ng.net/downloads/plugins/cloud/openqrm-soap-api/ [http://www.openqrm-ng.net/downloads/plugins /cloud/openqrm-soap-api/]

Coding Guidelines

Here the Coding and Design Guidelines of openQRM. The goal is it to keep the following specifications:

- KIS (keep it simple)
- run everywhere (support for every linux distribution, especially Debian+Ubuntu)
- write as less code as possible (less code = less bugs)
- using existing components (instead of providing own binaries/libs)
- Support multiple databases (db2, mysql, oracle, postgres)
- Comment source code !

Security Considerations

openQRM environments are based on a separated “management- and storage-network” plus one or more “public-network”. For production setups it is recommended to adapt a few security considerations to protect the “management- and storage-network” as described in the chapters below.

openQRM Security Concept

The openQRM Server consist of several secured subsystems to support managing mission-critical Datacentres.

SSL encryption for the UI access

All http communication between a client browser and the openQRM Server UI are fully encrypted via SSL [http://en.wikipedia.org/wiki/Transport_Layer_Security]. Also, the internal communication between the openQRM Client (e.g., for sending statistics) are using the SSL secured “https”.

Hint: If needed openQRM can still be setup using the (unsecure) “http” protocol by changing the

```
OPENQRM_WEB_PROTOCOL="https"
```

parameter in the main openQRM configuration file at:

```
OPENQRM_SERVER_BASE_DRI/openqrm/etc/openqrm-server.conf
```

to:

```
OPENQRM_WEB_PROTOCOL="http"
```

SSL encrypted command execution layer

The openQRM remote command execution subsystem is based on Dropbear [http://matt.ucc.asn.au/dropbear /dropbear.html], a tiny SSHD server automatically installed on behalf of the openQRM Client. It is using shared public-key mechanism to securely transmit commands to remote systems, encrypted by SSL.

Storage authentication

For every deployment action openQRM automatically makes to setup authentication for the affected volume(s) on the involved storage systems. It takes care that only the involved "resource" is able and allowed to mount the remote disk from the storage. This mechanism prevents situations where user A is able to mount users B disk image in a robust and secure way.

Defined command execution

In the openQRM Cloud the Cloud users do not directly interact with virtual- or physical systems in the Datacenter but all commands are pre-defined and executed on behalf of the openQRM Cloud. The openQRM Cloud is then just submitting the pre-defined commands to the "SSL-secured command execution" subsystem.

Securing the Network

Using firewalls between the management- and public-network

To improve network-security package-filtering firewalls should be placed between the management- and public-network. Via the appliance start- and stop-hook the firewall rules can be adapted dynamically according the network-traffic requirements from the appliance resource to the storage.

Using VLANs to physically separate management- and public-network

To further improve network-security VLAN tagging can be used to provide a full physical network separation for the managed systems in openQRM. Same as for the dynamic adaption of the firewall rules the appliance start- and stop-hook are good candidates to connect them to the network-switch console for automatic applying updated VLAN configurations according the network-traffic requirements from the appliance resource to the storage.

Securing your Services

Custom firewalling for Appliances

To implement custom firewall rules for the managed appliances it is recommended to create Puppet-recipes which sets up specific firewall configurations according the appliances services. Those Puppet-recipes can be manually assigned to appliances by the Puppet-Manager (part of the Puppet openQRM plug-in) or applied automatically using the appliance start- and stop hooks.

Automated Service Monitoring

To be best informed and up2date about the services running in the Datacenter openQRM offers and automated service mapping via the Nagios3 integration. This special Nagios mode will display exactly which network-services are available in the openQRM managed environment. For further, fine grained system- and service-monitoring it is recommended to use the Collectd- and Zabbix plug-ins available in openQRM.

Getting Support

OPENQRM AUSTRALIA PTY LTD is the main sponsor of the openQRM Data Center Management and Cloud Computing Platform openQRM Australia provides professional services and long-term support with “first hand” competence.

With a huge pool of experience in managing data centers in a flexible, consistent and transparent way, openQRM Australia enables IT organizations to efficiently provide their IT services in a robust, standardized, performant and high-available mode.

openQRM Australia consolidates the technical competence of the core members and developers of the openQRM Project, experienced with every detail of this open-source solution, to supply expert knowledge for custom, sustainable data center setups in best practice. openQRM Enterprise's focus is to lower the Total Cost of Ownership (TCO) for IT departments using a proven open-source framework.

Thanks