Community Experience Distilled

Getting Started with oVirt 3.3

A practical guide to successfully implementing and calibrating oVirt 3.3, a feature-rich open source server virtualization platform

Alexey Lesovsky

[PACKT] PUBLISHING open source community experience distilled

Getting Started with oVirt 3.3

A practical guide to successfully implementing and calibrating oVirt 3.3, a feature-rich, open source server virtualization platform

Alexey Lesovsky

[PACKT] PUBLISHING
open source
community experience distilled

BIRMINGHAM - MUMBAI

Getting Started with oVirt 3.3

First published: November 2013

Production Reference: 1151113

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78328-007-0

www.packtpub.com

Cover Image by Abhishek Pandey (abhishek.pandey1210@gmail.com)

Credits

Author
Alexey Lesovsky

Reviewers
René Koch
Andrii Salnikov

Acquisition Editor
Vinay Argekar

Commissioning Editor
Deepika Singh

Technical Editors
Pragnesh Bilimoria
Aman Preet Singh
Anand Singh

Copy Editors
Roshni Banerjee
Sarang Chari
Gladson Monteiro
Sayanee Mukherjee

Project Coordinator
Michelle Quadros

Proofreader
Ameesha Green

Indexer
Hemangini Bari

Graphics
Abhinash Sahu

Production Coordinator
Manu Joseph

Cover Work
Manu Joseph

About the Author

Alexey Lesovsky is a system administrator at a software development company that works with federal mobile operators. He has eight years of IT experience, which includes over three years on KVM virtualization technologies and products. His primary areas of interest are Linux, KVM virtualization products, system monitoring, and web infrastructure solutions oriented on a high load. This is his first book.

I want to thank my wife and son for their patience and understanding. Thank you! I love you very much.

About the Reviewers

René Koch is a senior solution architect and consultant focusing on open source virtualization, Linux, system management, and system monitoring. He started working with Red Hat Enterprise Virtualization and oVirt in 2010, and implemented various environments on the customer side. As part of the oVirt community, he is not only an active member on the oVirt mailing list, but also gives lectures about oVirt in Austria and Germany. Furthermore, he is the author of two open source projects. The Nagios plugin check_rhev3 is used to monitor the whole oVirt and RHEV environment with Nagios and Icinga. The Monitoring UI-Plugin is a user interface plugin for oVirt and RHEV, which integrates Nagios-based monitoring environments into the oVirt web administration portal.

Andrii Salnikov, a qualified systems engineer and lecturer in the fields of High Performance Computing (HPC) and networking, was born in 1986 in the small provincial town of Chernigov, Ukraine. In 2000, he moved to Kiev where he obtained higher education in applied physics and computer science, and currently resides there with his spouse and little daughter. In 2008, he started his career as a Cisco Networking Academy instructor on the industry's leading technologies and moved forward to gain an insight into other computer science fields. Today, he is recognized as a Cisco Certified Networking Associate Instructor Trainer and HP Certified Instructor. He has proved his qualifications by acquiring the widely recognized industry certifications, Cisco Certified Network Associate Security, Linux Professional Institute Level 1 Certified, Novell Certified Linux Administrator, Novell Data Center Technical Specialist, HP Accredited Integration Specialist Networking Infrastructure, and Extreme Networks Associate.

Along with teaching, he applies all his knowledge and skills to develop various IT products. His primary research and development is devoted to Grid and Cloud Computing technologies. He took part in the technical design and implementation of several HPC clusters connected to the grid infrastructure powered by the Nordugrid ARC middleware in the National Academy of Sciences of Ukraine. His Ph.D. thesis that is devoted to theoretical methods and applications of grid technologies has just been finished.

He is currently employed by Taras Shevchenko National University of Kyiv as an Assistant Lecturer in the Computer Engineering department of the Radiophysics faculty and as Senior Systems Engineer in the Parallel Computing laboratory of the Information and Computer Center.

www.PacktPub.com

Support files, eBooks, discount offers, and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

oVirt is one of the most dynamically developed tools for creating your own virtualization infrastructure. With great potential, oVirt combines modern and advanced virtualization technology. Based on KVM, it allows you to create very flexible configurations that are able to solve a variety of issues. If you are interested in virtualization, oVirt is a good choice.

This book will guide you through the process of creating and customizing your own virtualization infrastructure. Step-by-step, you will learn what you need to do to create a complete infrastructure. In addition, you will learn about more advanced features and how to use them.

What this book covers

Chapter 1, *Before you Begin*, introduces the overall architecture and the internal structure of oVirt. It also provides an overview of the system requirements that must be met.

Chapter 2, *Installing oVirt*, will guide you through the installation and initial configuration of oVirt Engine and virtualization hosts. We also look at the Administrator Portal, which is the main control center.

Chapter 3, *Configuring oVirt*, describes the process of setting up the environment, including the creation and configuration of data centers, clusters, and virtualization hosts. Particular attention is paid to the connection of storage to oVirt and the creation of logical networks.

Chapter 4, *Managing Virtual Machines*, talks about how to create virtual machines. oVirt allows you to create virtual machines in a variety of configurations. Here we look at templates—convenient tools for the rapid deployment of virtual machines.

Chapter 5, *Advanced Features*, talks about the additional features offered by oVirt, such as user management and resource quotas, pools with prestarted virtual machines and snapshots, live migration and high availability, and more.

Chapter 6, *oVirt Troubleshooting*, helps us understand what to do when problems occur, and the steps that need to be taken to solve them.

Appendix A, *NFS Storage Setup on CentOS*, covers how to set up the NFS server with CentOS Linux

Appendix B, *iSCSI Storage Setup on CentOS*, covers how to set up an iSCSI server with CentOS Linux.

What you need for this book

To work with oVirt, you need hardware that can run oVirt. oVirt installation is required on at least one computer with preinstalled CentOS/RHEL. It is also necessary to have a simple understanding of virtualization in general. Some of the work is done on the Linux command line, so you also need some basic skills to work with the console.

Who this book is for

This book is intended for technician or Linux system administrator who is interested in the scope of virtualization, especially KVM virtualization, this is the book for you. This book will be of interest to both beginners and advanced users as it reveals a wide range of issues related to oVirt.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: If any of the items are not correct, type `NO`.

Any command-line input or output is written as follows:

```
# yum localinstall -y http://dl.fedoraproject.org/pub/epel/6/x86_64/
epel-release-6-8.noarch.rpm
```

New terms and important words are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: In this menu, select **Administrator Portal**.

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Before You Begin

oVirt is a flexible, multifunctional, feature-rich, and holistic virtualization management solution that provides easy and practical virtual infrastructure management. The oVirt virtualization platform avoids complexities and significantly reduces the cost of implementing of virtualization and subsequent operations related to its infrastructure maintenance. The main goal of oVirt is to help users build easily managed, high-availability infrastructure for handling **VMs** (**Virtual Machines**). oVirt provides ample opportunities for the rapid deployment of desktop and server virtualization infrastructure. oVirt would be a good choice in the following cases:

- Combining various types of hardware in a single virtualization platform
- Establishing a center for managing VMs with GUI control
- Simplifying the management of large numbers of VMs
- Automating VM's clustering and load balancing
- Automating the hardware and VM's failover through live migration

oVirt is an open source software developed with backing from Red Hat, and is the base for **Red Hat Enterprise Virtualization** (**RHEV**). RHEV is a stable commercial product, while oVirt is an upstream product. Features from oVirt get merged into RHEV when tested and become stable. oVirt is a bleeding-edge development which RHEV is based on. oVirt is modern and very new but not as stable as RHEV. oVirt has no commercial support. RHEV is not as advanced but is stable and recommended for enterprise and mission-critical systems. oVirt also plays a part in the support and development of projects involving the following companies: IBM, Cisco, Intel, Canonical, NetApp, and SUSE.

The oVirt platform provides the following features:

Flexible management of virtualization infrastructure:

- Centralized management portal for administrative tasks
- Multilevel control that allows you to manage the physical infrastructure at the level of logical objects
- The ability to add existing virtual machines on existing servers into the oVirt environment
- Flexible user management with external directory servers

High availability:

- Tools for building fault-tolerant virtual machines
- Live migration tools to move virtual machines between physical hosts

Resource usage efficiency:

- Resource scheduler is able to dynamically maintain the balance of resources used
- The ability to control a potential reduction in energy costs for cooling
- Quotas and resource limitations

Fast deployment of virtual machines:

- VM's template management that may need to create and manage virtual machines
- Snapshots, cloning, and pre-started virtual machines that are ready for usage

Flexible storage management:

- Storage virtualization for consistent treatment of shared storage from any server
- Ability to use different types of storage

oVirt is a dynamically developed product that is based on modern technologies. oVirt is built on Linux and Libvirt (the official FAQ is available at `http://wiki.libvirt.org/page/FAQ)`. Libvirt is a tool for virtualization management that allows managing virtual machines hosted on Qemu/KVM, Xen, VirtualBox, and LXC. However, oVirt is focused on Qemu with a **Kernel-based Virtual Machine** (**KVM**). (For more information about KVM refer to `http://www.linux-kvm.org/page/Main_Page`.)

oVirt uses KVM that requires processors with hardware virtualization extensions (for more information on virtualization extensions refer to `http://en.wikipedia.org/wiki/X86_virtualization`), such as Intel VT-x or AMD-V. KVM supports x86 processors and has been ported to ARM, IA-64, PowerPC, and S/390 platforms.

oVirt can be used widely. We can set up a small experimental installation on the desktop or create a large infrastructure that will ensure the needs of the whole enterprise. This is achieved by supporting a wide variety of hardware platforms. The data storage level also supports different types of storage such as NFS, iSCSI, Fibre Channel, and GlusterFS.

oVirt will be of interest to engineers, system administrators, and professionals who work with virtualization and Linux. To start working with oVirt, you must have basic skills in working with the Linux console. For more information, you can visit the following resources:

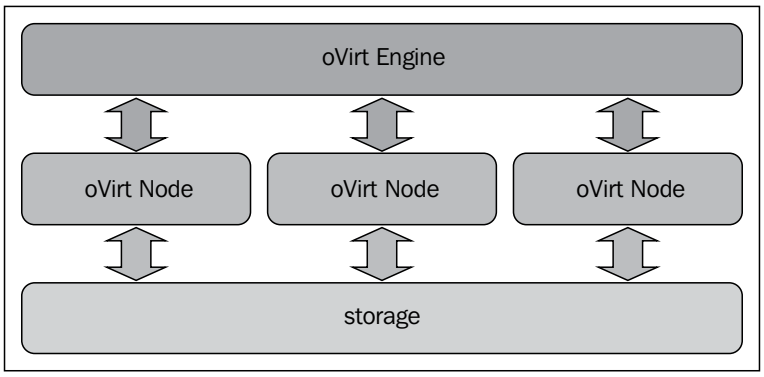
- oVirt related documentation at `http://www.ovirt.org/Documentation`
- oVirt official mailing lists at `http://lists.ovirt.org/mailman/listinfo`

Architecture overview

The oVirt platform consists of the following three logical components:

- **oVirt Engine**: It is a control unit used for administrative tasks related to the management of the global configuration of the entire virtualization infrastructure, the management of virtual machines, storage, and network settings.
- **oVirt Nodes**: It computes virtualization units that directly run the virtual machines.
- **Storage** and **network infrastructure** (external disk capacity units): These can be direct or network-attached storage (DAS/NAS) or high-performance storage area networks (SAN). Disk capacity units hold virtual machine images and OS installation images. Network devices, such as switches, provide connectivity between engines, nodes, and storage.

The oVirt architecture is as shown in the following figure:



oVirt Engine is a set of software and services that implement the functionality of the central control infrastructure. This control unit is platform's core and provides an interface for other infrastructure components. Using oVirt Engine interfaces, the administrator can run the whole setup inside oVirt. So with the help of oVirt Engine, we achieve one of the main goals of oVirt: centralized management.

Virtualization hosts (oVirt Nodes) are servers using Linux x86_64 with the installed **libvirt** daemon and **VDSM** (Virtual Desktop and Server Manager) (host-agent) service. These are the set of packages and support services that are required for the rapid deployment of virtualization. The most supported and preferred distributions to build the nodes is CentOS or Red Hat Linux. Also, we can use oVirt Node—a special stripped Fedora Linux minimalistic distribution containing only the necessary packages for integration into the oVirt platform.

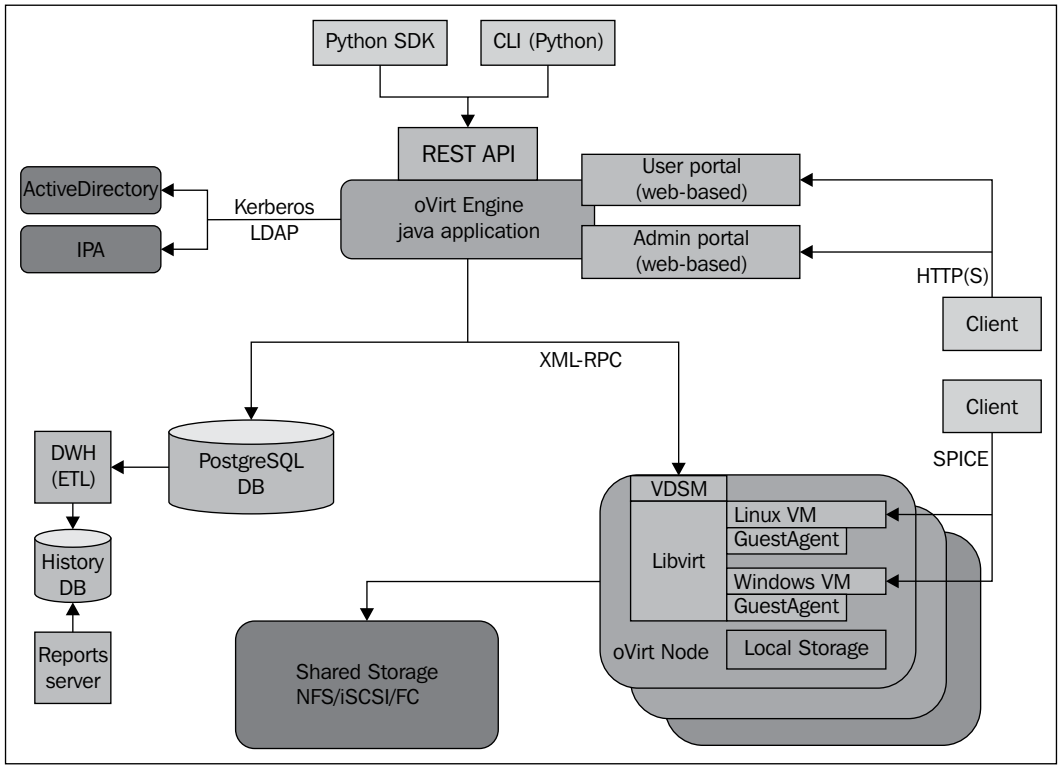
Storage is an external component of the oVirt infrastructure but is required for oVirt. However, we can use a local type of storage when the storage is located on the compute node. The storage nodes use block or file storage type and can be either local or remote, accessible via the protocols: **NFS** (additional information on **Network-Attached Storage** is available at `http://en.wikipedia.org/wiki/Network-attached_storage`), iSCSI and Fibre Channel (**SAN** information about **Storage Area Networks** is available at `http://en.wikipedia.org/wiki/Storage_area_network`). The cluster filesystem, **GlusterFS** (GlusterFS community `http://www.gluster.org/`) is also supported through a special type of storage called POSIXFS since in oVirt 3.3, available as an additional storage type. For most cases, NFS or GlusterFS storages are a good choice that do not cost much. GlusterFS storage nodes are grouped into a single storage filesystem known as *volumes* that guarantee high data availability and high data redundancy. Storage can be configured to replicate data. Thus, when a failure occurs with one of the GlusterFS nodes, the storage can continue its work.

oVirt provides the ability to simultaneously work with multiple types of storage. However, there is a significant limitation as a data center can use only one type of storage.

Additionally, oVirt Engine can be set to an external service identification and authorization such as **Active Directory** (the Active Directory wiki page can be found at `http://en.wikipedia.org/wiki/Active_Directory`) or **IPA** (FreeIPA's official website is `http://www.freeipa.org/page/Main_Page`) for user authentication. Such services are third-party in relation to oVirt and is not included in oVirt packages.

oVirt component relationships

We have reviewed the overall architecture of oVirt; however, we need to know the components of a platform and their purpose. It's especially important when troubleshooting. oVirt consists of several components, each responsible for a part of the work. It is shown in the following diagram:



At first glance, it might seem that the internal structure of oVirt is quite complex. However, this is not the case. Once you understand the basic components, which are given in the following points, it becomes clear that everything is simple.

- **Engine** (**oVirt Engine java application**): It controls the configuration of nodes and oVirt and is engaged in the management of virtual machines.
- **Host agent** (**VDSM**): It is an oVirt agent on the side of working nodes that executes commands which come from oVirt Engine.
- **Admin portal**: It is a web application that provides a user interface for administrators.
- **User portal**: It is a simplified user interface that provides access to the functions you need for the most frequently occurring tasks.
- **Database**: It is the PostgreSQL database that oVirt Engine uses for permanent storage platform configuration.
- **Guest agent**: It is a special agent running on the virtual machine that provides information on resource usage.
- **Data Warehouse**: It is the **Data Warehouse** (**DWH**) component based on Talend data integration software that performs **ETL** (**Extract Transform Load**) processing, and its results are loaded in the dedicated database history. Additional information related to Talend software can be found at `http://www.talend.com/products/data-integration`.
- **Reports engine**: It is a module for creating reports based on the data stored in the database history.
- **Remote SPICE or VNC client**: These are the utilities through which you can access the virtual machines' console.
- **REST API**: It is the programming interface that you can use to perform tasks for third-party applications that use a command-line interface and Python SDK.
- **CLI/SDK**: It a command-line interface, and its development tools provide a means of interaction with oVirt Engine.
- **ActiveDirectory/IPA**: It is a directory service. Optionally, oVirt can use the directory service to obtain information about users and groups and their further use in their own access schemes.

oVirt Engine is a Java-based application running as a web application. This service communicates directly with the VDSM agents placed on virtualization hosts. It is a scalable, centralized management tool for server and desktop virtualization, which internally uses modern technologies such as JBoss application server (for more information visit `https://access.redhat.com/site/documentation/JBoss_Enterprise_Web_Server/#`) , Java, and Python programming languages. oVirt Engine provides the following functions:

- Virtual machines' full life cycle management
- Authentication with LDAP providers (ActiveDirectory or IPA)
- Network configuration management is used for the creation of logical networks and connecting them to the hosts
- Storage management is used to manage domain's storage (NFS, iSCSI, Fibre Channel, GlusterFS, or Local) and disk images of virtual machines
- High availability functions to automatically restart virtual machines on other nodes if there is hardware or network failure of the source host
- Live migration manages the movement of VMs between physical servers without downtime
- System Scheduler is used for the implementation of load balancing of virtual machines based on resource usage policies
- Image Management is used for allocation based on templates, thin provisioning, and snapshots
- It uses objects' platform monitoring, such as monitoring virtual machines, hosts, network environment, and storage

As we can see, this component provides most of the functionalities inherent to oVirt. Engine is the core of the system through which the main oVirt goal is achieved—centralized management and task automation.

Admin and User portals

Admin portal is a convenient web-based graphical administration interface designed for centralized infrastructure management and the administration of virtual machines. Admin portal allows you to manage all aspects of your virtual infrastructure, ranging from the creation of data centers and clusters to the maintenance of the VMs' life cycle. Admin Portal is an intermediary between the administrator and oVirt Engine, all of the commands sent to the administrator use oVirt Engine for direct implementation.

User portal is a simple graphical interface that provides access to the life cycle management of virtual machines. User portal does not contain the global configuration settings, and allows you to influence the work of the entire infrastructure. Through the User portal, we can start and stop virtual machines, template management, and simple monitoring.

User and Admin portals allow you to achieve another goal, that is, to provide a convenient interface to manage the virtual infrastructure.

VDSM agents

oVirt Node or compute virtualization node implements the VDSM component (Host Agent). This component is developed in Python, which provides all of the functionalities of oVirt related to the compute nodes. The tasks assigned in the admin portal are passed to oVirt Engine and then transferred to the agent VDSM for direct execution. The tasks can be very different, such as creating or running a virtual machine, starting a migration, adding or removing a VM device, and reconfiguring the network environment. The host agent for virtual environments uses libvirt—virtualization management library. Thus, it can be concluded that VDSM agents perform all of the work associated with virtualization.

Guest agents

Guest agent's component provides communication between the host system and the virtual machine through **VirtIO** connection. A VirtIO (VirtIO para-virtualized drivers are available at `http://www.ibm.com/developerworks/library/l-virtio/`) serial channel is connected to the host via a character device driver. Guest agent provides additional information for oVirt Engine, for example, information about the use of memory, devices, and the internal state of the guest environment. Based on the data obtained, the engine gets an idea of what is going on inside the infrastructure.

Database

Database storage based on PostgreSQL RDBMS is used to store different information related to oVirt. This is used for global configuration, configuration of virtual machines, clusters and datacenters, and various journals and statistical data accumulated in the process. It uses the database achieved in the goals—centralized configuration storage.

Data Warehouse

Data Warehouse (**DWH**) is an internal component required to process the data that the reports will be built on. It is the data that is associated with the work of data centers, clusters, and virtual machines. This resource usage data is collected at the time of working with oVirt. Data is stored in a PostgreSQL database and can be used for further processing. The DWH **ETL** (**Extract Transform Load**) component uses the mechanisms of the *Talend* company related with data integration. The DWH component periodically collects data from the underlying PostgreSQL database for processing and then folds into a separate history database data ready for reporting.

Report Engine

Report Engine is a tool to generate reports on the use of resources within the infrastructure based on Jasper Reports (Jasper Report's official website is `http://community.jaspersoft.com/project/jasperreports-library`). It is an open source reporting engine. Written in Java, Jasper Reports can use different data sources for creating reports. This is quite a flexible tool that lets you add or export different types of reports. Report Engine is quite functional and provides features such as the creation of scheduled reports and filters, export to various formats, and special tools to create reports. Reports can show a variety useful information. Based on reports, we can perform analysis and make predictions about how resources are used and how the infrastructure can be developed.

Developer interfaces

The developer interfaces are REST API and CLI SDK. A REST-oriented programming interface is used for integration with oVirt Engine. This interface can be used with any programming language for various oVirt tasks (creating oVirt resources, VM's management, and so on). A programming language that is used must be able to perform HTTP requests and deal with XML. **Software development kit** (**SDK**) is available, which is written in Python and Java. With the SDK, we can perform operations on objects in oVirt while providing a complete abstraction of the protocol; it is fully compatible with the architecture of oVirt API. A development kit is simple to use and easy to learn. The **command-line interface** (**CLI**) is also written in Python, and provides information and performs various operations on objects in oVirt. Like the development kit, CLI is also fully compatible with oVirt API and is intuitive by nature.

Hardware and system requirements

This section describes the minimum and recommended hardware requirements that must be met for successful installation of the oVirt platform. In order to deploy the oVirt environment, it will require several components that will act in different roles. Depending on the role of the executable, it can be a dedicated server or a workstation. We will need the following components:

- One machine will act as the server manager.
- One or more machines are used as virtualization nodes—we need at least two machines to perform a live migration or for the implementation of power management features.
- One or more machines will act as a client to access the administrator portal. This may be a simple desktop workstation.
- Storage infrastructure that will provide its storage in one of the supported protocols (NFS, iSCSI, FibreChannel, GlusterFS, and POSIX).

The most difficult task is the selection of the storage infrastructure; it is often the most expensive component of infrastructure virtualization. Shared storage can be represented by a variety of equipment. These can be dedicated storage servers, NAS, or SAN infrastructure that can provide resources for oVirt. Shared storage must be available and enabled for virtualization hosts.

It is clear that the requirements for a more complex configuration require a more careful approach and analysis of the requirements to make the final decision. However, there are some simple and universal rules:

- A large number of VMs require more virtualization hosts
- A large amount of disk space inside VMs requires a large capacity of shared storage
- Good performance requires high performance of hardware (network bandwidth, storage and disk performance, CPU, and so on)
- More memory is better

The following requirements are the lower limits below which oVirt performance will be poor.

Management server requirements

To install the management server, we need:

- A dual or a quad core CPU
- 4 GB system RAM that is not being used by existing processes
- 10 GB of locally accessible and writable disk space
- 1 **Network Interface Card** (**NIC**) with bandwidth of at least 1 gigabits per second
- An installed and running x86_64 operating system such as CentOS 6, RHEL 6, or Fedora 18/19

When you create a management server, it is worth bearing in mind that the load on this server is not constant, so increasing the number of cores does not significantly improve its performance. However, as we know memory is not enough for oVirt services to work comfortably, 4 to 8GB RAM will be enough. Disk performance of SATA drives will also be quite important as the disk is operating on the management node; there is no constant load on the disk I/O. Gigabit network cards have become an integral part of the server hardware, so the bandwidth should not pose any difficulties.

Virtualization host requirements

Servers with the role of virtualization hosts should also be suitable for certain minimal requirements:

- A dual or quad core CPU
- 4 GB system RAM that is not being used by existing processes
- 25 GB of locally accessible and writable disk space
- 1 NIC with bandwidth of at least 1 gigabits per second
- An installed and running x86_64 operation system such as CentOS 6, RHEL 6, or Fedora 18/19

Virtualization hosts must have at least one CPU that must support the Intel64 or AMD64 CPU extensions and the AMD-V or Intel VT hardware virtualization extensions. In this case, more the number of CPUs, the better it is. More number of CPUs or cores means that we can start more VMs simultaneously. Currently, a maximum of 128 physical CPUs per virtualization host is supported. To check whether your processor supports the required virtualization extensions and that they are enabled in the Linux terminal, enter the following command in the terminal:

```
# grep -E 'svm|vmx' /proc/cpuinfo
```

This command searches for the presence of flags that define virtualization support in the installed CPUs. No output indicates that the hardware virtualization is not supported, which is bad. We should also check the BIOS settings, as some manufacturers, by default, turn off support for hardware virtualization.

RAM requirements

In the case of memory, the phrase "No amount of memory is too much" is fully justified. The minimum amount of memory required by a virtualization host is 4 GB (although oVirt developers suggest to keep the value as 10 GB, you can start with a smaller RAM size); however, it all depends on how many virtual machines and resources they configured with. More VMs need more memory! There is a fairly universal rule that specifies that the amount of memory for the host system is the sum of all the amounts of VM's memory and 2 GB RAM for the host operating system itself.

It is also known that KVM can perform physical memory overcommit. This allows you to allocate for the virtual machine memory size exceeding the amount of the available physical memory. However, we must be very careful because the simultaneous peak load on these virtual machines can lead to swapping and significant performance degradation—use with care. Also, oVirt currently supports the control of memory ballooning and KSM, which are allowed dynamically to adjust the memory usage of virtual machines and effectively utilize memory by combining duplicate pages. Memory is added and the upper limit is 1 TB RAM per virtualization host.

Storage requirements

Normal operation of virtualization hosts requires a local storage that will store host configuration files, work logs, core dumps, and more. The recommended amount of space for local storage is 10 GB.

The minimum supported internal storage for each oVirt Node is the total amount of memory required to provision the following partitions:

- Root partition with operation system files with at least 2 GB
- Logfiles storage with at least 2048 MB
- Data storage with at least 1 GB
- Swap partition with at least 2 GB

Swap partition can also be determined by the following formula:

TOTAL_SWAP = (TOTAL_RAM x 0.5) + 4 GB

PCI device requirements

Physical virtualization host must have at least one network card with the speed of at least 1 gigabits per second for normal tasks' execution, which is directly dependent on network bandwidth, such as live migration. In large oVirt installations where intensive exchange with the shared storage is expected, it is also recommended to use a high-speed network connection. If possible, it is recommended to use two network cards. One network card can be used for the main production network and the second network card can be connected to a separate management network. Alternatively, we can aggregate network cards into a bond device for network connection backup or performance reasons.

Worst case

It may happen that we have no free physical servers and really want to try oVirt. Well, we can use a virtual server in an existing virtual environment. Yes, oVirt Engine can run inside a virtual machine. To run virtualization hosts, our physical virtualization server must support and have **nested virtualization** enabled (all modern processors are able to do this); remaining are the storage nodes. Take a virtual machine with large virtual disks and run the NFS server. It can't be used in production of the oVirt deployment, but it is quite suitable for experiments. Additional information about nested virtualization can be found at `http://www.ibm.com/developerworks/cloud/library/cl-nestedvirtualization/`.

Summary

In this chapter, we discussed the basic theoretical issues to be aware of when installing oVirt. When we install any complex system, it is important to know how it works from the inside; this understanding further simplifies working with the system and helps to better understand the subtleties. Knowledge of the overall architecture and system requirements will help in choosing hardware and better use of available resources. Information about the components and their interaction provides you with a clear understanding of how to further optimize the use of resources in respect of each to the components. In *Chapter 2*, *Installing oVirt*, we move on with the installation of oVirt Engine and the virtualization hosts for oVirt. We will have a detailed look at the main stage of the installation and the initial configuration of oVirt.

2
Installing oVirt

This chapter describes the process of initial configuration and installation of oVirt Engine packages and oVirt virtualization hosts. Before we begin the installation, we need to answer some simple questions. The installation process takes a bit of time but at the same time it is clear enough. After installing the oVirt Engine, we consider the basic graphical user interfaces of oVirt Admin portal and user portal.

oVirt Engine setup

oVirt installation can be divided into two stages: the first is the oVirt Engine package installation and second is setting the oVirt virtualization hosts. Installing oVirt Engine is primary and most important. The installation of the virtualization host is secondary and can be done later or even postponed. This is due to the fact that the virtualization hosts are a variable component in the oVirt environment. Virtualization hosts can be dynamically added or deleted at any time.

oVirt Engine installation

The main Linux distribution in line with the evolving oVirt is CentOS Linux. However, due to the open model of development with other distributions, installation packages are present for other distributions as well. But unfortunately, the current versions may have some delay which is difficult to predict. As mentioned previously, we will consider installing CentOS Linux distribution because this is the native distribution for oVirt. The most profitable option of settings to install CentOS Linux is minimal install. Minimal install assumes that our system has the minimum number of packages in the system and is running the minimum number of services. This will avoid potential conflicts during installation. Installation of packages is performed using the yum package manager from the official repository of oVirt or from official mirrors.

The following steps show how to configure the host with CentOS Linux before installing oVirt Engine:

1. For setting up the hostname, it is worth remembering that the hostname must be resolved by direct and reverse DNS lookups as while this condition is not met engine-setup configurator cannot continue. To adjust the forward and reverse lookups, you should contact your DNS administrator or run your own DNS server and configure your own private domain. This hostname must also be saved at the path `/etc/sysconfig/hostname`. If the hostname has not been configured, then it's time to do it; use the following command:

   ```
   # hostname ovirt-engine.example.com
   ```

2. For setting up oVirt and EPEL repositories configuration, it is necessary to define the repositories configuration from which yum will download software. Use the following command to define the repositories configuration:

   ```
   # yum localinstall -y http://dl.fedoraproject.org/pub/epel/6/
   x86_64/epel-release-6-8.noarch.rpm
   ```

   ```
   # yum localinstall –y http://resources.ovirt.org/releases/ovirt-
   release-el.noarch.rpm
   ```

   Installed repositories configuration provides several repositories, such as stable, prerelease alpha, beta, and nightly builds. For now, we are interested in the stable release. Configuration is set up in such a way that the installation of packages from this repository is disabled. If you need to install packages from that repository in yum, you need to pass the parameter `--enablerepo=ovirt-nightly`.

   It should also be noted that the packages are not signed by PGP. Be careful as the GPG check will not run during the installation. Use oVirt packages from trusted official sources only.

3. Now do a system update. Updating the system is not a mandatory step, but it guarantees that your system has the latest version of packages. Use the following command to perform a system update:

   ```
   # yum update –y
   ```

4. To install oVirt packages, run installation of the oVirt Engine package and related dependencies using the following command:

   ```
   # yum install ovirt-engine -y
   ```

oVirt is also available in an all-in-one mode. This is a very useful opportunity that allows you to run oVirt when you have limited resources. Additional info can be found at the link `http://www.ovirt.org/Feature/AllInOne` in the **Installation flow** section.

Once the command is complete, we can go to the initial configuration.

Initial configuration

After the package installation is complete, oVirt Engine must perform the initial configuration. In this initial setup, we configure the services and the services themselves will be automatically started. To start the setup process, use the command for engine setup, which provides the interface to determine the configuration. The `engine-setup` command will ask a few questions that must be answered. Based on these answers, an oVirt configuration will be created. All the required services are started after this.

If we decide that the installed configuration does not suit us, we can clean it with the `engine-cleanup` command. However, this command does not remove all of the previously created configuration. So after `engine-cleanup`, we should delete the directory that was defined as a repository for the `ISO` and clean the path `/etc/exports` of oVirt NFS shares. The steps are given as follows:

1. For starting the `engine-setup` script, start the configuration process we need to run the engine setup as root:

   ```
   # engine-setup
   ```

   Setting the initial configuration takes place in the form of a dialogue in which the installation script asks us questions and we need to give accurate answers. During the dialogue, `engine-setup` asks questions and at the same time, offers a default answer. We can accept the default answer and press *Enter* or enter our own value.

2. For configuring the network, the first step is to enter the *fully qualified domain name* and configure the firewall. During the operation, the script tries to automatically determine the fully qualified domain name. A certain value will be offered as an option by default. However, you can refuse the offered option and enter another option. It is important to specify the name that was successfully allowed for the forward and reverse DNS lookup. In our example, we will use the `example.com` domain:

   ```
   Host fully qualified DNS name of this server [ovirt-engine.
   example.com]: ovirt-engine.example.com
   ```

```
iptables was detected on your computer. Do you wish Setup to
configure it? (yes, no) [yes]: yes
```

3. For configuring the database, install the database and define a password to access the database. In this step, we can choose local or remote database location. Enterprise users who have dedicated database servers can choose remote database installation. If there is no dedicated database server choose local database installation. In this case, the database will be installed on the same host with the oVirt Engine. In this example, a local database installation will be chosen:

   ```
   Where is the database located? (Local, Remote) [Local]: Local
   Setup can configure the local postgresql server automatically for
   the engine to run. This may conflict with existing applications.
   Would you like Setup to automatically configure postgresql, or
   prefer to perform that manually? (Automatic, Manual) [Automatic]:
   Automatic
   Using existing credentials
   ```

4. For the oVirt Engine configuration, we need to include the admin password setup, determining application mode, and specifying storage type for the default data center.

   - First we have to create an internal domain authentication for the default administrative account. Created domain authentication is named `<<internal>>`, and the administrative account name is `admin`. In this step, you need to set the password for the administrative account:

     ```
     Engine admin password:
     Confirm engine admin password:
     ```

   - Next we must determine the oVirt Engine application mode. This option allows users to configure the engine for virtualization management only, GlusterFS management only, or both. When only Gluster mode is used, it uses GlusterFS bricks management and doesn't allow to create and start the VMs. When the Gluster option is chosen, the engine setup will skip questions related to storage type for the default data center and NFS. In our example, we will select both modes, which allows virtualization and Gluster management. This is shown in the following command;

     ```
     Application mode (Both, Virt, Gluster) [Both]: Both
     ```

- We also need to specify storage type of the data center that will br created by default and named as `Default`. Here we can specify the type of storage we want to use. In this example, we choose the NFS type but in the next chapter we will consider the creation of data centers with the remaining types of storage:

  ```
  Default storage type: (NFS, FC, ISCSI, POSIXFS) [NFS]: NFS
  ```

5. For the PKI configuration, we need to create our own **certification authority** (**CA**), which will be used for security reasons. This CA is used for issuing certificates that can be used for client and server identification (application server, VDSM, and SSH authentication). The set value for organization name is used in the certificates. In our `example.com` domain name will used in certificate, as shown in the following command:

   ```
   Organization name for certificate [example.com]: example.com
   ```

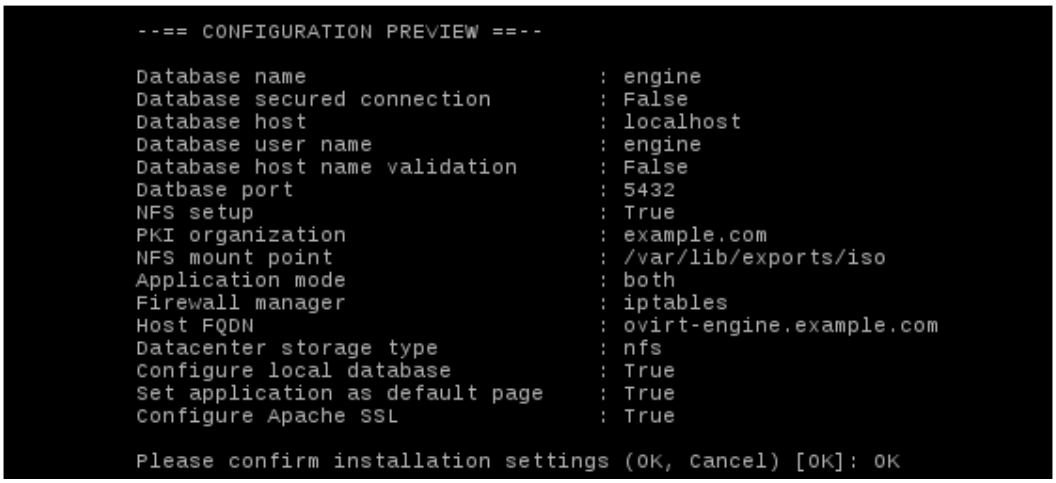
6. For the Apache configuration step, we need to set up the Apache web server. Apache is a frontend for our engine application which is run by the JBoss application server. This step helps prevent damage to the existing HTTP server configuration. If HTTP server is not configured, we can choose automatic installation. Also, we will perform the issuing and signing of the certificate during this step, as this is required for the HTTP service. We can choose issuing and signing through the previously created CA. For enterprise users who have their own enterprise CA, they can choose the `Manual` option and perform this operation manually. In our example, we choose default action with the automatic setup:

   ```
   Setup can configure the default page of the web server to present
   the application home page. This may conflict with existing
   applications.
   Do you wish to set the application as the default page of the web
   server? (Yes, No) [Yes]: Yes
   Setup can configure apache to use SSL using a certificate issued
   from the internal CA.
   Do you wish Setup to configure that, or prefer to perform that
   manually? (Automatic, Manual) [Automatic]: Automatic
   ```

7. For configuring the system, we can create a local NFS `ISO` domain on the current server as a shared NFS directory. This NFS share is used to store the `ISO` images that can be used to boot the virtual machine. oVirt Engine can be used as an `ISO` domain—as a local store that is located on the same server as the oVirt Engine—and as remote storage. In both cases, the `ISO` storage must be available for NFS. If the dedicated NFS storage is used, we can skip this step and perform the setup later. Best practices recommend us to use an external `ISO` storage. When an oVirt Engine system update occurs (related to a Kernel or glibc update), the Engine host should be rebooted. At this time, if a local `ISO` storage is used, it becomes unavailable. This is not critical, but we should know about this moment. In our example, we will configure a local NFS share as shown in the following command:

```
Configure an NFS share on this server to be used as an ISO Domain?
(Yes, No) [Yes]: Yes
Local ISO domain path [/var/lib/exports/iso]: /var/lib/exports/iso
Local ISO domain name [ISO_DOMAIN]: ISO_DOMAIN
```

8. To confirm the configuration, the final step is to confirm the information entered. Before we start setting, the engine setup will display the values entered and ask for confirmation. The following image shows the confirmation dialog box:



If the entered information is correct, enter `OK` and press *Enter*. If any of the items are not correct, type `NO` and press *Enter* to refuse the configuration process.

As a result, engine-setup will configure and run the necessary services. It will take some time for this and in any case, it does not interrupt the configuration process. At the end, the engine setup will display additional information which should be read carefully.

oVirt virtualization hosts setup

For further platform deployment, we need at least one virtualization host. Of course, it is better to have two virtualization hosts. If we have more than one virtualization host, we can use the advanced virtualization features such as live migration and high availability. Install the virtualization hosts to utilize the physical servers. oVirt Engine as well as the virtualization hosts must be registered in DNS.

So the next step is to install, or to be more precise, prepare the server for the role of virtualization host. What does this mean? In earlier versions of oVirt, the virtualization host setup could be run in the following two ways:

- Install the packages' VDSM on preinstalled Linux, CentOS/RHEL being the recommended package
- Use a special Linux distribution called oVirt Node, which in fact is stripped Fedora Linux

Now we can start the process of initializing the virtualization host directly from the administrator portal. While running the installation procedure, oVirt Engine independently connects to a remote host and performs all the necessary steps, such as downloading and installing packages, adding and configuring services at the startup and then rebooting the remote host and adding it to the oVirt infrastructure. It is very convenient and fast.

In cases when oVirt Node is used, we can only setup oVirt Node, and later add the host to the cluster in the admin portal.

When using preinstalled CentOS or RHEL, we need to install the RPM packages with the repositories configuration. These are oVirt and EPEL repositories. EPEL is required because it stores some oVirt dependencies (novnc, python-ply, and so on) in this repository. In general, this is a very useful repository. Note that the GlusterFS repository already contains the oVirt repository configuration.

Repositorie's configuration will be installed with yum, this step is similar to the repository setup for oVirt Engine that can be found in the previous section *oVirt Engine installation*.

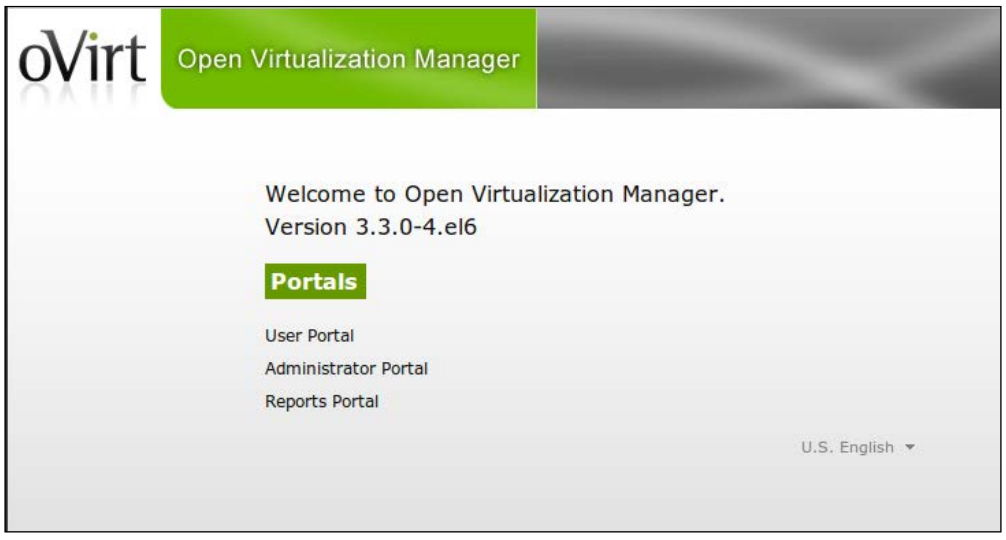
This may seem surprising, but with this host, preparation is completed. Further work on setting up the virtualization host can be done from the administrator portal. But installing the engine and virtualization hosts is a small part of the global process as a general configuration oVirt. Therefore, they should get acquainted with oVirt administrator portal.

If you don't have a a storage device ready, you can go to *Appendix A*, *How to Setup NFS Storage*, and *Appendix B*, *How to Setup iSCSI Storage*. GlusterFS storage creation will be discussed in the next chapter.

Administrator portal

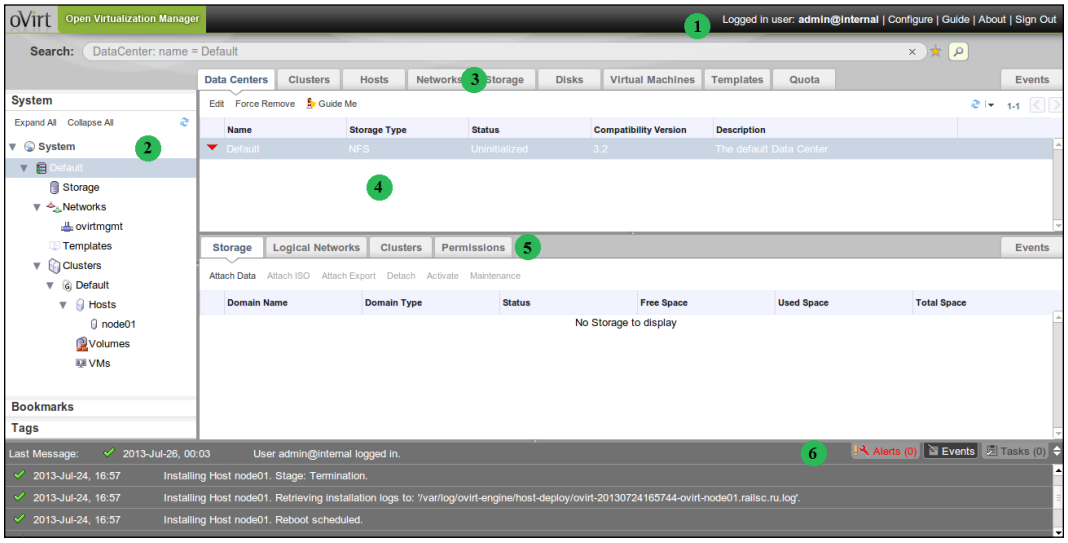
Before we start configuring oVirt, we should become familiar with the administrator portal. This interface provides administrators with a management system that allows managing virtual machines, creating and organizing VMs within the platform datacenters and clusters, managing virtualization hosts and storage, and more.

To get started with the administrator portal, we need to enter the address of the server with oVirt Engine in the web browser that we pointed out when we configured it with the engine setup. When we first try to connect, we will see a menu of three items as shown in the following screenshot:



In this menu, select **Administrator Portal**. Then enter the admin username and password that you specified during the engine setup configuration.

This is the main control center of the oVirt platform where we can configure and manage all the resources available to us. The following image will show the **Administrator Portal** main window:



The important components of the **Administrator Portal** window are explained as follows:

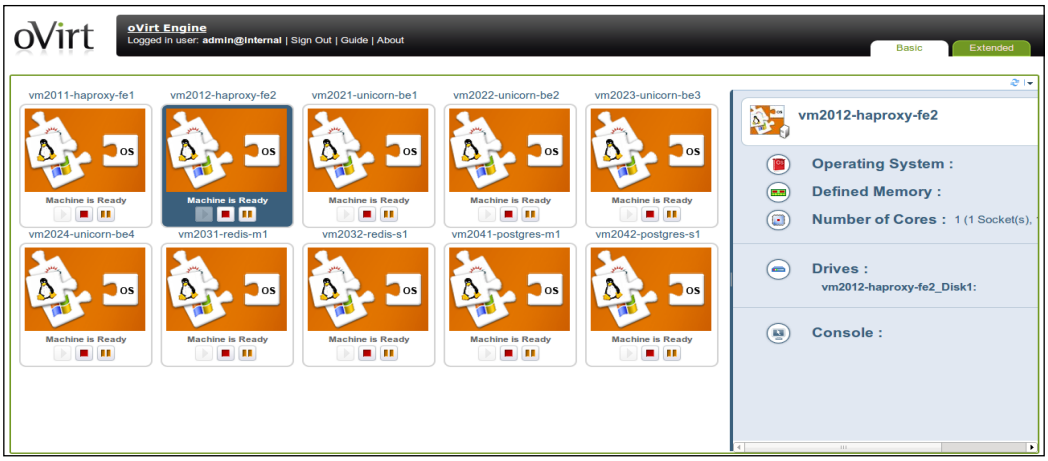
- **Header**: This panel is located at the top of the screen and it displays the name of the registered user, an exit button, and the option to set up user roles. With this option, we can create new users and define their roles.
- **Navigation pane**: The navigation bar allows us to navigate between the **Tree**, **Bookmarks**, and **Tags** tabs. In the **Tree** tab, tree mode displays the overall look and provides a visual display of the entire virtualization environment. **Bookmarks** allow us to jump to the most used places. And **Tags** are used to mark the virtual machines to make it convenient to sort.
- **Resource tabs**: These tabs provide access to oVirt resources. In oVirt, there are several types of resources that make up the whole virtualization platform. Datacenter is on the top—all resources are grouped by datacenters. Datacenters are divided into clusters and have attached storage. Clusters contain virtualization hosts. VMs are running on virtualization hosts. VM's virtual disk stores data on the attached shared storage that belongs to a data center. All of these resources can be accessed using the corresponding resource tabs.

- **Resource list**: By selecting a specific tab in the resource tabs, this area displays a list of the available resources, for example, hosts, clusters, VMs, disks, and so on. The resource list area can also be accessed by the buttons to add or remove specific resources. For each resource, we can make specific operations that are available on the toolbar (or on the context menu) in this area. For example, the virtualization hosts have the ability to move from an active state to the maintenance mode and back, or VMs can be started, stopped, or migrated.
- **Details pane**: When you select a resource in the resource list, an additional panel is activated that displays the details of this resource. Details are grouped in several subtabs, using which, you can perform various operations for the resources. This operation is usually defined by the interaction, or the relationship of the resource with the other resources.
- **Event pane**: At the bottom of the screen, the events panel is located, which displays all the events taking place in oVirt. When we perform an operation, the status and progress can be monitored in the event pane. For ease of use, the event pane has the following three tabs:
    - **Alerts**: It shows the alert events, errors, and accidents
    - **Events**: It displays all the events occurring in the system
    - **Tasks**: It is used to display the tasks that are running

Administrator portal fully covers the infrastructure and thus achieves centralized management. Using the administrator portal, we can add or delete resources to change the configuration and affect the whole oVirt platform.

User portal

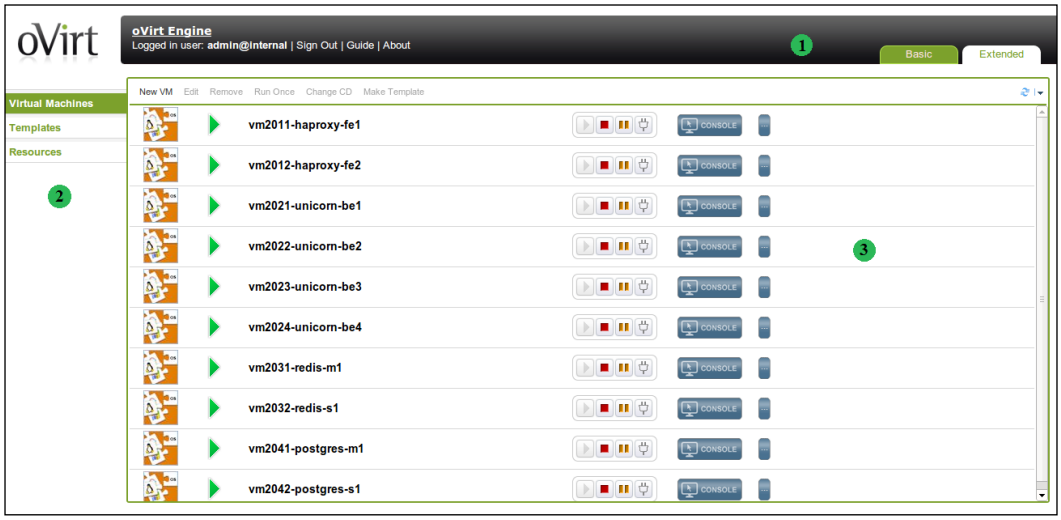
The oVirt user portal provides standard and advanced user access to the oVirt environment. The User portal is needed in cases where users do not need to provide access to administrative functions. Here we can create and run virtual machines and create templates based on these machines. Structurally, a user portal also consists of two views: **Basic** and **Extended**. The **Basic** view is used for virtual machine selection and further connection to a VM's consoles via SPICE/VNC. The following image show the **Basic** view of the user portal:



The **Extended** view is used for creating VMs and consists of several sections given as follows:

- **Header**: Displays information about a registered user, contains a button, and sends out information to the guide.
- **Content pane**: This panel is divided into three sections and defines the basic content which we can work within the user portal:
    - **Virtual Machines**: In this section, we are working with virtual machines. We can create, edit, run, delete, and create a template from a virtual machine—the basic operations associated with the life cycle of virtual machines.
    - **Templates**: This section is associated with templates and their management; it allows you to edit or delete the existing templates.
    - **Resources**: It is a summary screen which displays the current load and resource utilization. It is used for monitoring and surveillance of the overall state of resources.
- **Content List**: This is the main area with a list of VMs or templates. On selecting one of the sections on the content pane, this area displays a list of virtual machines, templates, and resource usage on the status screen.

The **extended** view allows us to create and running virtual machines. It allows us to create, add, and edit templates, and helps us to monitor the created machines. The following screenshot shows the **Extended** view of the user portal:



The user and administrator portal are the main interfaces that we will have to configure and work with oVirt. Now we have a clearer idea about the graphical user web-interface, we can begin setting up the oVirt infrastructure.

Summary

In this chapter, we covered the practical side of using oVirt. We installed oVirt and performed the initial configuration. After completing the installation and configuration, we must configure future virtualization hosts and possibly shared storage. However, this procedure can be performed at any other time. Instead, quickly run the administrator and user portals to get a hands-on approach of oVirt is now ready for further configuration of the virtualization environment. We will learn more about the configuration of oVirt in *Chapter 3*, *Configuring oVirt*.

3
Configuring oVirt

After the installation of oVirt Engine through the engine setup utility, the next step is to configure the oVirt virtual environment. Further configuration is done using the already familiar administrator portal. The administrator portal performs the integration of existing resources in a virtual infrastructure. The oVirt environment is formed from a set of logical components. However, these components are very real physical devices, given as follows:

- **Data centers**: They are the top-level containers and may incorporate several clusters
- **Clusters**: These are a group of hosts with some properties
- **Hosts**: They are the physical servers on which the virtual environments will work

In addition to these components, there are some more components too: **Logical networks**, **Volumes**, and **Storage**. An important element is **Storage** that connects to the data center. The type of data center and Storage is determined during configuration of oVirt Engine or in the process of creating the data center. Dedicated storage is used to store virtual disks that belong to virtual machines. oVirt's significant advantage is that it supports different types of storage facilities, and it can create additional data centers using various types of storages. Note that the data center can only work with one type of storage. Supported types of storages are NFS, iSCSI, Fibre Channel, and GlusterFS.

Networks are an essential component of any data center, both physical and virtual. Logical networks can be used for separation of traffic and building complex network environments. Volumes in oVirt are preconfigured and ready-to-use GlusterFS storage, which are distributed across virtualization hosts.

Data centers

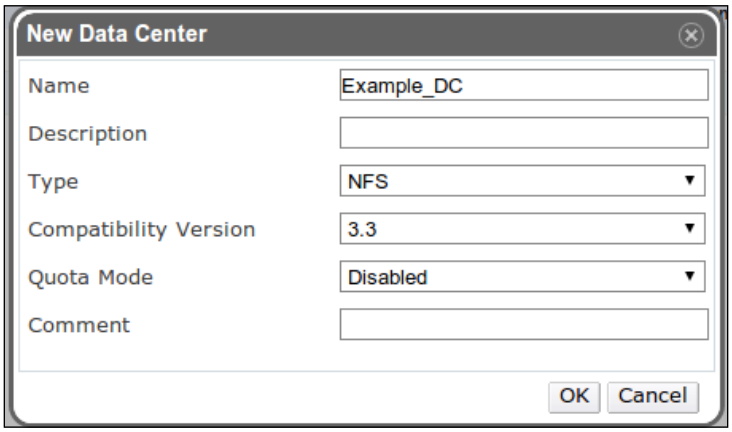
Data centers are top-level containers for any resource available in the oVirt environment. Data center consists of a set of clusters, which are composed of sets of hosts. The data centers contain logical networks and various types of storage. After the completion of the engine setup in our infrastructure, we create a data center named `Default` and a preconfigured cluster, also named `Default`. This predefined data center and cluster are a good starting point for getting started. We can even use the predefined data center. We will show how to build the infrastructure from scratch and create our own data centers. It is recommended not to remove the predefined data center. We will create a single data center with the NFS storage and this example will describe the setting up of oVirt. There will also be examples to connections of other storage types.

Creating a new data center

The first step in the oVirt virtual environment is the process of creating the data center. Log in to the administrator portal and go to the **Data Centers** section in the resource pane. Click on **New**. The dialog box named **New Data Center** should be filled with the settings for the new data center. The components of the dialog box are given as follows:

- **Name**: Here we need to specify a unique data center name.
- **Type**: Select the type of storage, choose **NFS**, or any other type, depending on the type of your available storage.
- **Compatibility Version**: It must be the latest version. This field is used to upgrade data center version in case of oVirt Engine upgrade. For example, from 3.2 to 3.3.
- **Quota Mode**: Here we can enable quotas that are used to limit resources of oVirt users. Quota is an extended feature and will be covered in *Chapter 5*, *Advanced Features*.

The following screenshot shows the **New Data Center** dialog box:



Click on the **OK** button and open the data center wizard named **Guide Me**. This is a very useful wizard for those who have never seen the settings of oVirt.

The next step is to create a cluster; click on the **Configure Cluster** button in the **Guide Me** wizard. Note that there is a way to create a cluster without a master. To do this, select the **Clusters** tab in the resource pane and click on the **New** button in the toolbar.

Clusters

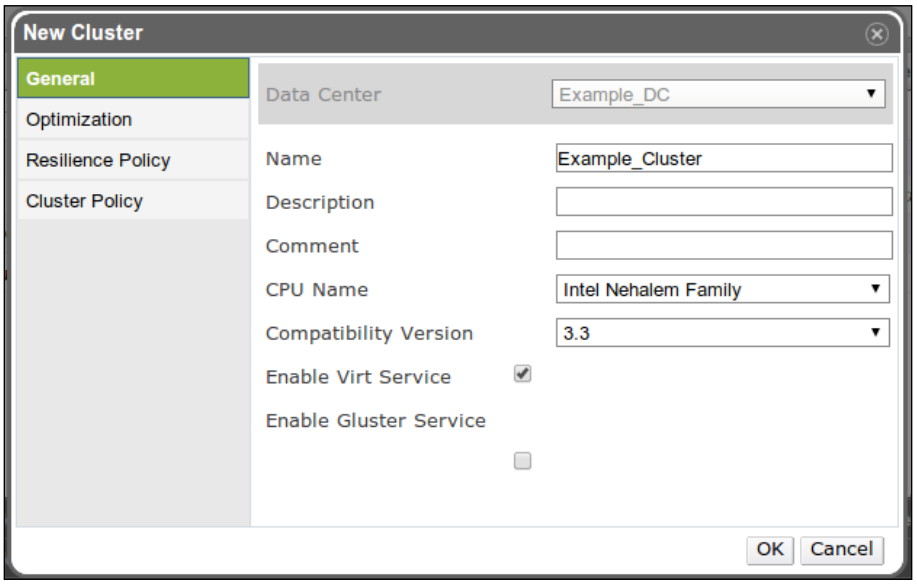
Clusters are groups of physical servers that use a shared storage and have one type of CPU. VMs in a cluster may move from one physical host to another. In this case, the VM does not stop. This process is called **Live Migration**. There is one limitation, the virtual machines cannot migrate between clusters. Any virtualization host is a member of the cluster. Any cluster in the oVirt environment necessarily belongs to data centers. Using this hierarchy along with the cluster policies, we will have the ability to dynamically allocate the available resources and to determine which of the hosts will run the VM. When the VM is started, it can be migrated between hosts within their cluster. This is done either at the request of the Administrator or automatically to comply with the cluster policy. VM migration is also very useful when we need to perform virtualization host maintenance. The migration process is transparent to the end user, and the user can continue working as usual at this time.

Creating a cluster

After clicking on **Configure Cluster**, a dialog box is displayed in which you must fill in the parameters for the future cluster. These parameters are grouped into several sections, which are given as follows:

- **General**: It describes the basic parameters and the appointment of a cluster. Particular attention should be paid to the CPU type in the **CPU Name** list. Choose the type of processor from the available list. Virtualization hosts that are members of the same cluster must have the same type of CPU; this is a very important condition when we have to perform VMs live migration within a cluster. If we use servers with a different CPU, the lowest family needs to be chosen, or it will be better if you create multiple clusters. When you try to join the host with an older CPU family to a cluster with a newer family configured, you can see the following message in the event pane: `Host moved to Non-Operational state as host does not meet the cluster's minimum CPU level`. CPU family can be determined through the `/proc/cpuinfo` interface.
- **Compatibility Version**: The version of the cluster must be the same as the version of oVirt Engine. This option used when oVirt Engine is upgraded.
- In fact, cluster may be used in two ways: as a virtualization host and as a Gluster storage. The previous points define the role of the cluster; the cluster can be used in virtualization purposes to run VMs for storage purposes, which will allow us to create GlusterFS volumes or a combination of both. Click on the **Enable Virt Service** checkbox, which enables the ability to run VMs in a cluster. We can also click on the **Enable Gluster Service** checkbox to enable Gluster storage service. Alternatively, click on both the checkboxes to enable both the services.

The following screenshot shows the **New Cluster** dialog box:



The components of the **New Cluster** dialog box are as follows:

- **Optimization**: It allows us to set the parameters for the optimization of memory and processors:
    - **Memory Optimization**: It specifies the use of technology **KSM** (**Kernel Samepage Merging**) which allows you to combine identical memory pages into one page. With the use of KSM memory utilization, efficiency increases. It has the following options:

      None: It allows you to disable the use of KSM.

      For Server Load: It allows us to use moderate settings KSM.

      For Desktop Load: It allows the use of aggressive parameters in KSM.

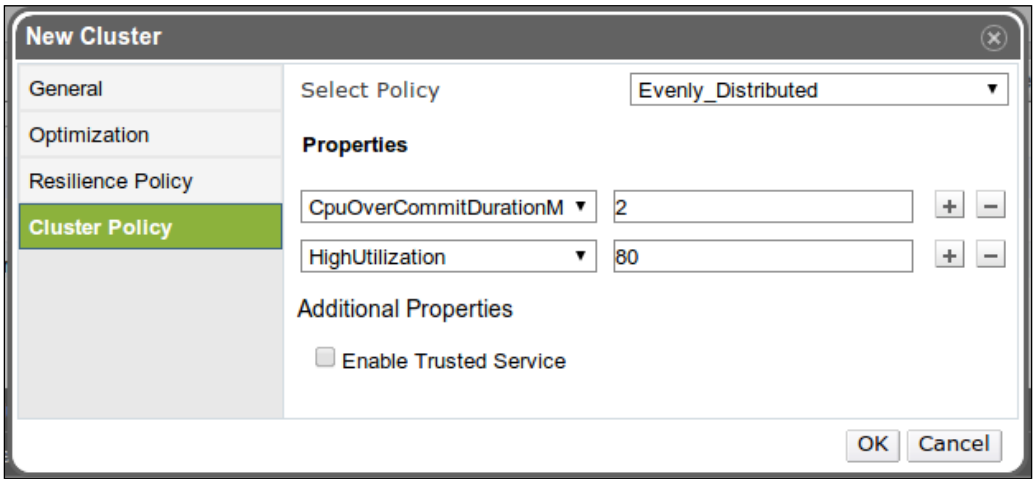
It should be noted that the use of KSM implies an additional load on the CPU, because to find the same pages requires additional computing power. And the more aggressive options, the more CPU processing power required. Note that heavy-loaded relational databases such as Oracle, PostgreSQL, MySQL/MariaDB, will cause a high load due the fact that their memory changes a lot. More info about KSM can be found on Fedora KSM page at `https://fedoraproject.org/wiki/Features/KSM`.

    - **CPU Threads**: It allows the hypervisor to use separate threads for each core as a separate virtual processor. This requires that the physical processor supports **AMD Clustered MultiThreading** or **Intel Hyper-Threading**. Enabling this option may be useful when it is planned that the cluster will not have a high load.
    - **Memory Balloon**: It allows shrinking the memory used by the hypervisors and increases VMs memory. Ballooning allows the VMs to dynamically change their memory usage by evicting unused memory.

- **Resilience Policy**: This policy is here to define the management policies that are associated with the configuration of VM's high availability. Depending on the need and the importance of running machines in a cluster, you can choose one of the following three policies:
    - **Migrate Virtual Machines**: This is used if it is necessary to use an automated migration for any VMs in cluster
    - **Migrate only Highly Available Virtual Machines**: This option is used in case there is a need to migrate only high availability VMs
    - **Do not Migrate Virtual Machines**: This option is used if we do not need to perform automatic migration of VMs

- **Cluster Policy**: It is a policy which determines the VM's allocation strategy within the cluster. In this case, it is determined by the load limits for virtualization hosts. When the limits are reached, the VM will be permanently migrated to other nodes in the cluster to return to a state policy of satisfying the cluster. By default, there are two policy clusters, **Even Distribution** and **Power Saving**. Even Distribution allows you to evenly distribute the load on the cluster nodes. Power Saving allows you to collect the load on multiple cluster nodes to permit further opportunities for the newly independent nodes. oVirt 3.3 allows you to create your own cluster policies; this feature will be covered in *Chapter 5*, *Advanced Features*.

This figure shows the **Cluster Policy** section:



After defining the cluster parameters, click on **OK** and return to the data center configuration menu named **Guide Me**. It should be noted that the data center may have several clusters and more clusters can be created in many ways.

The next step is to add virtualization hosts. To do this, in the wizard **Guide Me**, we need to click on **Configure Hosts**. Or go to the **Hosts** tab in the resource pane.

Hosts

The main components of the infrastructure are the virtualization hosts called *Hosts*.

Start and run the VMs on the hosts. All hosts are members of the cluster. Joining Hosts into clusters accomplishes many goals, for example, creating a logical structure in the virtualization environment or creating a separated group of servers for special purposes. Also, hosts can be joined into clusters by CPU type.

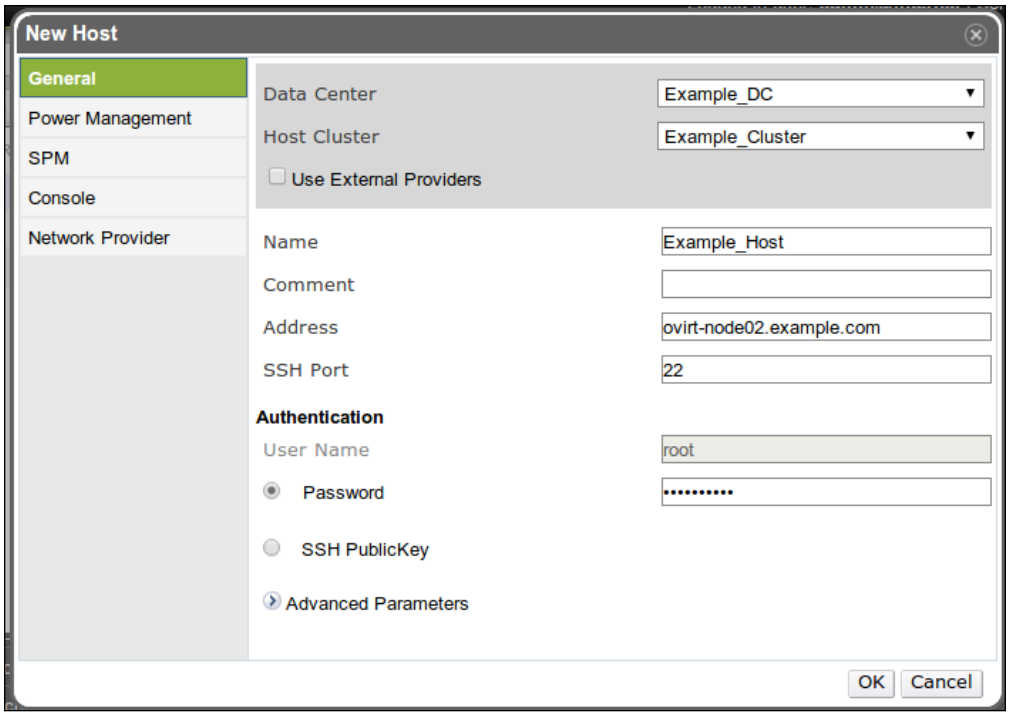
Configuring hosts

The configuration of virtualization hosts in oVirt is simple and requires only a physical server that meets the hardware requirements with CentOS/RHEL preinstalled and accessible via SSH. This host must have a preconfigured VDSM repository. This issue is described in *Chapter 2*, *Installing oVirt*, in the *oVirt Engine setup* section.

After clicking on **Configure Hosts** in the **Guide Me** wizard, a dialog box for adding virtualization host is displayed, which has the following three sections:

- **General**: Here you should enter the basic settings for the new host. These settings are given as follows:
    - **Name**: It allows us to specify the unique name for the new host.
    - **Address**: It allows us to specify the **FQDN** (**fully qualified domain name**) or IP address of the host.
    - **SSH Port**: It specifies a port number to connect to the host. Be sure that the sshd service is running on the target host.
    - **Root Password**: It allows us to define the password to connect to the root account on the server.

    The following screenshot shows the **New Host** dialog box with the **General** section:

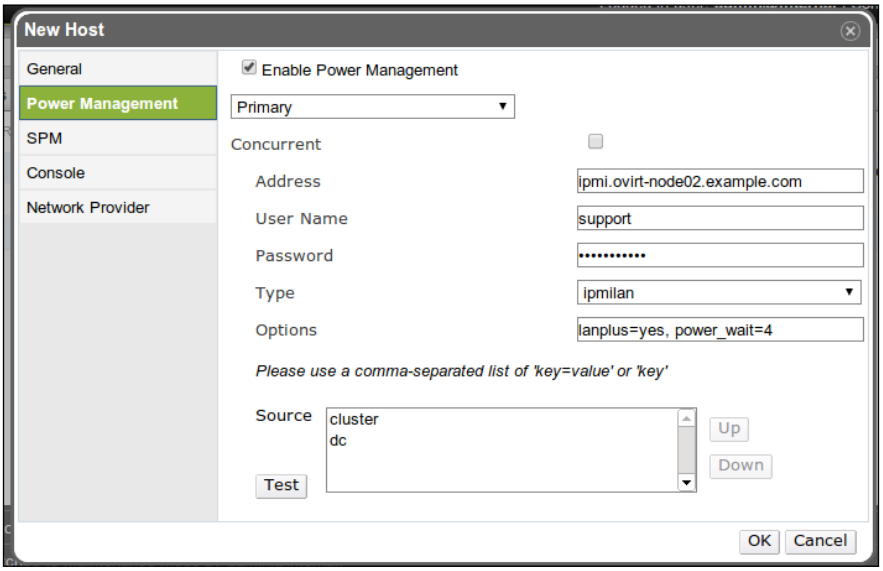


Here we define the parameters for the virtualization host. Using these parameters, oVirt Engine will connect to the host and run a fully automated installation process VDSM agent. There is also a clause to automatically configure the remote host firewall.

- **Power Management**: Power management is a mechanism that allows the cluster to shutdown the host on which the failure occurs. oVirt Engine works directly with the hardware through special fence agents. When a failure is detected, oVirt sends a signal to the control module that powers off the server. Power management can be configured if the integrated management module is present in server. Click on the section named **Power Management** and enable the **Enable Power Management** option and specify the following required parameters:
    - **Address**: Here we need to specify the remote management module IP address or FQDN.
    - **User Name** and **Password**: Here you can specify the user and password under which oVirt will connect.
    - **Type**: It determines the type of device that will control the power. oVirt supports different types of remote management device. Additional info about the specific modules can be found in the vendor hardware documentation.
    - **Options**: It consists of optional parameters that are passed to fence agent. Detailed information about the available options can be found in the man pages of the respective fence agents.

  To check the power management agent, click on the **Test** button. Note that at least two nodes need to be in a cluster, otherwise power management won't work.

  The following figure shows the **Power Management** window:



For high availability, it is required to set up power management. Without power management, high availability won't work. But if you don't have hardware allowing you to perform power management, this step can be skipped and keep the **Enable Power Management** option disabled.

- **SPM**: **SPM** (**Storage Pool Manager**) priority feature allows the admin to define priorities between hosts regarding the **SPM** election process. SPM is a role assigned to one host in a data center which has the power to allocate, create, delete, and manipulate virtual disk images, snapshots, and templates in storage. This role can be migrated to any host in data center. SPM role is assigned to a host with high priority. A host can be given **Low**, **Normal**, or **High** priority. There should be only one SPM in a data center.
- **Console**: This section allows us to override IP address on which VMs consoles will be run.
- **Network Provider**: It allows us to specify OpenStack Neutron as an external network provider and use Neutron capabilities such as network discovery, provisioning, security groups, and others. More information about Neutron integration can be found at `http://www.youtube.com/watch?v=S16AfFylcHk`.

By clicking on **OK**, we'll be back in the configuration dialog of the data center and a background process setting up the virtualization host is run. During the installation process, oVirt Engine will connect to the specified host and run the software installation process. After this, it will restart a new host. The new host will automatically be added to the specified cluster. The next step is to configure the storage.

Configuring storage

Storage is independent of an oVirt component unless you use local storage or GlusterFS on virtualization hosts. Storage should be preconfigured, so in order to proceed further, you must have ready-to-use storage.

For configuring the storage in the **Guide Me** dialog box, you must click on **Configure Storage**. Or you can do it in the **Storage** tab is in the resource pane. In both cases, it will open the same dialog box allowing you to add and connect storage to the data centers. The attached storage must match the type that was specified when creating a data center. Note that multiple storage of the same type can be connected to the data center. All types of storage listed in the following section allow the administrator to be flexible in the oVirt environment. The storage can be connected, depending on the availability of resources.

In oVirt, there are two additional storage required to perform regular functions. These are the ISO and the export storage domains which will be covered after data storage types.

Configuring the NFS storage

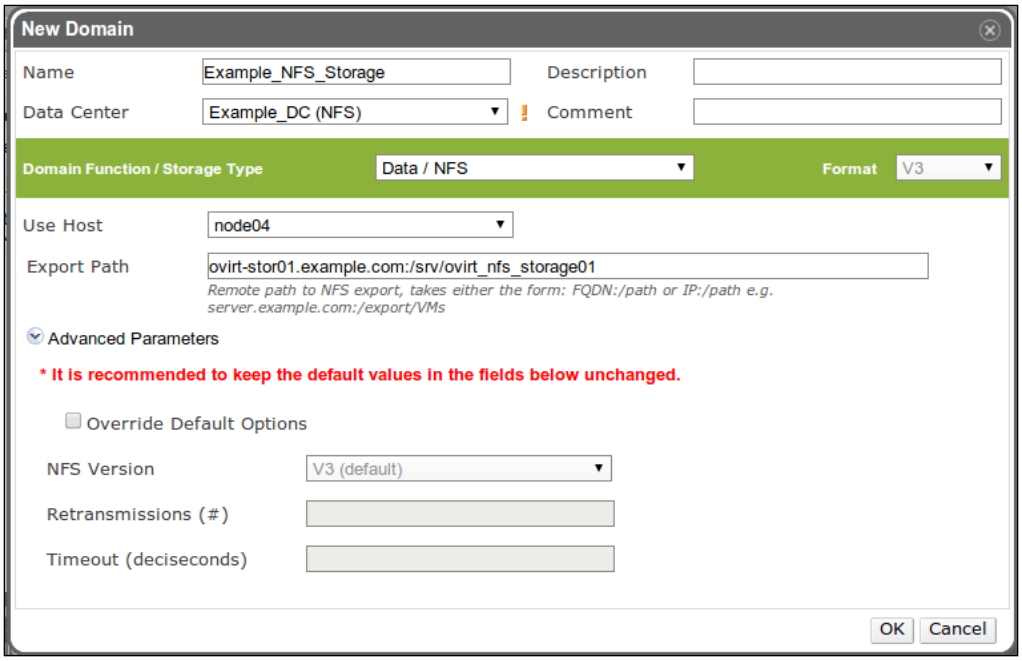
NFS storage is a fairly common type of storage that is quite easy to set up and run even without special equipment. You can take the server with large disks and create NFS directory. This process is covered in *Appendix A*, *NFS Storage Setup on CentOS*. But despite the apparent simplicity of NFS, setting s should be done with attention to details, which covered in *Appendix A*, *NFS storage setup on CentOS*.

Make sure that the NFS directory is suitable for use; go to the procedure of connecting storage to the data center. The following options are displayed after you click on the **Configure Storage** dialog box in which we specify the basic storage configuration:

- **Name and Data Center**: It is used to specify a name and target of the data center for storage
- **Domain Function/Storage Type**: It is used to choose a data function and NFS type
- **Use Host**: It is used to enter the host that will make the initial connection to the storage and a host who will be in the role of SPM
- **Export Path**: It is used to enter the storage server name and path of the exported directory
- **Advanced Parameters**: It provides additional connection options, such as NFS version, number of retransmissions and timeout, that are recommended to be changed only in exceptional cases

Fill in the required storage settings and click on the **OK** button; this will start the process of connecting storage.

The following image shows the **New Storage** dialog box with the connecting NFS storage:



Configuring the iSCSI storage

This section will explain how to connect the **iSCSI** storage to the data center with the type of storage as iSCSI. You can skip this section if you do not use iSCSI storage.

iSCSI is a technology for building **SAN** (**Storage Area Network**). A key feature of this technology is the transmission of SCSI commands over the IP networks. Thus, there is a transfer of block data via IP. By using the IP networks, data transfer can take place over long distances and through network equipment such as routers and switches. These features make the iSCSI technology good for construction of low-cost SAN. oVirt supports iSCSI and iSCSI storages that can be connected to oVirt data centers.

Then begin the process of connecting the storage to the data center. After you click on the **Configure Storage** dialog box in which you specify the basic storage configuration, the following options are displayed:

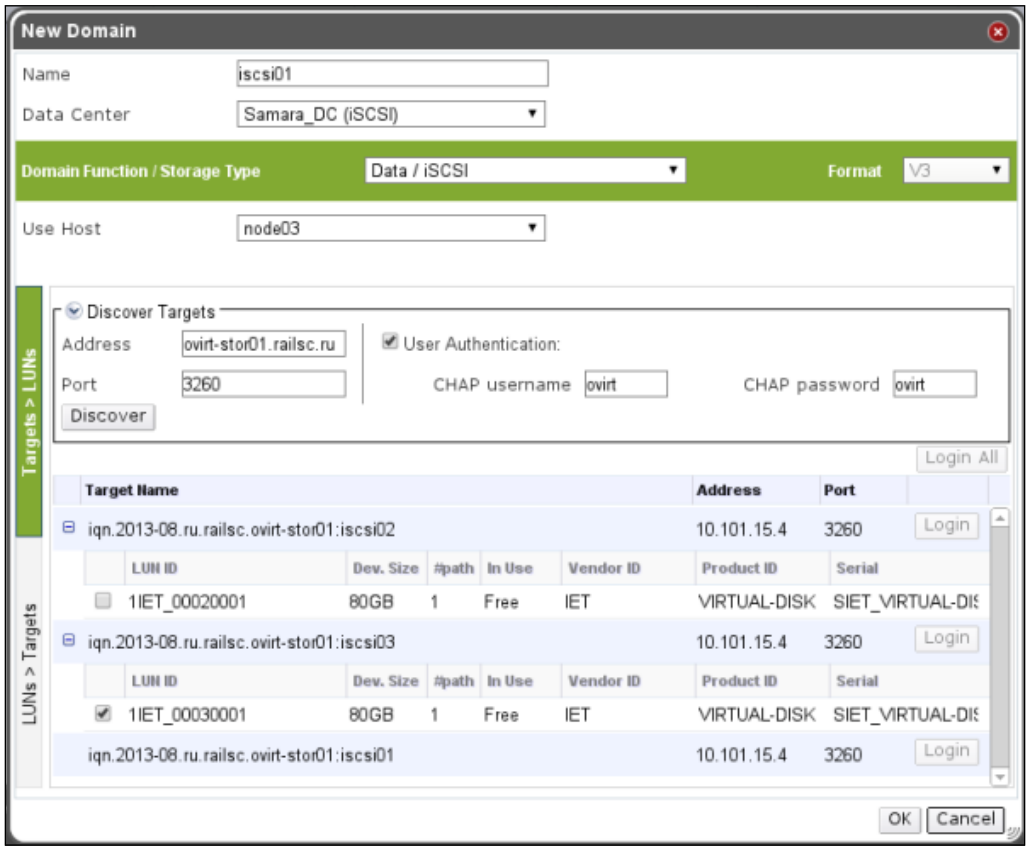
- **Name and Data Center**: It is used to specify the name and target of the data center.
- **Domain Function/Storage Type**: It is used to specify the domain function and storage type. In this case, the data function and iSCSI type.
- **Use Host**: It is used to specify the host to which the storage (SPM) will be attached.

The following options are present in the search box for iSCSI targets:

- **Address and Port**: It is used to specify the address and port of the storage server that contains the iSCSI target
- **User Authentication**: Enable this checkbox if authentication is to be used on the iSCSI target
- **CHAP username and password**: It is used to specify the username and password for authentication

Click on the **Discover** button and oVirt Engine connects to the specified server for the searching of iSCSI targets. In the resulting list, click on the designated targets, we click on the **Login** button to authenticate. Upon successful completion of the authentication, the display target LUN will be displayed; check it and click on **OK** to start connection to the data center. New storage will automatically connect to the data center. If it does not, select the location from the list and click on the **Attach** button in the detail pane where we choose a target data center.

The following screenshot shows the **New Storage** dialog box with iSCSI storage type:



Configuring the Fibre Channel storage

If you have selected **Fibre Channel** when creating the data center, we should create a Fibre Channel storage domain. oVirt supports Fibre Channel storage based on multiple preconfigured **Logical Unit Numbers** (**LUN**). Skip this section if you do not use Fibre Channel equipment.

Begin the process of connecting the storage to the data center. Open the **Guide Me** wizard and click on the **Configure Storage** dialog box where you specify the basic storage configuration:

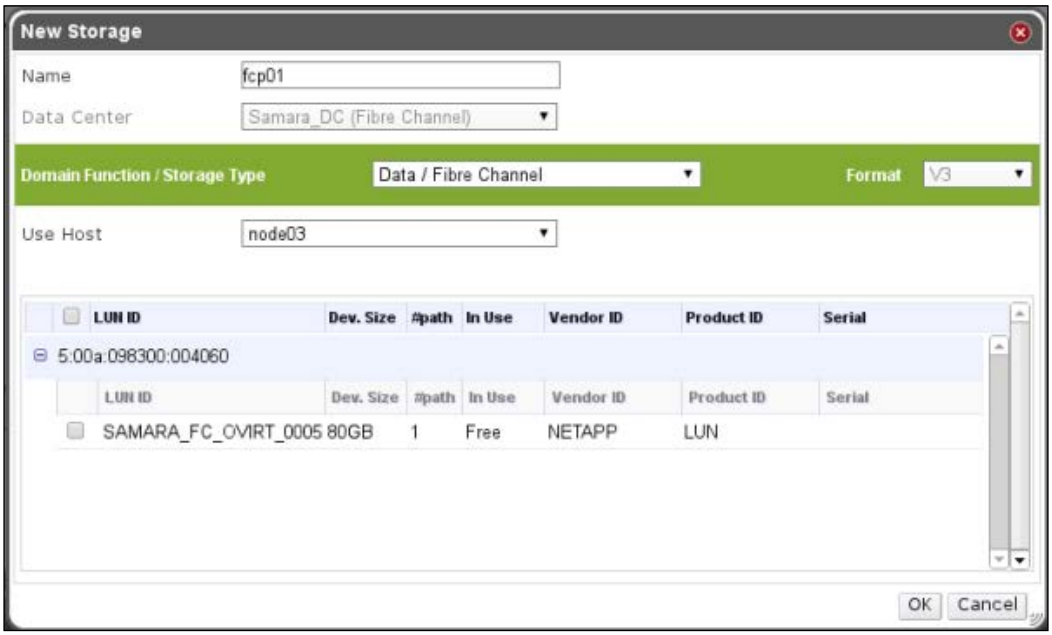
- **Name and Data Center**: It is used to specify the name and data center
- **Domain Function/Storage Type:** Here we need to specify the data function and Fibre Channel type

- **Use Host**: It specifies the address of the virtualization host that will act as the SPM

In the area below, the list of LUNs are displayed, enable the **Add LUN** checkbox on the selected LUN to use it as Fibre Channel data storage.

Click on the **OK** button and this will start the process of connecting storage to the data centers. In the **Storage** tab and in the list of storages, we can see created Fibre Channel storage. In the process of connecting, its status will change and at the end new storage will be activated and connected to the data center. The connection process can also be seen in the event pane.

The following screenshot shows the **New Storage** dialog box with Fibre Channel storage type:



Configuring local on host storage

It is possible to create a local on host type of storage for the cases when the virtualization host, cluster, and data center is located on the single physical server. When using this type of storage storage type is a significant limitation. For VMs that are created on a cluster of the local type, we cannot perform migrations, fencing, or scheduling. Skip this section if you do not plan to use local on host data centers.

Preparing the local storage

Before connecting the storage data center, we need to create a storage directory on a virtualization host:

1. Create a separate partition that will have enough space to contain all VM images. Make a directory that will serve as the local store.
2. Directory must have permissions to read and write for the user's VSDM and group for KVM.

As an example, we may use the directory `/srv/vmdata`. This is done as shown in the following commands:

```
# chown vdsm:kvm /srv/vmdata
# chmod 0755 /srv/vmdata
```

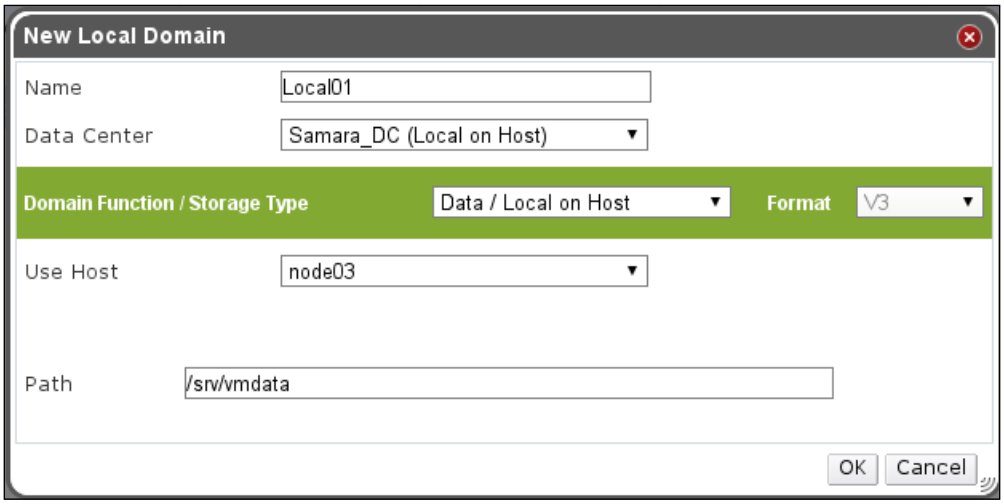
Connecting the local storage into data center

Go to the **Storage tab** in the resource pane. Click on the **New Domain** button. A dialog box opens, set up the storage options:

- **Name and Data Center**: It is used to specify a name and data center for the new storage
- **Domain Function/Storage Type**: It specifies the data function and the local on host type
- **Use Host**: It allows to only select a single host that is available in a data center
- **Path**: It specifies our catalog that will be used as storage `/srv/vmdata`

Click on **OK** and then start the process of connecting to a storage data center. When it is ready, our local storage will automatically attach to the data center.

The following figure shows the **New Storage** dialog box with the **Local on Host** storage type:



Configuring the GlusterFS storage

GlusterFS is a distributed, parallel, and linearly scalable filesystem. GlusterFS can combine the data storage that are located on different servers into a parallel network filesystem. GlusterFS's potential is very large, so developers directed their efforts towards the implementation and support of GlusterFS in oVirt (GlusterFS documentation is available at `http://www.gluster.org/community/documentation/index.php/Main_Page`). oVirt 3.3 has a complete data center with the GlusterFS type of storage.

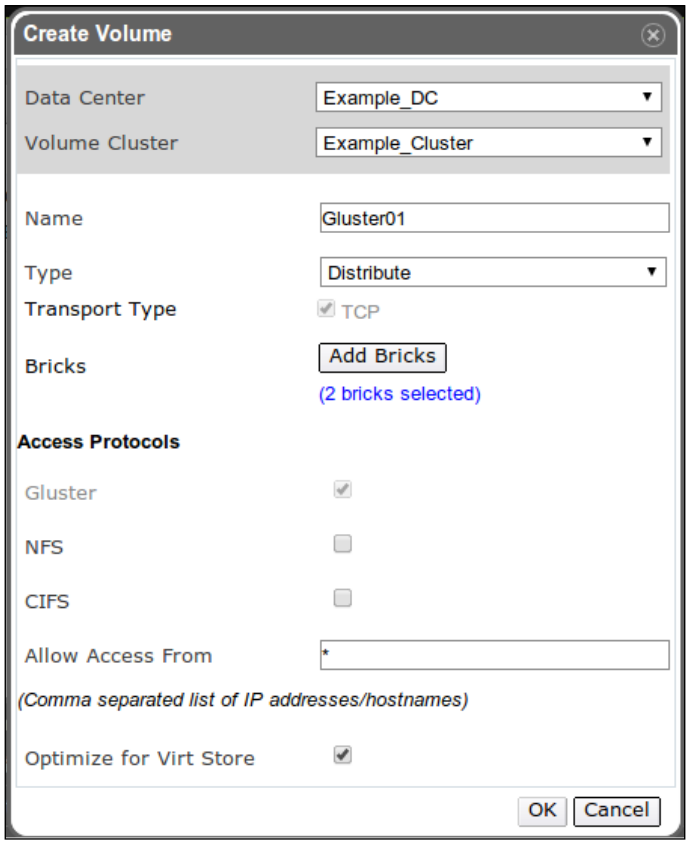
Configuring the GlusterFS volume

Before attempting to connect GlusterFS storage into the data center, we need to create the volume. The procedure of creating GlusterFS volume is common in all versions.

1. Select the **Volumes** tab in the resource pane and click on **Create Volume**.
2. In the open window, fill the volume settings:
    - **Data Center**: It is used to specify the data center that will be attached to the GlusterFS storage.
    - **Volume Cluster**: It is used to specify the name of the cluster that will be created.

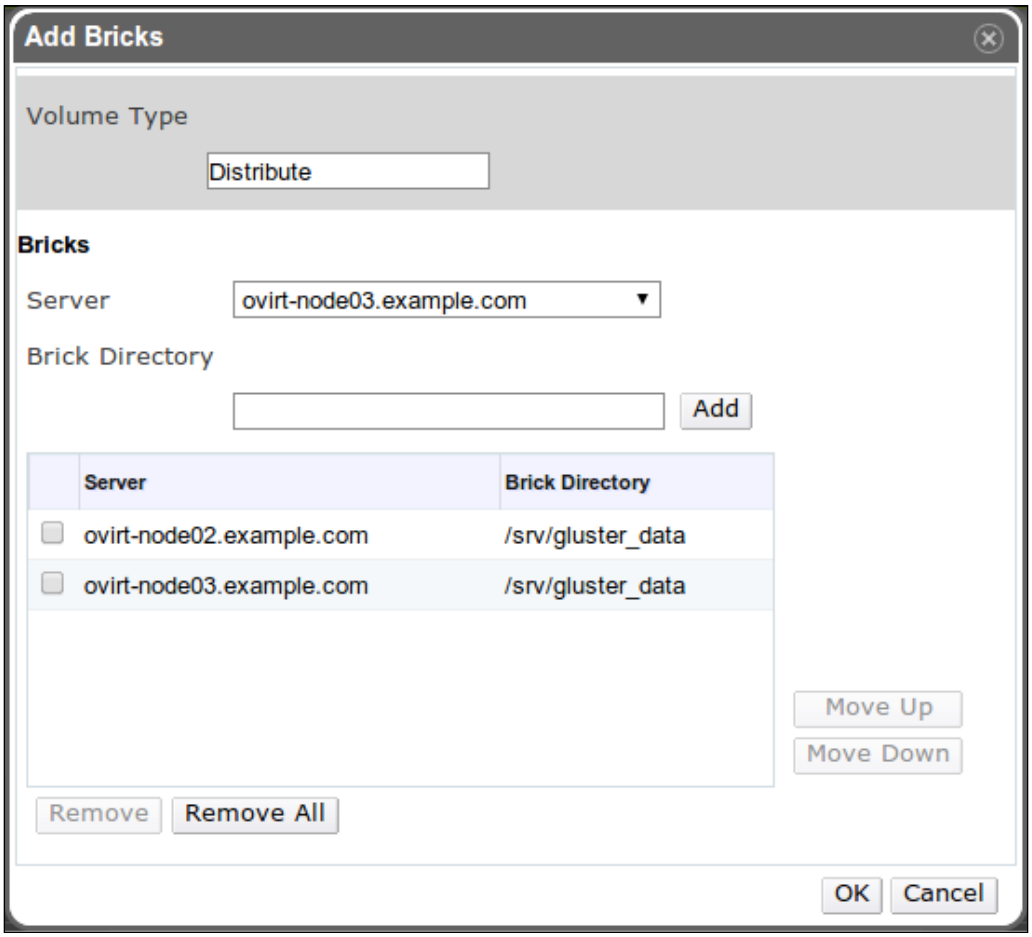
- **Name**: It is used to specify a name for the new volume.
- **Type**: It is used to specify the type of GlusterFS volume available to choose from, there are seven types of volume that implement various strategic placements of data on the filesystem. Base types are Distribute, Replicate, and Stripe and other combination of these types: Distributed Replicate, Distributed Stripe, Striped Replicate, and Distributed Striped Replicate (additional info can be found at the link: `http://gluster.org/community/documentation/index.php/GlusterFS_Concepts`).
- **Bricks**: With this button, a list of bricks will be collected from the filesystem. Brick is a separate piece with which volume will be built. These bricks are distributed across the hosts. As bricks use a separate directory, it should be placed on a separate partition.
- **Access Protocols**: It defines basic access protocols that can be used to gain access to the following:
    i. **Gluster**: It is a native protocol access to volumes GlusterFS, enabled by default.
    ii. **NFS**: It is an access protocol based on NFS.
    iii. **CIFS**: It is an access protocol based on CIFS.
- **Allow Access From**: It allows us to enter a comma-separated IP address, hostnames, or * for all hosts that are allowed to access GlusterFS volume.
- **Optimize for oVirt Store**: Enabling this checkbox will enable extended options for created volume.

The following screenshot shows the dialog box of **Create Volume**:



3. Fill in the parameters, click on the **Bricks** button, and go to the new window to add new bricks with the following properties:
    - **Volume Type**: This is used to change the previously marked type of the GlusterFS volume
    - **Server**: It is used to specify a separate server that will export GlusterFS brick
    - **Brick Directory**: It is used to specify the directory to use
4. Specify the server and directory and click on **Add**. Depending on the type of volume, specify multiple bricks. After completing the list with bricks, click on the **OK** button to add volume and return to the menu.
5. Click on the **OK** button to create GlusterFS volumes with the specified parameters.

The following figure shows the **Add Bricks** dialog box:



Now that we have GlusterFS volume, we select it from the list and click on **Start**.

Configuring the GlusterFS storage

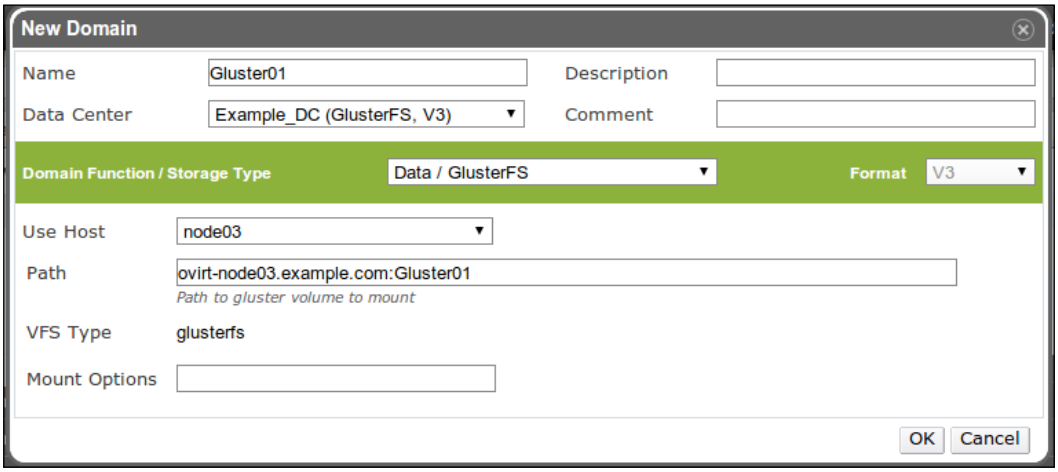
oVirt 3.3 has support for creating data centers with the GlusterFS storage type:

1. The GlusterFS storage type requires a preconfigured data center.
2. A pre-created cluster should be present inside the data center. The enabled Gluster service is required.
3. Go to the **Storage** section in resource pane and click on **New Domain**.

4. In the dialog box that opens, fill in the details of our storage. The details are given as follows:
    - **Name and Data Center**: It is used to specify the name and data center
    - **Domain Function/Storage Type**: It is used to specify the data function and GlusterFS type
    - **Use Host**: It is used to specify the host that will connect to the SPM
    - **Path**: It is used to specify the path to the location in the format `hostname:volume_name`
    - **VFS Type**: Leave it as **glusterfs** and leave **Mount Option** blank
5. Click on the **OK** button; this will start the process of creating the repository.

The created storage automatically connects to the specified data centers. If not, select the repository created in the list, and in the subtab named **Data Center** in the detail pane, click on the **Attach** button and choose our data center. After you click on **OK**, the process of connecting storage to the data center starts.

The following figure shows the **New Storage** dialog box with the GlusterFS storage type.

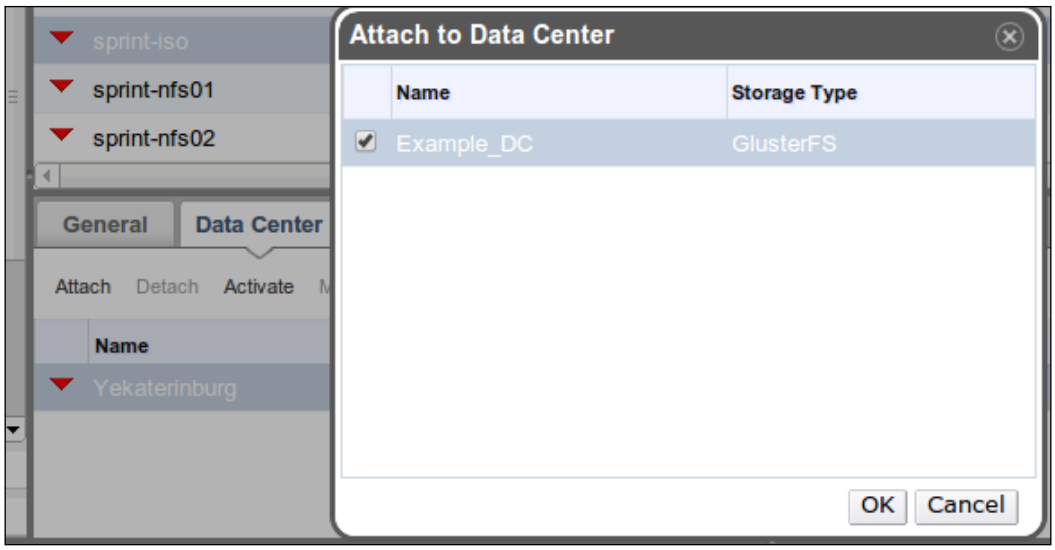


Configuring the ISO domain

Most general installation methods for installing VMs are the installing it with ISO images. For storing ISO images, use the ISO domain, which is the NFS directory. In most cases, ISOs connecting the ISO domain is a required step in configuring the data center. These steps are given as follows:

1. To connect to the ISO domain, we need to open the **Storage** tab in the resources pane. Here, the ISO domain that was created during the configuration engine setup is already present.
2. Select the predefined ISO domain from the list, and the details pane opens; click on the **Attach** button.
3. In the dialog box that opens, click on the checkbox for our data center and click on **OK**.
4. Now we need to go to the **Data Centers** section in the resource pane. Select the data center to which ISO domain was attached. In the detail pane, select the ISO domain and click on on the **Activate** button.

The following image shows the **Attach to Data Center** dialog box:



After a while, the ISO domain icon turns green, which means that the ISO domain is connected and ready for use.

We can now upload the installation image to the ISO domain storage, this step is performed in the Linux console. These steps are given as follows:

1. Copy the ISO image to the directory of the system, which is running oVirt Engine.
2. Log in with the root account to the system running oVirt Engine.
3. To download the ISO image, use a command line utility `ovirt-iso-uploader`. The process of uploading the image takes some time depending on the size of the ISO image, and the network bandwidth between oVirt Engine and the ISO domain.

After the ISO image is uploaded, we can install a VM in the oVirt environment.

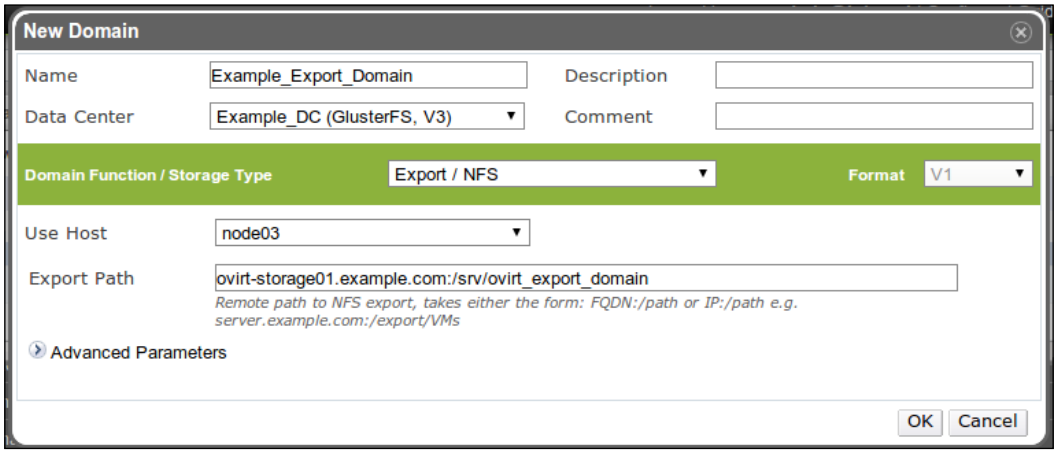
Configuring the export domain

The export domain is needed to store the exported VMs during oVirt software upgrades, moving VMs across data centers, import VMs with *virt-v2v/virt-p2v* or *Acronis* utilities. Export domain is also an NFS storage, but it is not created by default. So, we can create the export domain manually. Export domain is not required for the operation of the data center so it is always possible to create it later when it is necessary. Export domain creation procedure is given in the following steps:

1. Click on the **New Domain** button in the **Storage** tab.
2. In the dialog box that opens, fill in the following parameters of the new domain:
    - **Name and Data Center**: It is used to specify the storage name and data center
    - **Domain Function/Storage Type**: It is used to provide a domain function and the type of storage selected as an export function, and the type is NFS
    - **Use Host**: It is used to specify the host that will be connected to the storage or a host that will be in the role of SPM
    - **Export Path**: It is used to provide a storage server name and path for the exported directory
    - **Advanced Parameters**: It is used to specify additional connection options that are recommended to be changed only in exceptional cases

3. Fill in the required storage options and click on on the **OK** button; this will start the process of connecting the storage.

   The following figure shows the **New Domain** dialog box

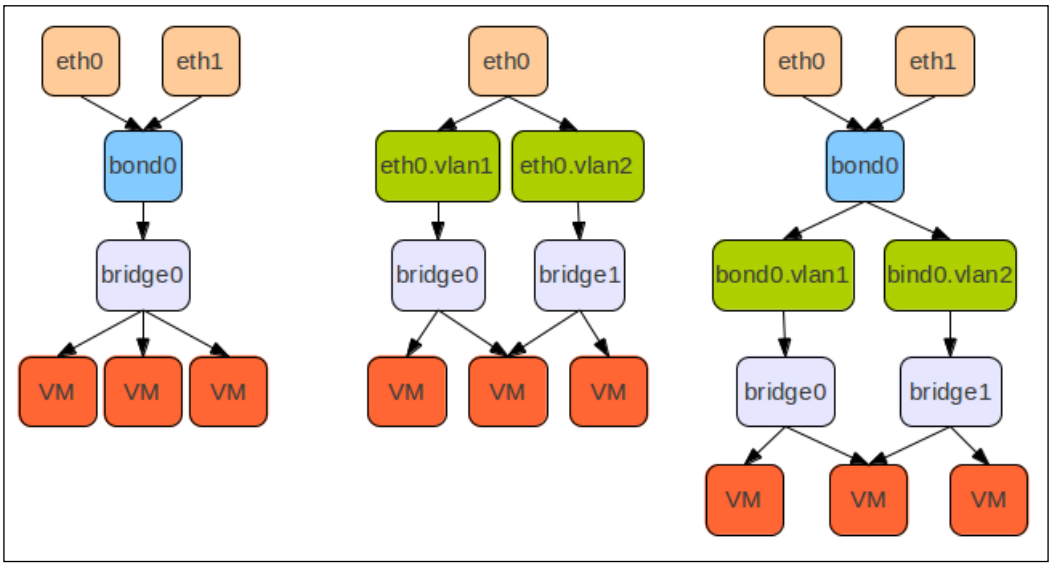


oVirt networking

To start using oVirt, each physical machine should have a network card connected to a network and an IP address manually assigned or received by DHCP. This approach is already providing a functional environment. By default, oVirt Engine creates a single *logical network* and uses this network for all types of traffic. oVirt's default network is created and labeled as `Management logical network`. The oVirt logical network is designed for the management of traffic between the oVirt Engine and the virtualization hosts.

Practice shows that for efficient operation of the network environment, the different types of traffic should be separated. oVirt can assign different types of network traffic into dedicated logical networks, and each logical network has to be associated with the network interface of the virtualization host. Such a network interface can act as a physical device or logical, such as bond or virtual NIC.

The following diagram shows network usage scenarios:



Logical networks

By default, the logical network named `ovirtmgmt` is attached to the precreated data center named `Default`. At the same time, we can create new logical networks for some purposes, for example, data, storage, or display. In most cases, logical networks are used for network separation or for a virtualization physical topology. In addition, other logical networks can be used to separate VMs traffic from the management networks. Another purpose may be traffic isolation between VMs groups in the same cluster. In oVirt Engine, network definition, type, and function are integrated in a logical entity called a logical network.

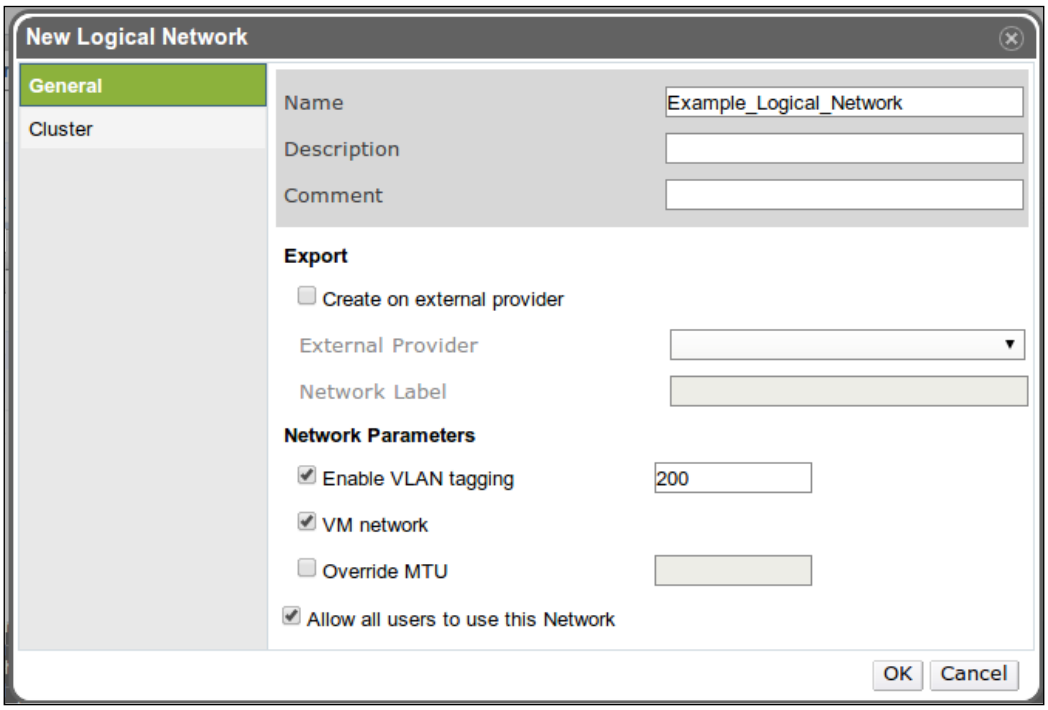
Creating a new logical network

Creating a logical network is process independent and can be executed at any time, regardless of the current situation in the oVirt environment. The steps for creating a logical network are as follows:

1. Go to the **Data Centers** section in the resource pane and select data center in which a logical network will be created.
2. In the **details pane**, select the **Logical Networks** section and click on **New**.

3. In the dialog box that opens, fill in the network parameters in the **General** section:
    - **Name**: It is used to specify the name for the new logical network
    - **Enable VLAN tagging**: It is used to specify the VLAN tag
    - **VM network**: Tick this checkbox if network should be used by VMs
    - **Override MTU**: It is used to change the maximum transmission unit if required
4. Select the **Cluster** section, mark the target data center, and click on **Attach**.
5. Click on **OK** and it will start creating the new logical network and will automatically attach it to the data center.

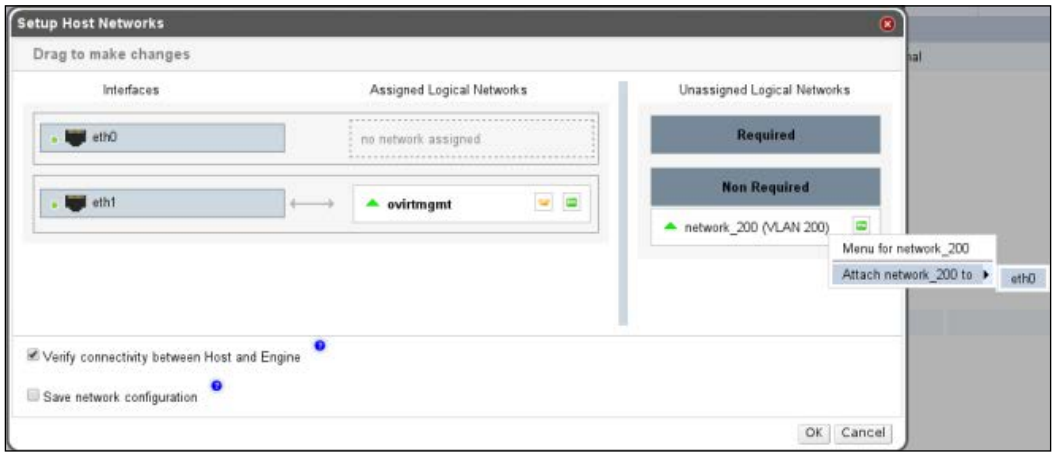
    The following screenshot shows the **New Logical Network** dialog box:



The final step is assigning the logical network to the virtualization hosts; this is given in the following steps:

1. Go to the **Hosts** tab in the resource pane and choose the target host.
2. In the details pane, select the **Logical Networks** section and click on **Setup Host Network**.
3. The current network configuration of the host, which was selected previously, is determined on the left side of the window. On the right-hand side of the dialogue box, there is a list of available logical networks. Click on the desired network and select the **Attach network to** item from the drop-down list. Then select the interface to which it must be connected to a logical network.

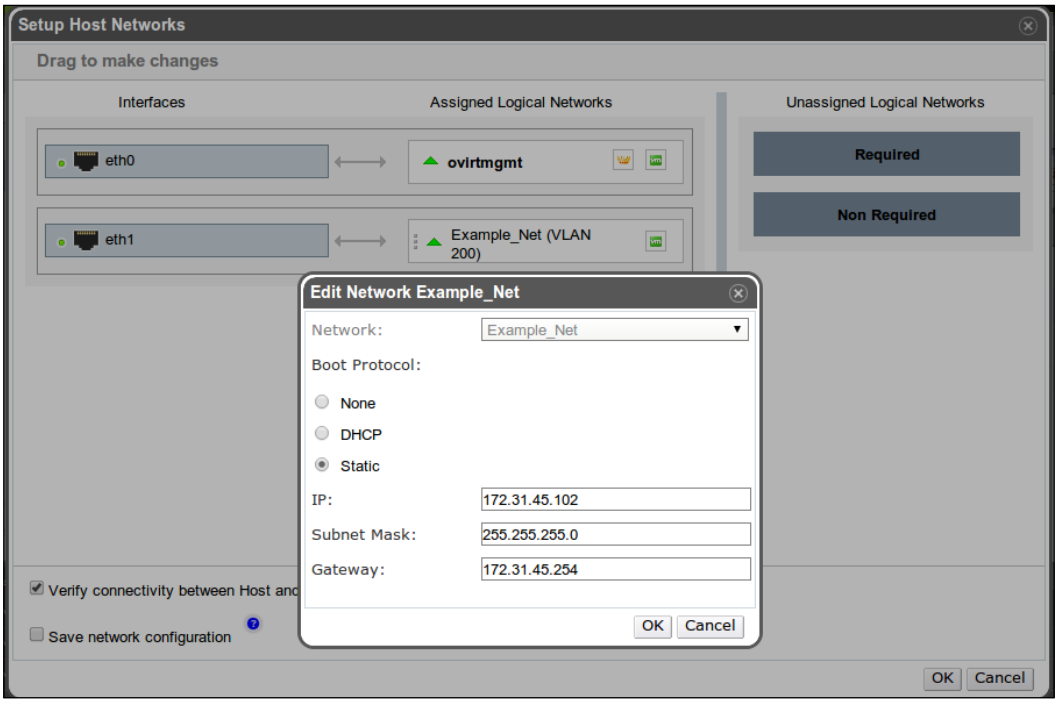
   The following screenshot shows the **Setup Host Networks** dialog box:



After connecting to the logical network interface, you must configure a network connection; the steps for this are given as follows:

1. Select the logical network from the left-hand side of the screen and click on the pencil icon.
2. In the **Edit network** dialog box, specify the network connection type. Here, **DHCP** is available, which allows you to use DHCP for automatic TCP/IP configuration and **Static**. When static configuration is selected, we can set the IP address manually with the required **IP**, **Subnet mask**, and **Gateway**.

3. The following figure shows the **Edit Network** dialog box and the attached logical network from the previous **Setup Host Networks** dialog box.



4. After completing the settings, click on the **OK** button and return to the **Host Networks** setup menu**.** The following two checkboxes are present:
    - **Verify connectivity between Host and Engine**: This ensures we won't lose connectivity to the engine. If connectivity from the Host to the engine is lost after changing the network configuration, the changes are rolled back.
    - **Save network configuration**: Here, all changes done to the networking configuration are saved temporarily until they are explicitly saved. Enable the checkbox to make the changes persistent.
5. Click on **OK** and repeat this procedure for all hosts that you want to connect the logical network to.

In this step, the logical network is finished. We created a logical network and connected virtualization hosts to it.

Summary

In this chapter, we have done the step-by-step setup and configuration of the oVirt environment. We started with setting up data centers and clusters. Next, we placed our virtualization hosts in clusters. Afterwards, we attached storage. oVirt supports multiple storage types, it allows a very flexible design environment, oVirt and depending on your needs to choose the right type of storage. Finally, we met with the mechanism for creating logical networks, which is an important part of the virtualization infrastructure. In *Chapter 4*, *Managing Virtual Machines*, we proceed with the creation of VMs. This is the final step; after this, the oVirt environment is ready for use.

4
Managing Virtual Machines

Now that we know how to configure the oVirt environment, we can move on to the configuration and startup of virtual machines. A virtual machine has its own lifecycle of creation, startup, maintenance, and maybe removal. In this chapter, we will learn how to make and configure virtual machines. We will also consider the process of creating a template and then creating a virtual machine based on the template.

Creating virtual machines

oVirt allows creation of virtual machines from scratch or from existing templates. In fact, the VM creation process is a declaration of a few basic parameters. To start using the templates, they also need to create VMs. After creation, VMs can be booted with ISO images, virtual hard drives or through the PXE network boot protocol.

This section explains how to create and boot up the virtual machine from the ISO image.

In *Chapter 3*, *Configuring oVirt*, we used the `engine-iso-uploader` utility. With this tool, we are uploading an ISO image to a specially created ISO storage domain. Before creating and installing a virtual machine operating system from an ISO image, it is necessary that the ISO storage has at least one ISO image. With this image, we will boot up our virtual machine.

This means that adding a virtual machine is a three-step process:

1. Creating an empty virtual machine.
2. Connecting the Virtual Disks and network interfaces to the VM, if necessary.
3. Installing operating systems into the VM.

Creating Linux virtual machines

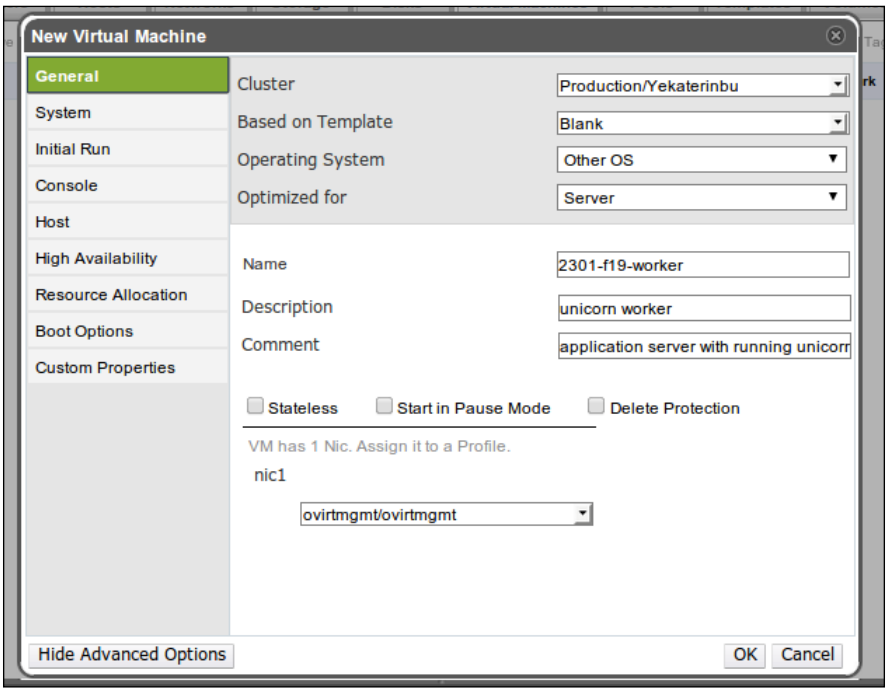
In most cases, all machines in oVirt are created in **Administrator Portal**. To create a new VM, log in into the **Administrator Portal** and open the **Virtual Machines** section in the resource pane.

To start the configure process, click on the **New VM** button. It will display the **New Virtual Machine** dialog box in which we need to define the parameters of the virtual machine. Visually, the configuration dialog consists of the given sections.

The **New Virtual Machine** dialog box is the main section that defines the basic parameters of the virtual machine:

- **Cluster**: It is used to specify the cluster and datacenter in which VM will run.
- **Based on Template**: It is used to specify a template which will be used when the virtual machine is created.
- **Operating System**: It is used to choose an operating system from the list. In the absence of the desired operating system, you can select the universal type **Other** or **Other Linux**.
- **Optimized for**: It is used to set the type of optimization, **Server** or **Desktop**.

The following screenshot shows the **New Virtual Machine** dialog box:



- **General**: It defines the general settings a virtual machine should have; these settings are as follows:
    - **Name**: It specifies a unique name for the virtual machine.
    - **Stateless**: If enabled, it determines that the virtual machine will be stateless. This means that when the VM stops, all changes inside the VM are lost.
    - **Start in Pause Mode**: In this mode, the VM will be immediately paused after startup and will be resumed only after an administrator request. This mode is very useful if you need to connect the SPICE/VNC console before the VM OS starts.
    - **Delete Protection**: This item can protect the virtual machine from accidental deletion. Such a machine cannot be removed unless this option is turned off in the VM configuration.
    - **Network Interface**: It is used to specify the network interface and attach it to existing logical networks.

  These parameters are sufficient to add a new virtual machine. Nevertheless, at the bottom of the dialog box, the **Show Advanced Options** button is present, which opens up additional sections related to configuring. This section covers the options related to resource allocation, high availability, VM placement, and others.

- The **System** section defines RAM and CPU resource options; these options are as follows:
    - **Memory Size**: It specifies the size of the memory. The size must not exceed the physical memory and the total amount set aside for the virtualization host.
    - **Total Virtual CPUs**: It specifies the number of virtual processors.
    - **Advanced Parameters**: There are additional parameters for determining SMP configuration inside the machine. All virtual processors can be represented by a number of *physical* processors with cores in each processor. For example, we can allocate 16 virtual processors as four physical processors (virtual sockets) with 4 cores each. Thus, KVM has SMP support within virtual environments.
    - **Virtual Sockets**: It is used to specify the number of sockets.
    - **Cores per Virtual CPU**: It is used to specify the number of cores in each socket.

- **Initial Run**: In this section, you can select the time zone for the virtual machine. Admin can set a VM with the desired clock offset, make a template from it, and each VM created from that template will have its clock set already. Secondly, if you are using MS Windows as a virtual machines operating system, you can specify the domain name for the *sysprep* tool. Additional information about sysprep can be found on Microsoft TechNet at `http://technet.microsoft.com/en-us/library/bb878150.aspx`
- **Console**: It is used to define the parameters associated with a graphical console that will be provided for remote access to the virtual machine's display:
    - **Protocol**: oVirt provides two protocols, namely, **VNC** and **SPICE**. VNC is an easy and convenient option if you do not want to support USBs and Smartcards. If you must use a USB or Smartcard, use the SPICE protocol.
    - **USB Support**: It enables support for USBs. There is a significant limitation—migration is not currently supported using SPICE's native USB redirection.
    - **Monitors**: It is used to enable support for multi-monitor virtual machines.
    - **Smartcard and Soundcard**: It enables support for the use of smartcards and soundcard, only when using the SPICE protocol.
    - **Virtio Console Device Enabled**: This checkbox specifies the use of a VirtIO serial console device for communication between the virtualization host and the new VM. Detailed information about the VirtIO serial console device was covered in *Chapter 1*, *Before You Begin*, in the oVirt components relationships section.
    - **Disable strict user checking**: When this option is disabled, two users can try to log in to a VM remote console and the latter will override. When option is enabled, one must reboot the VM before the other one can log in to the VM.
- **Host**: It lists the settings related to the startup and migration of virtual machines:
    - **Start Running on**: It determines where the virtual machine should run, on **Any Host in Cluster** or on a certain **specified** host. When the **Specific** option is chosen, we need to determine a particular host for the virtual machine to run on.
    - **Migration Options**: It determines settings related to live migration and allows the configuration of automatic or manual migration.

- **Use Host CPU**: When this checkbox is enabled, it allows you to run a virtual machine with an emulation of physical processors that are installed on virtualization hosts. In this case, the virtual machine may use all the features of the physical processor. The list of flags that define these features can be seen using the Linux console command `grep -m1 flags /proc/cpuinfo`. A set of flags can vary depending on the processor vendor, model, and family. This mode is disabled when automatic migration is allowed.

- **High Availability**: It is used to specify the settings associated with high availability features. Virtual machines can be configured with high availability options. In this case, all the affected virtual machines will be restarted as soon as possible on another physical host, thus minimizing downtime. This requires configured power management. In this section, you can also specify the priority of the queue that is related with run/migration tasks. In the case, where you want to start or migrate several VMs, it will first be done for the machines which have a high priority and will be done last for machines with low priority.
- In oVirt 3.3, add watchdog support, this allows adding watchdog cards into virtual machines. In this case, we can monitor the VM's operating system responsiveness through virtual watchdog device and configure specific actions that will be performed in case the operating system stops responding.
  - **Watchdog model**: It is used to specify the watchdog card model, *i6300esb* or *ib700*.
  - **Watchdog Action**: It is the action performed when the system is not responding, which can be reset, power off, pause, dump, or none.
- **Resource Allocation**: This section allows managing the allocation of resources such as CPU, RAM, and Storage.
  - **CPU Allocation**: It is known as *CPU Affinity* or *CPU Pinning*. It allows you to bind virtual CPUs to the physical CPU cores on the host system. This binding allows the execution of separate virtual machines on different cores of virtualization hosts. But in some cases, performance degradation may occur if the total number of virtual processors is more than the total number of physical cores, or if one physical core is associated with more than one virtual processor. The format used is: `vCPU#pCPU[_vCPU#pCPU]`. A few examples are as follows:
  - `0#1111 => pin virtual CPU 0 to physical CPU 1111`
  - `0#55_1#66 => pin virtual CPU 0 to physical CPU 55 and pin virtual CPU 1 to physical CPU 66`

- `2#62#6-10,^88 => pin virtual CPU 22 to physical CPU set 66 to 1010, excluding 88`

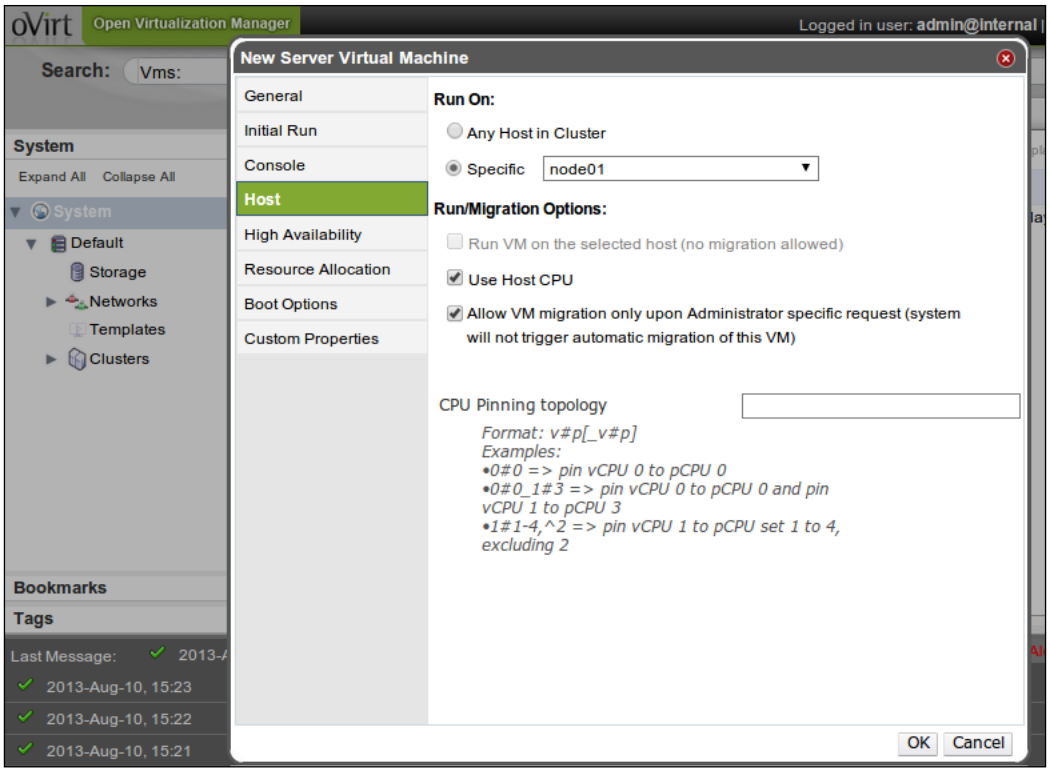
CPU Pinning is a complex mechanism where it is very easy to do the wrong configuration. As a result, performance can be degraded. A good explanation about how to configure CPU affinity can be found at: `http://frankdenneman.nl/2011/01/11/beating-a-dead-horse-using-cpu-affinity/`.

- **Memory Allocation** settings with **Physical Memory Guaranteed**: This option allows oVirt to over-commit until the virtualization host reaches the limit. Over-commit ensures that it will never run out of resources. For server-optimized VMs, oVirt Engine allows 150 percent over-commit and for desktop-optimized VMs, it allows 200 percent over-commit. These limits can be overridden with the `engine-config` command-line utility by editing the `MaxVdsMemOverCommit` and `MaxVdsMemOverCommitForServers` options.
- **Memory Balloon Device Enabled**: This checkbox enables memory ballooning, which used to reclaim unused memory.
- **Storage Allocation** settings with **Template Provisioning**: It is available only in the case of a machine from a template. Storage space can be identified in two ways:

  Thin: *Thin Provisioning* is used when the space is allocated on demand. In this case, only the deltas from the disk VM is compared to the template disk and are written on demand. This means that this VM depends on the template's disk and we can't remove the template unless this VM exists.

  Clone: In in this case, the virtual machine disk is cloned from the template disk. In this case, the VM is independent of the template and we can choose whether this cloned disk shall be allocated via thin provisioning or preallocation.

The following screenshot shows the **Host** tab settings, which will run Server on specific host, use Host CPU, and migrate only with administrator's request.



- **Boot Options**: This section contains combined options which will affect the virtual machine's boot up:
    - **Boot Sequence**: It specifies the order of boot devices to be used. The selections available here are Hard Drive, CD-ROM, PXE, and so on. You can also connect a CD in the form of a previously uploaded ISO image. Note that for installation of VMs, it's not necessary to change the boot order because the `RunOnce` mode can be used, and the boot order can be overridden for the first VM startup.
    - **Linux Boot Options**: It determines the boot configuration if a Linux operating system is installed on the virtual machine. In the **kernel path** and **initrd path** options, we can specify the kernel and initrd images stored outside the virtual machine, such as a dedicated directory inside the host system or on a shared storage that is connected to all the virtualization hosts. Thus, you can use the single kernel image and single initrd image to boot multiple machines. When booting from the outside kernel, we can pass additional parameters to the kernel with the **kernel parameters** option. The system will use these kernel parameters when booting virtual machines.

- **Custom Properties**: In this section, we can set up additional settings related with VDSM hooks (more info can be found at `http://www.ovirt.org/VDSM-Hooks`). If VDSM hooks aren't in use or are not installed on the hosts, there are no settings in this section. We can also set up miscellaneous parameters here. For example, `vhost` parameter can significantly increase the performance of a network (for additional information, please refer to `http://www.linux-kvm.com/content/how-maximize-virtio-net-performance-vhost-net`). This is particularly effective in case of network communication between machines located on the same host. To enable `vhost`, select it from the list and enter the value `eth0:true`.

We've pretty thoroughly examined all sections, but in the simplest case, we can specify a name, choose the operating system type, specify the boot device, and leave the other values by default. After filling in all the desired options, click on **OK** to start creating the virtual machine.

The next stage is creating and attaching **Network Interfaces** and **Virtual Disks** for the empty virtual machine. When oVirt Version 3.3 or above is used, Network Interface is created and attached. After you click on **OK**, the **Guide Me** wizard opens, which allows you to create virtual devices quickly.

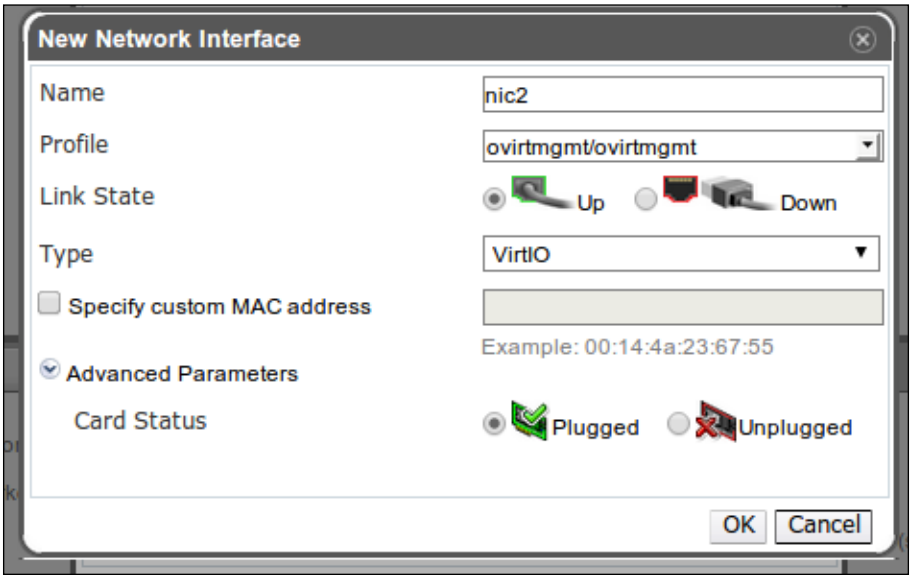
Configuring network interfaces

To create a network interface, click on the **Configure Network Interfaces** button. In the dialog box, configure the following network interface parameters:

- **Name**: It is used to specify a name for the network interface.
- **Profile**: It specifies the network profile and logical network to which the interface is connected.
- **Link State**: It indicates the status of the interface, **Up** or **Down**.
- **Type**: It indicates the type of interface. It is one of the most important options that affects performance. This option determines which driver to use for this interface at the level of virtualization. It is possible to use devices that emulate a network card *RTL8139* and *e1000*. However, the paravirtual VirtIO devices are more productive than the emulated network cards. When using VirtIO drivers, we should be aware if support for VirtIO should be included in the VMs, operating system. All modern Linux distributions have VirtIO driver support in the kernel. For Windows, supporting drivers need to be installed separately.
- **Specify custom MAC address**: It is be used to set the network interfaces' MAC address.

- **Advanced Parameters** with **Card Status** allows changing virtual network card status in a virtual machine as **Plugged** or **Unplugged**.

The following screenshot shows the **New Network Interface** dialog window:

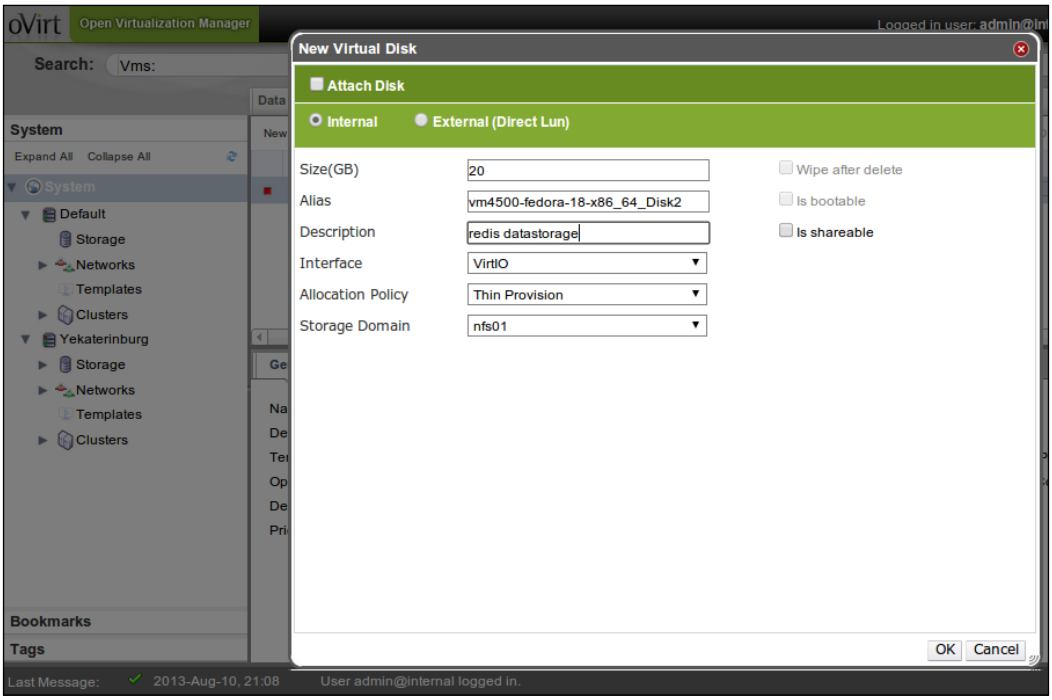


For filling the parameters, click on the **OK** button and then start the process of creating a network interface.

Configuring virtual disks

Go back to the **Guide Me** wizard. For virtual disk creation click on the **Configure Virtual Disks** button. Before creating the disc, it is worth noting that the virtual disk can be *internal* or *external*. It is about the relationship between a virtual disk and used storage that is connected to a data center. Internal means that the image of the virtual disks are placed on a storage configured in the oVirt storage domains section. External means that the virtual disks rests on a storage not managed by oVirt, for example, this can be a single block device (LUN) provided by SAN storage.

In the following screenshot, the **New Virtual Disk** with **Internal** disk type dialog box is shown:



Click on **Configure Virtual Disks**; it opens a dialog box where we can create a virtual disk or connect to an existing disk. Following are the options for specifying the virtual disk:

- **Internal** or **External**: These radio buttons determine the type of virtual disk.
- **Size**: This specifies the size of the virtual disk. It is not available **External** because the virtual disk is given full space from the allocated LUN.
- **Alias**: It specifies the alias that will be associated with the virtual disk.
- **Interface**: It is used to specify which interface will operate a virtual disk. The choices available are IDE interface emulation and VirtIO paravirtualized interface. It is recommended to choose VirtIO because it provides better performance compared to the IDE. Note that in this case, VirtIO drivers must be installed on the guest OS.
- **Allocation Policy**: It is used to define the strategy of allocating disk space. You can just select the entire amount specified in the **Size**, or use **Thin Provisioning** in which space will be allocated as needed. It's available only for the **Internal** virtual disks.

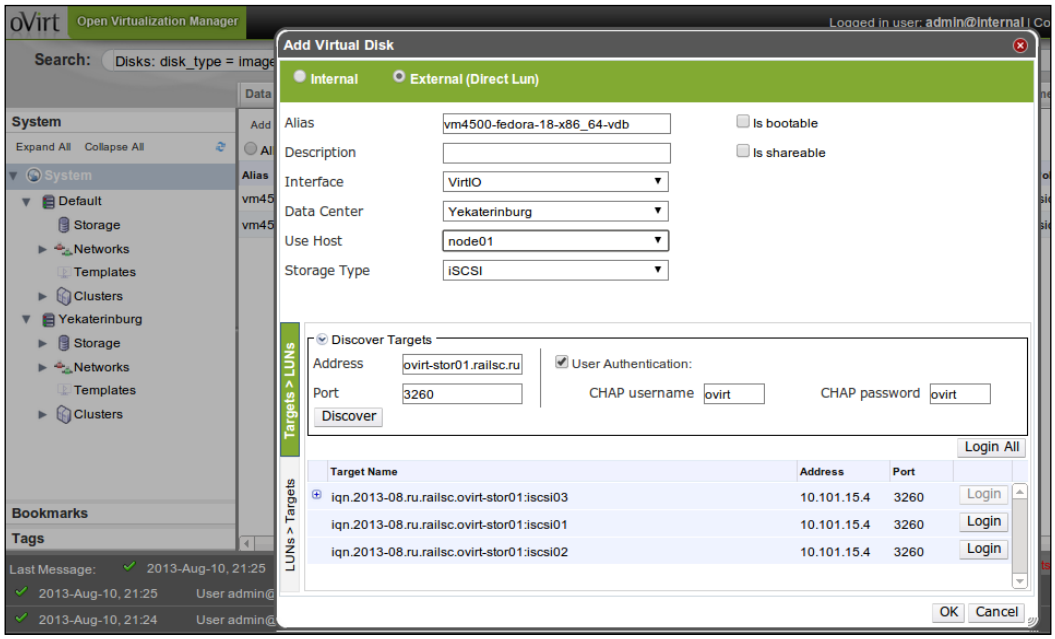
- **Storage Domain**: This specifies the storage where the virtual disk will store.
- **Wipe after delete**: It is used to write zeros to disk when delete is performed.
- **Is bootable**: It marks the disk as bootable.
- **Is shareable**: It is used to mark the disk as shareable. The shared disk feature should provide the ability to attach a disk to multiple VMs. It is the user's responsibility to make sure that the VMs do not corrupt disk data. The synchronization/clustering of data access to shareable disks is the responsibility of the guests. Attaching a shareable disk to non-cluster aware guests will lead to corruption of the data on the disk.

In the case of an **External** disk, we must specify the following additional parameters:

- **Use Host**: It specifies the virtualization host to be mounted on LUN.
- **Storage Type**: It specifies the SAN storage type, iSCSI or Fibre Channel.

  If you choose the iSCSI storage type, an additional iSCSI panel designed to find and connect iSCSI targets will be opened. If you select the Fibre Channel storage type, an area below it will be shown in the list of available LUNs.

The following screenshot shows **New Virtual Disk** with **External** disk type dialog box:



- **Attach Disk**: With this checkbox, you can connect a virtual machine to a virtual disk that has been created before. If enabled, the area below will display a list of available virtual disks. Additional disks can be created in the **Disks** tab in the resource pane.

Fill in your parameters and then click on **OK**. oVirt Engine will start the process of creating a virtual disk.

Running the Linux virtual machine

Go to the **Virtual Machines** section in the resource pane and select the virtual machine from the list. A toolbar is present above the list of virtual machines. Click on the **Run** button shown as an audio/video player play button. This is used to start the virtual machine. The virtual machine's startup progress can be seen in the event pane. After starting the connection to the virtual machine remote console, you have to click on a button with a picture of the monitor in the toolbar above the list of virtual machines. After clicking on the button, oVirt displays a window where the details to access the virtual machine remote console will be displayed. We use SPICE console access. SPICE clients can be downloaded at `http://spice-space.org/download.html`.

Creating a Windows desktop

To install the Windows machine, it is first necessary to get a floppy image with VirtIO drivers. This is required when VirtIO devices are used. For the IDE disk and RTL8139, this is not required. This image can be installed with the `virtio-win` package. This image will need to install VirtIO drivers to Windows machines, so you need to upload it into the ISO repository with `engine-iso-uploader`. oVirt automatically detects the format of the image when uploaded into the ISO storage. Surprised? Once the image is uploaded, we can proceed to creating a VM:

1. Go to the **Virtual Machines** section and click on **New VM**.
2. In the main section, set **Name** and choose Windows-like operating system's.
3. Specify desktop in the **Optimized for** field. We may change other settings and options but in this example, we will leave the defaults. Click on the **OK** button to start the process of creating the virtual machine.
4. Continue by clicking on the **Guide Me** button. This will define the logical networks for the created virtual machine. If required, press **Configure Network Interfaces** and add the network interface. For more details, refer to the section *Configuring network interfaces*. Set VirtIO's network interface type.

5. After adding the network interface, return to the **Guide Me** window. Click on **Configure Virtual Disks** and add a virtual drive to the VM. Refer to the section, *Configuring virtual disks*, for details. Set VirtIO's virtual disk type.
6. Close the **Guide Me** wizard. Windows virtual machine is ready for further installation.

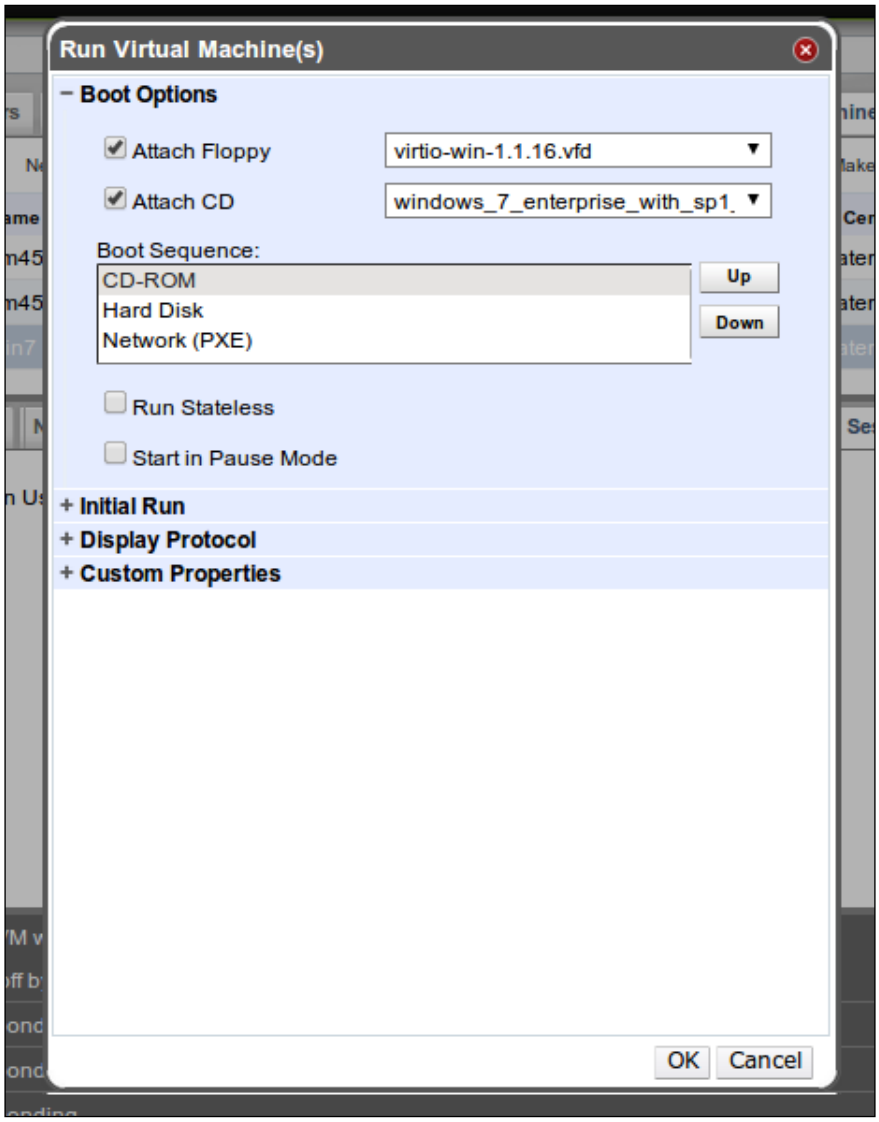
Running and installing Windows as a guest operating system

Remember that we configure VM with VirtIO devices. In this case, start the virtual machine in RunOnce mode. In this mode, we must connect a floppy drive image that contains VirtIO drivers. Windows doesn't ship with VirtIO drivers by default, so failure to select a driver means that the Windows installer will fail to detect a hard disk. When VirtIO devices are not used, this step can be skipped. The major steps for installation are as follows:

1. Right-click on the virtual machine and click on the **Run Once** item. In the dialog window, configure the following settings:
    - **Attach Floppy**: It is used to select the virtual floppy drive with VirtIO drivers, which were uploaded with `engine-iso-uploader`.
    - **Attach CD**: It is used to attach the operating system ISO image.
    - **Boot sequence**: It is used to select a CD-ROM device.
    - **Display protocol**: It is used to choose VNC or SPICE.

    Leave the default values for the other parameters and click on **OK** to start the virtual machine.

The following screenshot shows the **Run Once** dialog window with **Boot Options**:



2. Select the virtual machine from the list and click on the **Console** icon. Installation of the operating system will begin from the opened window.

3. The only thing that needs to be done manually is to specify the path to the VirtIO drivers that are located on the virtual floppy disk image (VFD). Otherwise, default values will be accepted by all the parameters. To do this, select the **Custom installation** option and click on the **Load Driver** button. After loading the driver, VirtIO devices will be detected by the installer and it will continue the installation process.
4. When Windows installation is finished, eject the CD and reboot the VM. After reboot, we can connect via a VNC or a SPICE client to the virtual machine remote console.

Using Templates

Now that we have built and run virtual machines, we can proceed to use **Templates** and save the virtual machine settings into a Template. Templates are based on virtual machines. When we create a Template, oVirt Engine will build a container with all the source VM settings. For storage, a new device is created as a separate virtual disk which will be permanently associated with the Template. When you create VM based on the Template, the Template's storage can be used for VM virtual disk creation.

Network configuration for the new VM will be inherited from the configuration that is stored in the Template. We can use this template for speedy creation of virtual machines from the original VM.

To make a Template, we need a virtual machine that is created and running beforehand. Before you begin, the VM needs to be prepared. This guarantees that the settings specific to the VM will not be distributed by other machines through the Template.

Preparing a Linux virtual machine

For Linux machines, preparation is to delete SSH host keys. Host keys on each server must be unique. Also, we need to remove udev network rules because machines created from templates have a new MAC address. For additional information about Linux preparation tools, refer to `http://libguestfs.org/virt-sysprep.1.html` and `http://linux.die.net/man/1/virt-sysprep`. The steps for creating a Linux virtual machine are as follows:

1. Via SSH, connect to the virtual machine that will be used as the template source. Now we need to create a mark that indicates the need for reconfiguration. Run the following command as a root for creating a mark in system:

   ```
   # touch /.unconfigured
   ```

2. Remove the SSH host keys:

```
# rm /etc/ssh/ssh_host_* -rf
# rm /etc/udev/rules.d/70-persistent-net.rules
```

3. Halt the virtual machine:

```
# shutdown -h now
```

4. Once the virtual machine is halted, it is ready to be used for a Linux virtual machine Template.

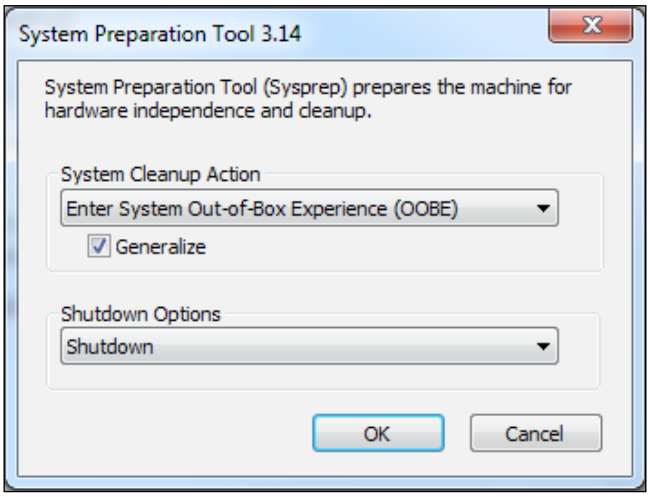
Preparing a Windows virtual machine

To prepare virtual machines with Windows, the *sysprep* utility is recommended. This guarantees that the settings specific to the VM will not be distributed to other machines through the template. The following procedure is suitable for creating templates based on Windows 7 and Windows 2008 R2:

1. Create a snapshot of Windows machine as described in *Chapter 5*, *oVirt Advanced Features*. Turn on VM booting from the snapshot.
2. Log in to the Windows virtual machine with the administrator account and press *Windows* + *R* on the keyboard. In the **Run** dialog window, type `regedit` and click on **OK**.
3. This invokes the **Registry Editor** program. Navigate to the left side to **HKEY_LOCAL_MACHINE** | **SYSTEM** | **SETUP**.
4. In the main window, add a new string value by right-clicking and navigating to **New** | **String Value**. Set the value name as `UnattendFile`. When the name is set, right-click on the added parameter and click on **Modify** in the context menu. In the dialog box **Edit String**, specify `a:\sysprep.inf` in the value data field.
5. Run the sysprep utility from `C:\Windows\System32\sysprep\sysprep.exe`. When the sysprep application starts, specify the following sysprep settings:
    - **System Cleanup Action**: Here we need to specify the **Enter System Out-of-Box-Experience (OOBE)**
    - **Generalize**: This checkbox ensures that the *system identification number* can be changed
    - **Shutdown Options**: Choose **Shutdown** to power off after preparation

    Click on **OK** and the virtual machine will now run the sealing process. And after finishing this, the system will power off automatically.

The following screenshot shows the sysprep utility dialog box:



6. After template creation, we must revert the snapshot as the master VM. This is required because if we run sysprep *three times* on Windows, it won't start anymore. With the snapshot trick, we can prevent this.

Note that Windows requires product keys. We can set these keys with the `engine-config` command-line utility and then restart oVirt Engine with the following commands:

```
# engine-config -set ProductKeyWindows7=<key> --cver=general
# service ovirt-engine restart
```

For getting all the available product keys, run the following command:

```
# engine-config -l |grep -i productkey
```

Creating Template

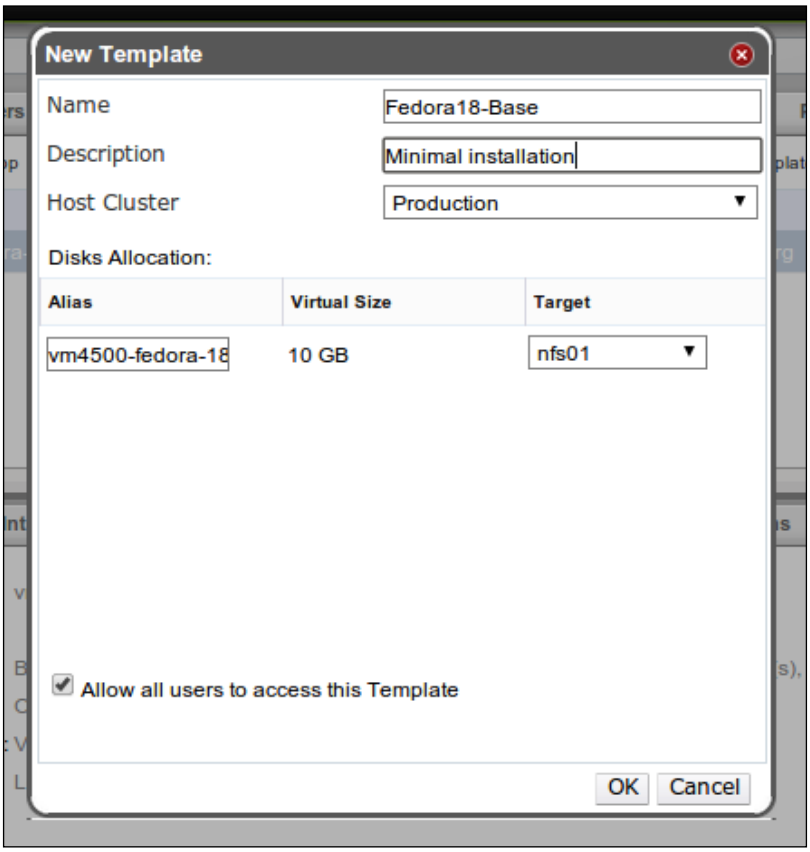
Now when the virtual machines are prepared and turned off, we can create a Template:

1. In the **Administrator Portal**, click on the **Virtual Machines** section. Select the prepared VM from the list. Ensure that it is powered off.
2. Right-click and select the **Make Template** item. The **New Template** dialog window is displayed where we need to specify information related to the template:
    - **Name** and **Host Cluster**: It is used to specify the new Template name and target parent cluster.

- **Disk Allocation**: It is used to specify the virtual disk allocation.
- **Allow all users to access this Template**: This checkbox assumes that the Template will be available for all users. This Template can be used by everyone.
- **Copy VM Permissions**: It allows you to make Template with permissions assigned to the source VM.

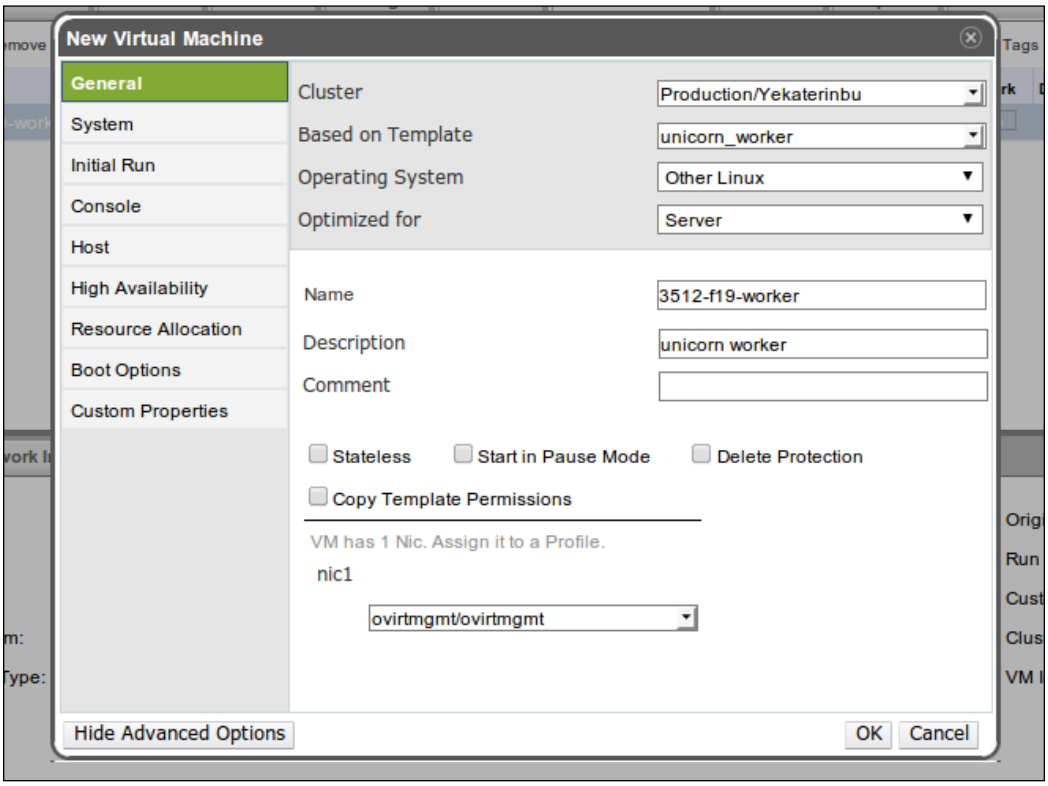
3. Click on **OK** and template creation will begin. The source virtual machine will change the status to *Image Locked* and will show this status until the template is created. After some time, depending on the virtual disk size, the template is created and it appears in the **Templates** section. During creation, the toolbar action buttons for the template are deactivated. After creation, the action buttons are activated, which shows that the template is created and ready for use.

   The following image shows the **New Template** dialog box:



Using Templates for virtual machine creation

Now we have a ready-to-use Template. This Template contains preconfigured storage, networking, virtual machine-related settings, and an installed guest operating system with packages and applications. Now we can deploy a preinstalled virtual machine from this Template. The following screenshot shows you how to add a virtual machine that is based on the Template:



The following steps show you how to create a VM using Templates:

1. Go to the **Virtual Machines** section and click on **New VM** on the toolbar.
2. In the **General** section, click on **Based on Template** and select the created Template from the list.
3. Specify **Name** and add **Description**. Other settings will be filled in by the Template but we can change them; it is not forbidden.

4. Click on the **Resource Allocation** section. Select the **Clone** option in the **Template Provisioning** setting. The remaining settings can be left unchanged. Click on **OK** to begin creation of the virtual machine. When creating a VM is complete, it will be added to the list of VMs and will be ready to run.

Summary

In this chapter, we looked at the options for creating VMs through **Administrator Portal**. A VM can be created in oVirt in two ways, from scratch or from a preconfigured Template. The process of creating VM in oVirt offers a very large number of available parameters. This allows creating a very flexible configuration for a VM in future. Till this stage, we have successfully completed the setting up of oVirt. In *Chapter 5*, *Advanced Features*, we consider the advanced features of oVirt that make it even more attractive.

5
Advanced Features

oVirt is a rapidly growing product. In each new version, there are new features and functions that make oVirt attractive and really interesting. In this chapter, we will learn about the functions and features that have appeared in different versions of oVirt and how to use them.

Live migration

Live migration is the process of moving running virtual machines between physical servers. All the information about running processes, such as memory, open files, and network connections, which are stored in memory, are transferred to the destination host that is part of the new virtual machine. During live migration, the total amount of VM memory transferred and incremental changes are retransmitted. The user does not notice the pause that occurs during the final transfer of memory and between stopping and resuming a virtual machine.

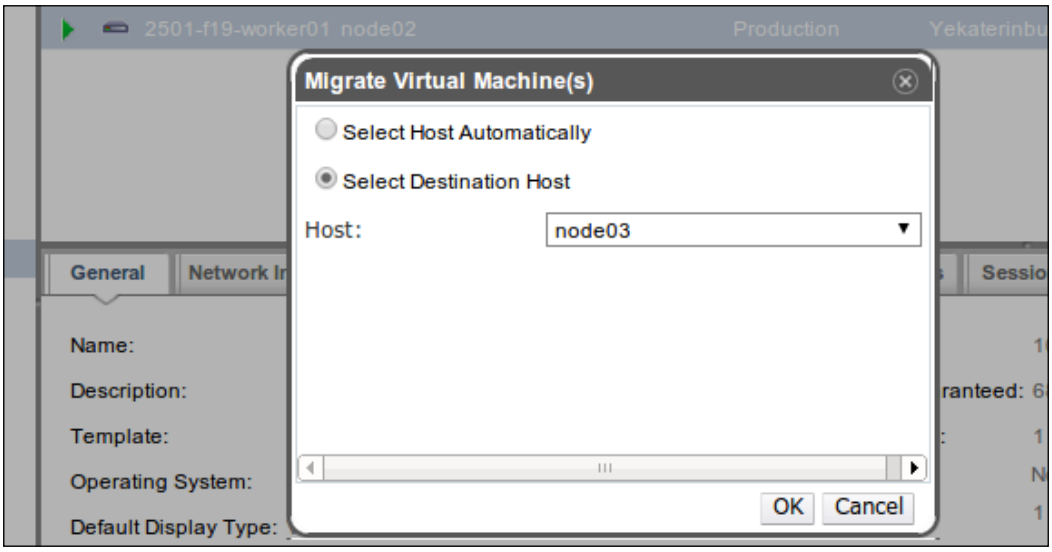
Live migration in oVirt has certain features that are worth keeping in mind:

- Live migration can be performed only within the cluster. Live migration between clusters is not available.
- Live migration can be performed automatically. This can be rebalanced with the use of a cluster policy.
- Live migration can be disabled for certain VMs.

Manual live migration can be performed as follows:

1. Go to the **Virtual Machines** section in the resource pane, select the virtual machine from the list, and click on the **Migrate** button in the toolbar.
2. In the dialog box, select the destination host and click on **OK**, then start the migration process. The live migration progress can be seen in the event pane.

The following image shows a migration dialog:



High availability

High availability is the ability to restart virtual machines on other physical nodes in the event of a hardware failure or scheduled scenarios, which will be described in a moment. The configuration of high availability is accessible only for virtual machines that are allowed to perform automatic migration. It is recommended to configure high availability for virtual machines running mission-critical tasks. High availability ensures that the virtual machine will be restarted in the following scenarios:

- When the virtualization host becomes nonoperational as a consequence of a hardware failure
- When the virtualization host goes into maintenance mode for scheduled downtime
- When the communication between the host and external storage or between hosts is lost
- When the virtual machine is no longer available due to a failure of the virtual guest operating system.

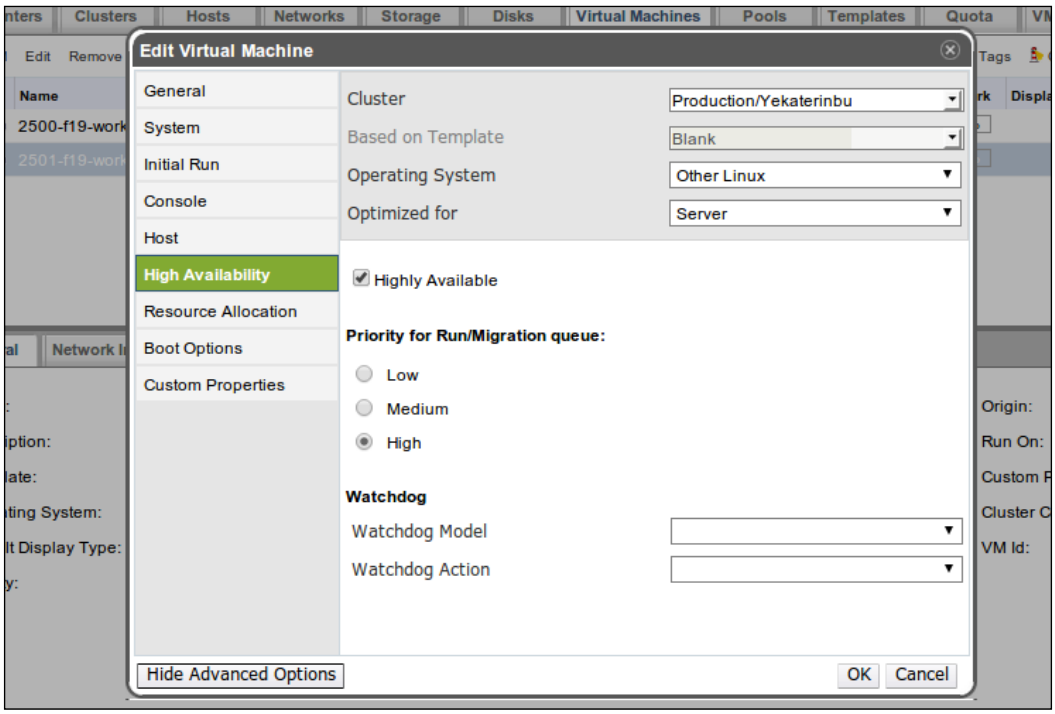
With a highly available VM, service downtime is reduced to a minimum since the virtual machine will restart automatically within a few seconds after the detection of the failure. In this case, administrator intervention is not required. Until Version 3.3, high availability options were available only for virtual machines such as servers.

Starting with Version 3.3, high availability options can be set for both types of machines, servers and desktops.

At first we must enable automatic migration and then perform the following steps:

1. Go to the **Virtual Machines** tab in the resource pane and select the virtual machine from the list. Then in the toolbar, click on the **Edit** button.
2. In the dialog box, go to the **Host** section and check that the **Migration Options** setting is set to **Allow manual and automatic migration**.
3. Go to the **High Availability** section and enable the **Highly Available** checkbox.
4. Set the **Priority for Run/Migration queue** option to **High**. Machines with high priority run first, those with low priority run last.
5. Click on **OK** and the settings will take effect.

The following image shows a **High Availability** section in the **Edit Virtual Machine** dialog box.



Cluster policies

oVirt allows you to configure the clusters in load balancing mode or power saving mode. When setting up oVirt, it is recommended to determine the clusters for one of the policies. For a cluster, you can assign only one policy. By default, there are two policies available:

- **Even Distribution**: In this mode, the load is distributed to all nodes in the cluster. To apply the policy, all virtualization hosts are measured by the load on the CPU.
- **Power Saving**: In this mode, the load is distributed on a minimum number of nodes in order to save energy.

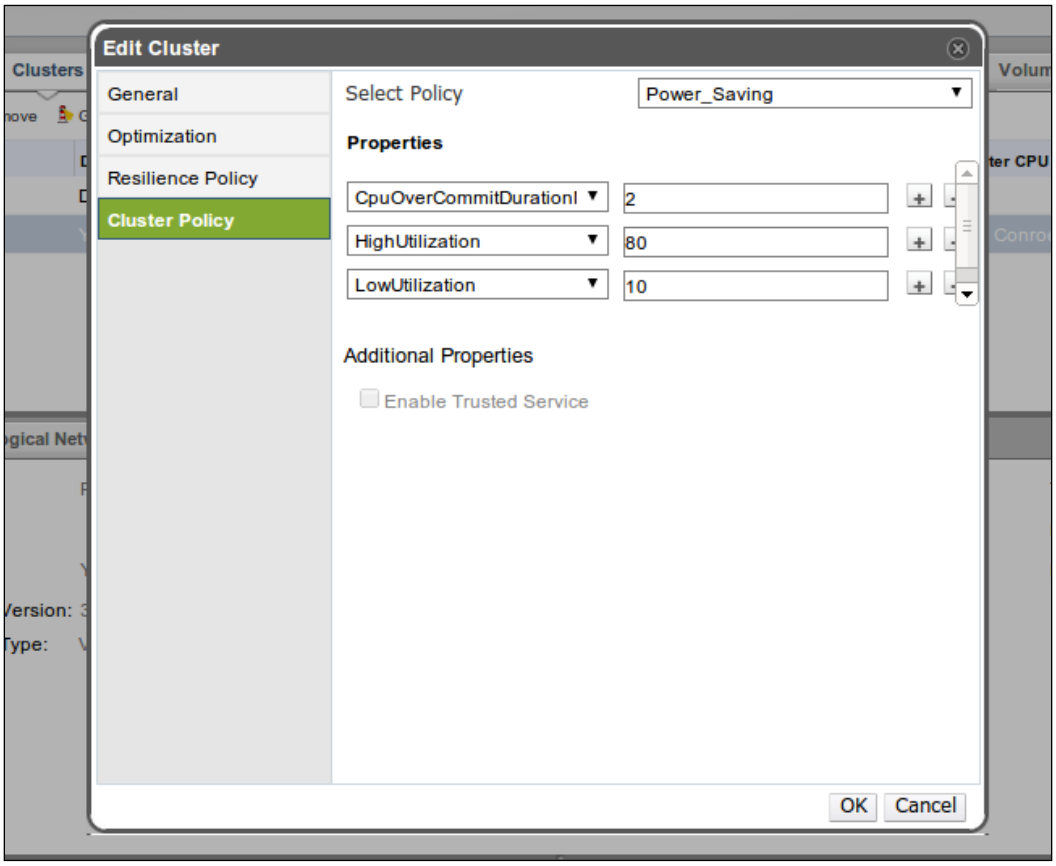
For setting up the cluster policy to the cluster, perform the following steps:

1. Select the **Clusters** tab in the resource pane. In the list of clusters, select the desired cluster and click on **Edit**.
2. In the dialog box, select the **Cluster Policy** section in the left-hand side of the window. In the right-hand side of the window, choose either one of the policies in the **Select Policy** section: **Even_Distributed** or **Power_Saving**.
3. When the policy chosen is activated, it will display the following policy properties:
    - **HighUtilization**: When the maximum level of load in the **Even_Distributed** policy is reached, oVirt will start the process of rebalancing and migrating machines.
    - **LowUtilization**: When the minimum acceptable level of load in the **Power_Saving** policy is reached, oVirt will start the process of migrating machines to other hosts in the cluster.
    - **CpuOverCommitDurationMinutes**: This parameter is used in the **Even_Distributed** policy and specifies the time. If during this time CPU usage is higher than the value stated in the High or Low Utilization, the scheduler will start to migrate the virtual machines to other hosts for load rebalancing purposes.

    Let us consider an example of the cluster policy. The hosts who have reached the maximum service level (HighUtilization) become unavailable for running virtual machines on them. For already-running virtual machines, an automatic migration process will start to the under-utilized hosts. A host that has reached the minimum service level (LowUtilization) performs automatic migration of running virtual machines to other hosts. Thus, the host is available to be turned off in order to save power.

4. Set the value for the policy and click on **OK**.

The following image shows the dialog window for **Cluster Policy**:

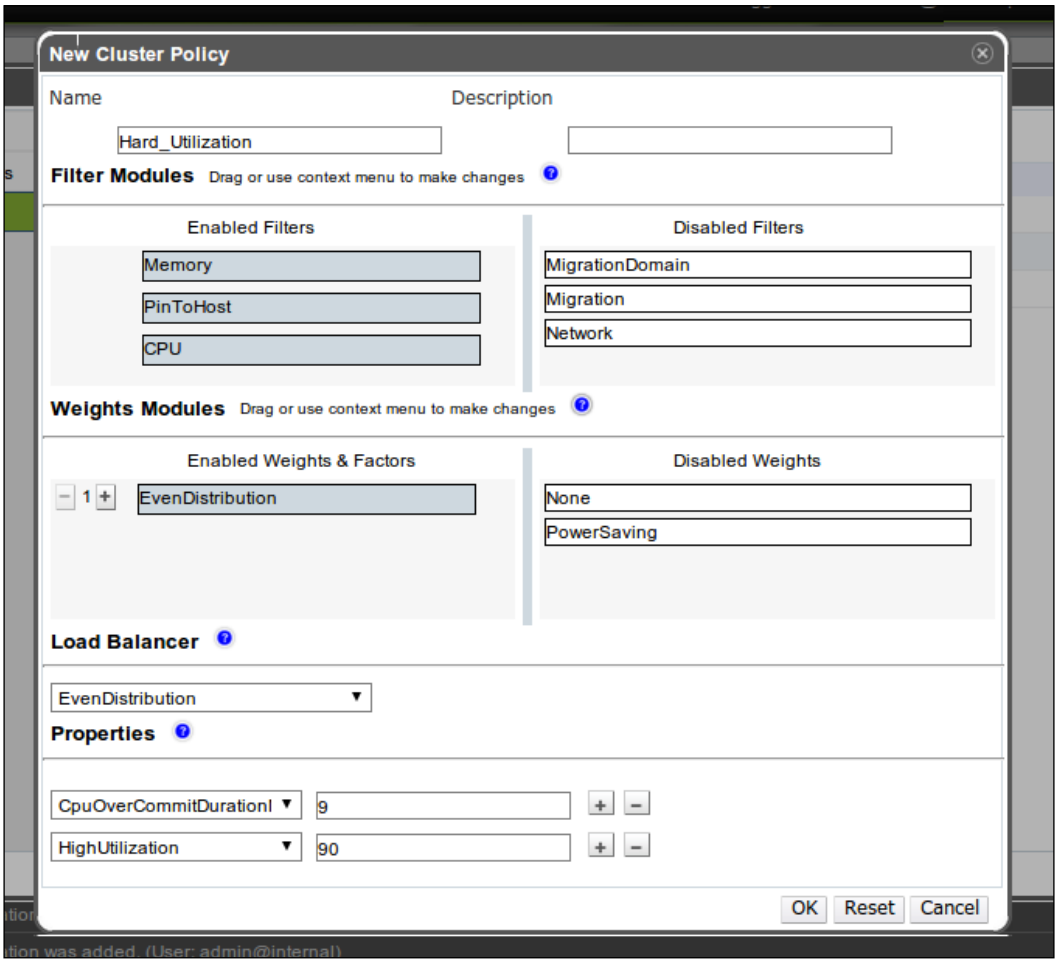


Another innovation of oVirt Version 3.3 is the ability to create your own cluster policies and bind these policies to the clusters. In the default policies, a specific set of filters are used that cannot be changed. Custom policies can create your own cluster rebalancing rules with a variable set of filters and options. This is useful if you realize that the default policies do not suit you. Perform the following steps to create a custom policy:

1. Click on the **Configure** button in the **Header** bar located in the admin portal's upper-right corner.
2. In the **Configure** dialog box, on the left-hand side, select the **Cluster Policies** section and click on the **New** button on the list of existing policies.

3. In the **New Policy Cluster** dialog window, set a name and fill in the policy settings as follows:
    - Enable the filters in the **Filter Modules** section. Filters are hard constraints that define the ability to run virtual machines. Each filter implements some logic to check the virtualization host. These checks are performed when the cluster policy is applied after virtual machines start automatic migration and oVirt Engine searches for destination hosts. Hosts that do not pass these checks are filtered out when choosing a destination host. This performs the optimal choice of the destination host for migrated VM's
    - Specify **Weights Modules** that define the soft constraints for running virtual machines. It is considered that the lower the mark, the better. The host with the smallest weight is selected by the scheduler first. Each module defines a particular logic. For example, to optimize the CPU load, the module will calculate the load on each host. The weights modules' scores are summed, so we can specify more than one module. One method for adjusting the priorities is to increase or decrease the used factor
    - Determine **Load Balancer** to understand which of the hosts are overloaded and which are not. After determining the balancing mechanism, the scheduler will attempt to perform automatic migration from overloaded hosts to the under-utilized ones. It is important to choose a policy that will not conflict with the weights modules. Otherwise, you may move a cluster in an unstable state. This action is supported by the choice of only one balancing policy
    - Fill the **Properties** section. Properties are necessary for one of the aforementioned modules and appear only when they are needed. The values indicated in the next screenshot are the default values at the time when the policy is assigned to a cluster. And at the same time, with the assigning to a cluster, these values can be overridden
4. Set the values and click on **OK**. The established policy will be displayed in the list of policies.

The following image shows the dialog box for creating a new policy:



The created policy can be applied to clusters. Now, when the loads are in the specified limits, oVirt Engine starts the process of rebalancing and performs the necessary migration of virtual machines for compliance with the policy chosen.

Device hot plug and hot remove

In the maintenance of the virtualization environment, we sometimes face different issues and challenges. One such problem is to change the virtual hardware configuration for virtual machines. For example, we need to add or remove a certain amount of memory, change the number of processors, or attach an additional virtual disk or a network interface. In oVirt there are also hot pluggable virtual disks and network interfaces that can be added without stopping the virtual machine. Hot plugging is only available for VirtIO devices. Other types of devices can be connected only when the machine is turned off.

Virtual disk hot plug

In order to hot plug a virtual disk, perform the following steps:

1. Go to the **Virtual Machines** tab and select a virtual machine from the list.
2. In the detail pane, select the **Disks** tab, which displays a list of virtual disks belonging to a selected virtual machine. In the toolbar, click on the **Add** button that opens the dialog box disk hot plugging.
3. The dialog window is similar to the dialog for creating a virtual disk, which we have seen in the process of creating a virtual machine. In this window, we must also fill in the parameters to connect the drive.
4. We are already familiar with these options and they should not cause any problems. Fill parameters and click on **OK**. In the list of virtual disks it will display the disk we just created.
5. Select the created disk and click on the **Activate** button in the toolbar. This starts the process of activating and connecting the disk to the virtual machine.

Inside the Linux virtual machine, make sure that the drive is connected via a command `lsblk` that shows a list of block devices available to the system. In the Windows VM we can use **Device Manager** (can be started with `devmgmt.msc` in the **Run** dialog box).

For the virtual disk hot remove, perform the following steps:

1. Click on the **Deactivate** button to turn off the drive, and then click on **Remove**.
2. In the **Remove Disk** window, select the **Remove Permanently** checkbox and click on **OK**. The disk will be disconnected and removed.

It should be noted that prior to removal of the disk, all the processes using the disk should be complete and the filesystems on the disk should be unmounted.

Network interface hot plug and hot remove

In order to hot plug the virtual network interface, perform the following steps:

1. Go to the **Virtual Machines** tab, and in the list of virtual machines choose the right one.
2. In the detail pane select the **Network Interfaces** tab, which will display a list of network interfaces belonging to the virtual machine. On the toolbar, click on the **New** button.
3. In the **New Network Interface** dialog box, fill in the parameters of the new interface:
    - **Name**: Enter the name of the interface
    - **Profile**: Fill in the previously created VNIC Profile and the logical network here
    - **Link State**: Determines the status of the network interface
    - **Type**: The user needs to define the interface type and specify VirtIO
4. Fill the parameters and click on **OK**.

In the virtual network, interfaces will be displayed in the network interface we just created. The activation process is executed automatically and does not require administrator intervention.

Inside the Linux virtual machine, make sure that the network interface is connected via the command `ip link show`, so that it shows a list of available network devices. In the Windows VM, we can use **Device Manager** again.

To hot remove the network interface, perform the following steps:

1. Select the desired interface in the list of network interfaces and click on **Edit**. In the dialog box, note the following options:
    - **Link State**: Set this to **Down**
    - **Card Status (Advanced Parameters)**: Set this to **Unplugged**
2. Click on OK, and then shut down the network interface.
3. Next click on the **Remove** button in the toolbar. Thus, the network interface is disconnected from the virtual machine.

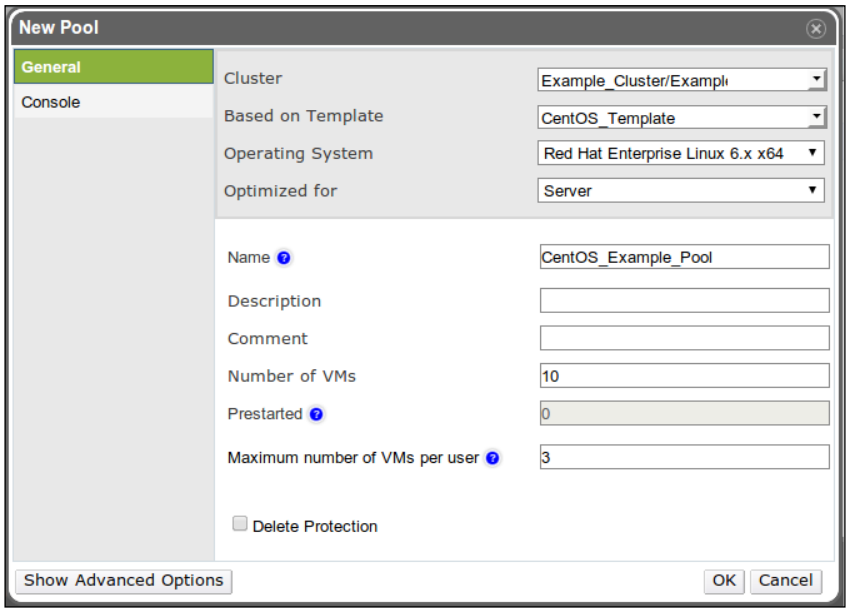
Pools and prestarted VMs

oVirt allows you to create prestarted VM Pools in which virtual machines are ready for use. The main goal of prestarted VMs is the possibility of immediate use of VMs as necessary, without waiting for virtual machines to be created. When a user shuts down the VM, it returns to the pool and then it's ready to be used by another user. These pools are created on the basis of pre-existing templates. Thus, VM Pools represent a group of identical virtual machines ready for use at any time. Virtual machines in VM Pool can be turned off, or they can be assigned to a specific user. When assigning virtual machines to the user, confirm that these machines are running before the user is able to use them. The user can start the VM manually, if he/she has appropriate permissions, they can be assigned a predefined role, UserRole.

Creating a pool

In order to create a pool with prestarted VMs, perform the following steps:

1. Go to the **Pools** tab in the resource pane. In the toolbar, click on the **New** button.
2. In the **New Pool** dialog box, which is very similar to the dialog for creating virtual machines, fill in the blank section to create VM. You will already be familiar with many parameters. However, when the pool is created, there are options that do not have to create virtual machines. In the following screenshot the **New Pool** dialog box is shown:



3. The **New Pool** option is responsible for the amount prestarted in the VM. So in the dialog box, fill in the pool settings:
    - **Cluster**: This specifies the cluster that will own the created pool.
    - **Based on Template**: In this, we define the template on which to run the virtual machines.
    - **Operating System**: Here we determine the type of operating system inside a virtual machine.
    - **Optimized for**: Here we indicate the type of optimization, Server or Desktop.
    - **Name**: Here we add a name of the new pool and a template for the names of virtual machines. This means that the name of the pool will be used to create a virtual machine's name. For example, in the pool with the name `fedora19-??`, the virtual machines will be generated by the following names: `fedora19-01`, `fedora19-02`, `fedora19-03`, and so on, up to `fedora19-99`.
    - **Description**: This is a short description for the pool.
    - **Number of VMs**: This should indicate the total number of machines in the pool.
    - **Prestarted**: This option specifies the number of VMs that will be launched in the pool and could be ready for assigning to the user. When the pool is created, this option is disabled and is available in edit mode only after the creation of the pool.
    - **Maximum number of VMs per user**: This specifies the maximum number of machines that can be assigned to one user.
4. Once the parameters of the pool are filled, click on **OK**.

After you specify details, a pool will be created for the number of virtual machines that were specified in the **Number of VMs** option. After creation of the pool, virtual machines are in the power-off state. To run multiple machines you need to perform the following steps:

1. Select the pool from the list and click on the **Edit** button in the toolbar.
2. In the dialog box, specify the number of prestarted VMs that will be immediately started after clicking on **OK**.

Also, when you edit pool settings, you can change a number of machines in the pool.

Detaching VM from a pool

If necessary, a virtual machine can be detached from the pool and can be treated as an independent virtual machine. The only limitation is that the virtual machine must be turned off beforehand. For detaching machines from the pool, perform the following steps:

1. Go to the **Pools** tab in the resource pane and select the pool that hosts the virtual machine.
2. In the details pane, select the **Virtual Machines** tab and choose the right machine in the list of virtual machines and click on the **Detach** button in the toolbar.
3. In the window that opens, click on **OK** and confirm the action.

Thus, the machine will be detached from the pool. When all the machines in the pool are faded out, the pool will terminate.

Snapshots

Snapshot is a special way to catch the virtual machine state. This feature is very useful when you need to fix the state prior to any change in the system, or vice versa after reaching a certain state. For example, this may be the software installation or a large-scale configuration update that affects a large number of system packages and files. The reasons may be very different. Creating a snapshot implies a checkpoint where the state of the filesystem is fixed. In addition to creating images for block devices, you can create a snapshot of memory.

In fact, a snapshot is a single file or block device. For storage with file access (such as NFS or Gluster), snapshot is a separate file. For storages with block access (such as iSCSI or FC) snapshot is a separate LVM volume, which is located at the LVM physical volume (oVirt virtualization hosts use the attached storage as LVM). However, snapshots cause decrease in the VM's performance. When a snapshot is created, the original disk image is frozen in a consistent state. All write accesses from the original image will go to a new differential image. Even worse, the differential image has the form of a change log that records every change made to a file since the snapshot was taken. This means that read accesses would have to read not only one file, but also all difference data (the original data and every change made to the original data). The number increases even more when you use cascade snapshots (snapshots of a snapshot). Therefore, do not abuse the snapshots unnecessarily.

oVirt has snapshots support and can perform other tasks with snapshots as follows:

- Creation and deletion of snapshots of virtual machines
- Switching to a snapshot and booting a virtual machine from a snapshot
- Returns from a snapshot to the previous state of the virtual machine
- Snapshot merging with the main block device to merge the changes into the existing snapshot
- The creation of a new machine from a snapshot

Creating snapshots

We can create and delete snapshots at any time, even when the virtual machine is up and running. To do this, perform the following steps:

1. Go to the **Virtual Machines** tab in the resource pane and in the list of virtual machines, select the desired machine.
2. In the details pane tab, select the **Snapshots**, click on the **Create** button in the dialog box, and enter the description of the snapshot.
3. If you want to create a snapshot of memory, enable the **Save Memory** checkbox.
4. Click on **OK** and then start the process of creating the image.

The process of creating the snapshot can be seen in the event pane.

Working with snapshots

Now that we've created a snapshot, we have a consistent state of the virtual machine at the time of creation of the snapshot. The snapshot may be needed if we want to view the saved state of a VM or return the VM to the state that was saved in the snapshot. It needs to run a virtual machine with settings when VM uses a snapshot as a virtual disk. In this case, the virtual machine must be powered off:

1. Go to the **Virtual Machines** tab in the resource pane and in the list of virtual machines, select the right machine.
2. In the **Snapshots** tab in the details pane, select the snapshot and click on **Preview**. The status of snapshot will change to **In Preview**, and virtual machine will be switched to the use of the snapshot.

3. Now the existing virtual machine will use snapshots as bootable virtual disks and you can boot and use VM. Any changes will be reflected in the used snapshot. The original block device of a virtual machine stays attached, but will remain unchanged.

Now, there are two ways this can be dealt with:

1. To return to the state of the virtual machine click on the **Undo** button on the toolbar. The virtual machine will be switched back to the original volume.
2. If you want to know the state of the snapshot you want to apply to the original block device, click on the **Commit** button in the toolbar. All changes are reflected in the snapshot and will be applied to the original block device.

Thus, it is possible to create various checkpoints and switch between them.

Cloning a virtual machine

We can create a new virtual machine based on the snapshot. In oVirt this ability is called cloning. The cloning process can be carried out even at the time when the original machine is turned on and working. You need to create snapshots before creating a clone of a virtual machine. Perform the following steps:

1. Go to the **Virtual Machines** tab in the resource pane and in the list of virtual machines, select the right machine.
2. Go to the **Snapshots** tab in the details pane and select the image that will serve as the basis for creating a new virtual machine.
3. Click on the **Clone** button and in the dialog box **Clone VM from Snapshot** will open.
4. This dialog box is very similar to the dialog for creating a virtual machine, so the options and parameters are something we are already familiar with.
5. Fill in the settings and options for the new virtual machine and click on **OK**.

Thus, on the basis of the snapshot we will create a new virtual machine.

Snapshots are very convenient tools that can be useful as backup tools for significant changes within the virtual machines.

Network QoS and VNIC profiles

Traffic shaping is often used in the management of the network environment. Traffic shaping allows the administrator to define policies that allow limiting the network bandwidth.

oVirt Version 3.3 has added a feature called Network **Quality of Service (QoS)**. Network QoS allows users to create restrictions for incoming and outgoing network traffic on the virtual network cards. For convenient use of Network QoS in oVirt Version 3.3, it adds an entity called VNIC Profiles. These profiles are containers in which virtual network cards' additional functions are integrated, such as Port Mirroring and QoS. These profiles are assigned to virtual network cards. Thus, with VNIC Profiles, users have the ability to configure Network QoS parameters for each network card.

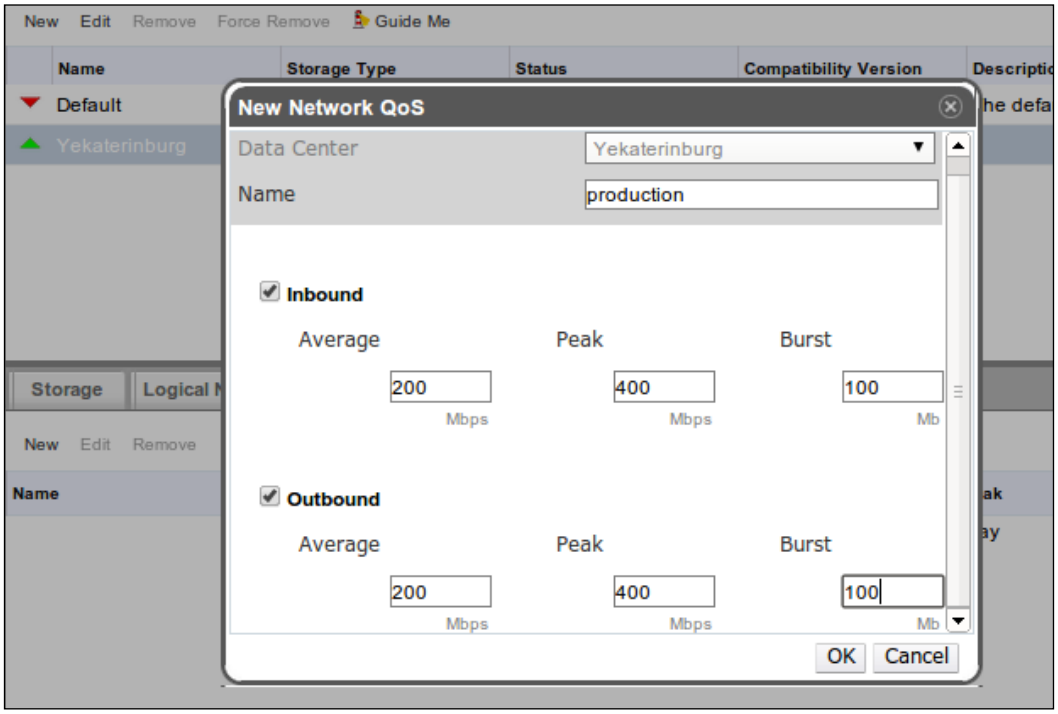
To start using QoS perform the following:

1. Create a QoS rule.
2. Create a VNIC Profile and add the QoS rule.
3. Associate the VNIC Profile with a virtual network card.

Adding a new rule to the QoS is necessary:

1. Go to the **Data Centers** tab and choose from the list the data center in which you plan to define QoS.
2. Select the **Network QoS** tab in the details pane. Click on the **New** button to create a new rule.
3. In the dialog box, **New Network QoS**, define parameters as follows:
    - **Data Center**: The data center for which the rule is created is already chosen
    - **Name**: In this, you need to specify a name for the rule
    - **Inbound/Outbound**: Check these checkboxes to apply the constraints to inbound and outbound traffic
    - **Average**: This is the long-term limit around which traffic should float in MBps
    - **Peak**: Here you can specify the maximum allowed bandwidth in burst in MBps
    - **Burst**: This specifies the permitted amount of traffic in burst in MBps
4. Specify the values and click on **OK**. A new QoS rule will be added in the list of rules.

Thus to set the rules, you need to specify what type of traffic will be used by the rule and set the value for the bandwidth. The following screenshot shows the dialog box **New Network QoS**:



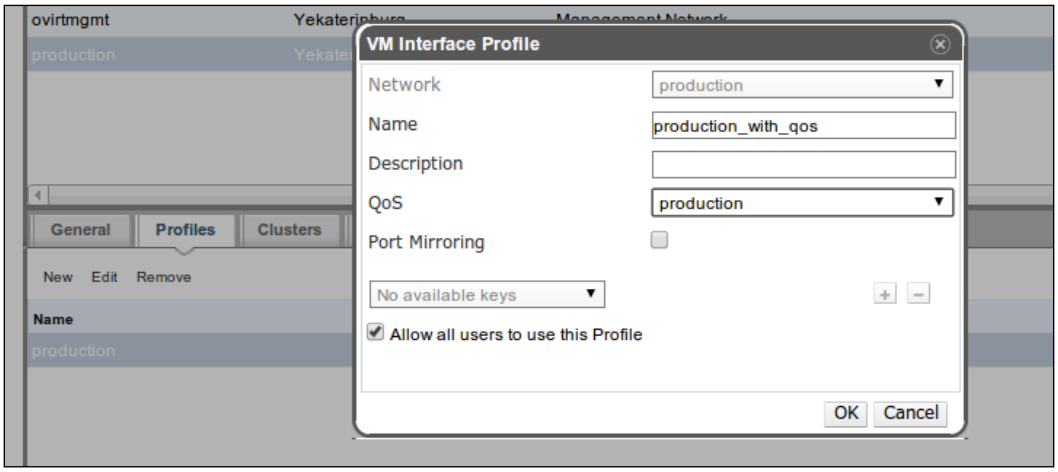
Now that we have created a rule, we can create a VNIC Profile and add the new rule:

1. Open the **Networks** tab and the list of logical networks and select the networks in which we want to use QoS settings.
2. Select the **Profiles** tab in the details pane and click on the **New** button.
3. In the dialog box, fill the parameters in **VM Interface Profile**.
    - **Network**: This specifies a logical network for which you are creating the VNIC Profile
    - **Name**: This specifies a name for the new profile
    - **Description**: Here you can optionally specify a brief description
    - **QoS**: From the drop-down list, select the QoS rule that will be used in the logical network

- **Port Mirroring**: Here you can optionally enable port mirroring
- **Allow all users to use this profile**: Check this checkbox if you want a profile to be accessible to all users

4. Complete all settings and then click on **OK**. This starts the process of creating a profile.

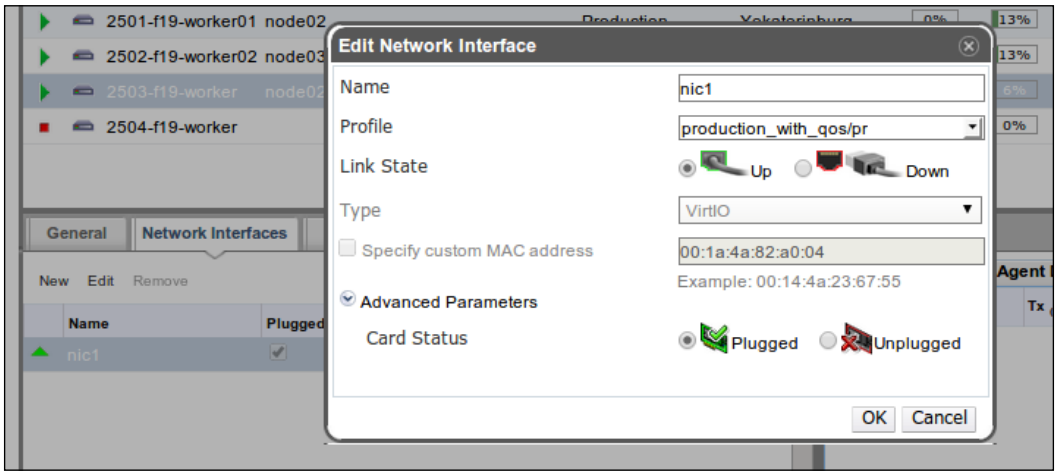
The following screenshot shows the **VM Interface Profile** dialog box:



Now that we've created a profile, we can specify the VNIC to use it:

1. Click on the **Virtual Machines** tab and select the desired virtual machine from the list.
2. Open the **Network Interfaces** tab in the details pane, select the network card from the list, and click on the **Edit** button.
3. In the dialog box **Edit Network Interface**, specify the QoS settings under the **Profile** option.
4. Click on **OK** to enable the settings.

The following screenshot shows the dialog box **Edit Network Interface**:



Now that we have applied the QoS policy for a single virtual network card, you can go even further and create more complex configurations based on QoS and VNIC Profiles. For example, you can create a multilevel model with QoS rules that define different settings for bandwidth. Make new profiles on the basis of these rules; using these profiles will adjust the bandwidth usage within the logical network for each virtual machine.

Authentication via an external directory service

After oVirt is up and running, there is a default authentication domain that stores the internal administrator account. But this domain cannot be used to create new accounts. To add new users to the oVirt environment, oVirt Engine has to connect to the directory service. oVirt supports user authentication through the IPA or Active Directory. For enabling and using external directory services, we need to use the `engine-manage-domains` command-line utility. This utility can manage domains provided by these directory services. oVirt allows the use of multiple domains.

It should be noted that the `engine-manage-domains` utility, in the process of connecting, performs checking of the availability of certain DNS SRV records. Therefore, before using an authentication domain, SRV and PTR records should be added in the DNS zone that points to the directory server. Note that the Active Directory and IPA have built-in DNS servers, which already have these records. In case a dedicated DNS server is used, we need to create records manually.

As a concrete example, consider the DNS domain named `example.com`, which is the name of the directory server `dc.example.com` (`10.202.0.20`), and assume that the directory server is set up to Kerberos REALM named `EXAMPLE.COM`. Record the `A` type that we already have, it remains to add SRV records `_ldap._tcp` and `_kerberos._tcp`, as shown in the following snippet:

```
dc                    A         10.202.0.20
_ldap._tcp          SRV     0 0 389 dc.example.com.
_kerberos._tcp      SRV     0 0 88  dc.example.com.
```

The similar records should be added in your DNS. Also, in the directory server there must be a correctly configured PTR record.

After a record is created, we can connect our domain to the oVirt. To do this, connect via SSH to the oVirt Engine server and connect the domain. The following command shows how to add IPA domain with IPA admin account:

```
# engine-manage-domains -action=add -domain=example.com -provider=ipa
-user=admin -interactive
```

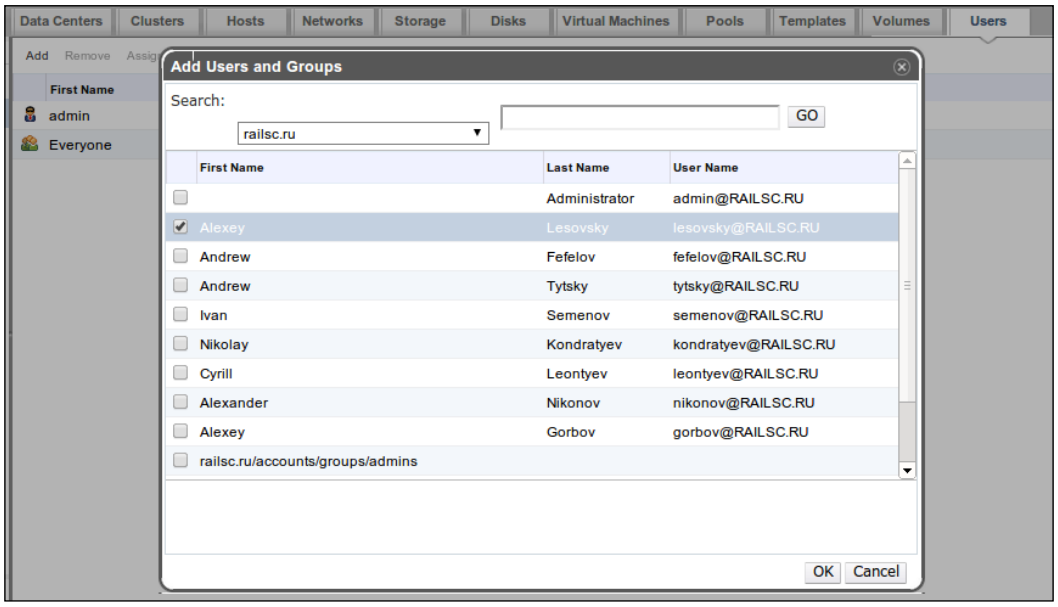
When oVirt Engine establishes a connection to the directory server, the user will be prompted for a password confirming that the user is connecting. In our case, this is admin. After entering the password, make sure the connections for the domain and the operation are complete; restart the oVirt engine service domain with `service`:

```
# service ovirt-engine restart
```

Now, when a domain is added, we can add new users to oVirt:

1. Go into the administrator portal, and in **Tree Panel** select the **System** root section.
2. Open the **Users** tab in the resource pane and in the toolbar click on the **Add** button. In the dialog box **Add Users and Group**, indicate your choice of authentication domain and click on **GO**.
3. After clicking on **GO**, the screen will display a list of users that are available to be added. Select the user and click on **OK**. Now, the added user will be displayed in the user list.

The following screenshot shows the dialog for adding a user:

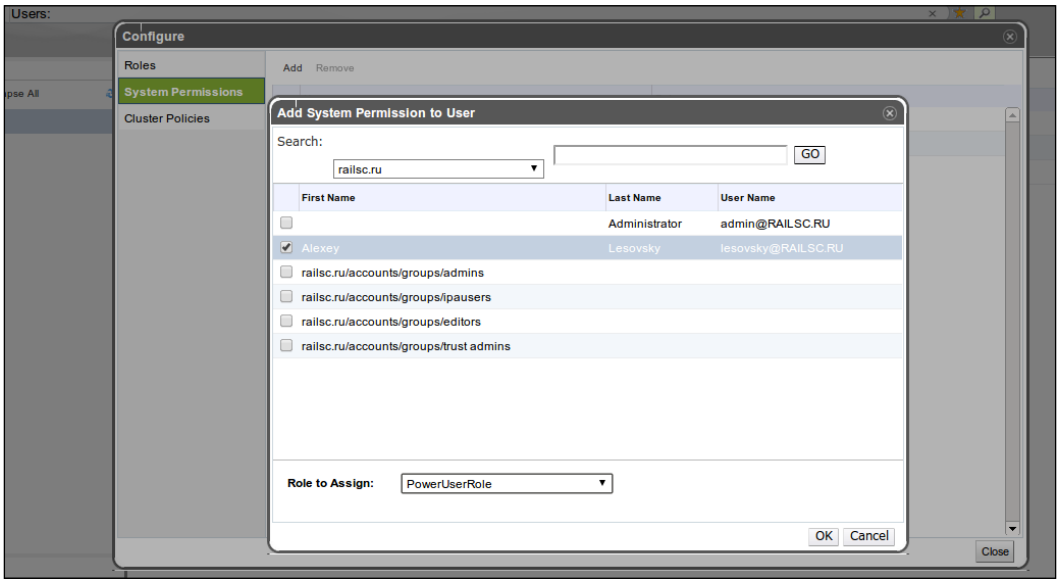


Now, when a user is added, we can assign him a system role that will allow him to perform various operations in oVirt, such as create virtual machines, create disks, maintain hosts, clusters, or data centers, and so on. To do this, perform the following steps:

1. Click on the **Configure** button in **Header**. In the **Configure** window, select the **Configure System Permissions** section in the left-hand side pane. Click on the **Add** button in the toolbar.
2. In an additional window **Add System Permission to User**, choose a domain and click on the **GO** button to search for available users.
3. Select the previously added user and at the bottom of the screen define its role. For example, **PowerUserRole**.
4. Click on **OK** and the new role will be displayed in the list of roles. Click on **Close** and exit the **Configure** window.

Now, when a user associates with a specific role, this user can perform operations that are allowed for its role.

The following screenshot shows the dialog for defining the role:



Quota

Quotas are a mechanism which control resource allocation within the data centers. With quotas, there can be controlled allocation of resources to users and groups. Quotas can be set for resources such as CPU, RAM, and disk space. Within the data center, you can create a few quotas and assign those quotas to specific users. Thus, the limitation of resources is realized in oVirt.

Quotas are designed primarily to limit the users, so it is advisable to use the quota if you have an external directory server.

Before you begin, perform the following steps to configure the data center on the use of quotas:

1. Go to the **Data Centers** tab, select the data center from the list, and click on the **Edit** button.
2. In the **Edit Data Center** dialog box within **Quota Mode**, select the type of quota:
    - **Audit**: This allows you to set quotas and produce warning messages, but allows an excess of limits

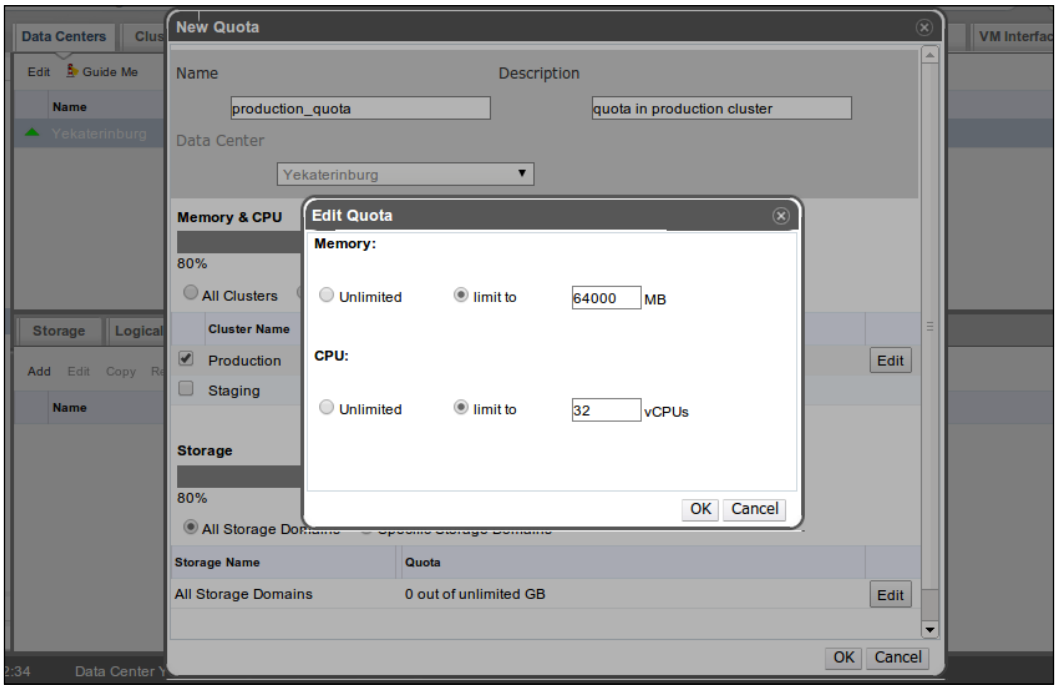
- **Enforced**: This allows you to set quotas and apply them; any attempt to breach the quota allocation of resources will be stopped
- **Disabled**: This is used to disable the use of quotas in the data center

3. Select **Enforced** and click on the **OK** button to activate quotas.

After turning on quotas, you can go to the direct creation of quotas within the data center:

1. Go to the **Data Centers** tab and select that data center where quotas have been enabled.
2. Select the **Quota** tab in the details pane and click on the **Add** button.
3. In the **New Quota** window, define the settings for the new quota as follows:
    - **Name**: In this, you should specify a name for the quota.
    - **Description**: Here you can enter a short description.
    - **Data Center**: This field is already filled in and the name is displayed in the data center which is determined by the quota.
    - **Memory & CPU**: In this section we define quota for CPU and memory usage.
    - **All Clusters**: Enable this option if you want to associate the quota with all the clusters in the data center.
    - **Specific Clusters**: Enable this option if you want to configure the quota for the individual clusters. Enabling this option will be available in the list and should be noted by the necessary clusters.
4. To determine the quotas associated with the CPU and memory, click on the **Edit** button. In the **Edit Quota** dialog box, specify a limit to include the parameter and set the limits for the use of CPU and memory. After specifying the values, click on **OK** and go back to the previous dialog.

The following screenshot shows the dialog box **Edit Quota**:

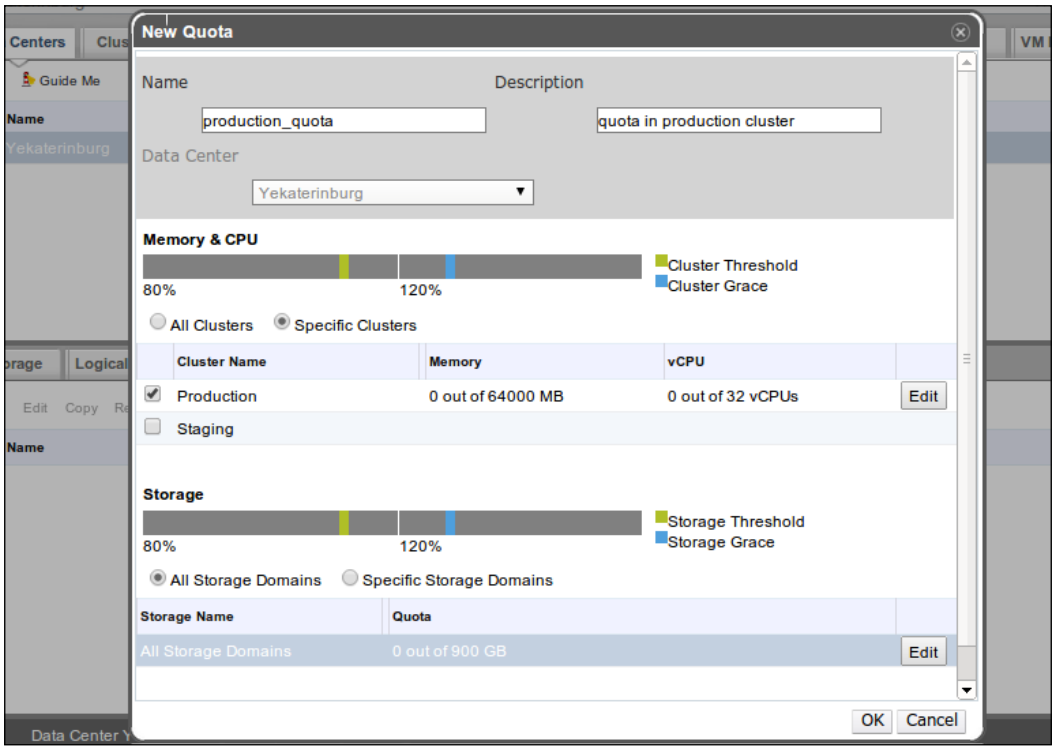


The **Storage** section defines the quotas related to the allocation of disk resources. Fill in the details:

- **All Storage Domains**: Enable this option if you want to associate quota with all storages connected to datacenter.
- **Specific Storage Domains**: Enable this option if you want to configure the quota for individual storage domains. Enabling this option will be available in the list and should be noted by the necessary storage.

5. To determine the quotas associated with the storage click on the **Edit** button. In the **Edit Quota** dialog box, enter the limit to include the parameter and set the limits for the use of storage. After specifying the values click on **OK** and go back to the previous dialog.
6. After specifying the required values, click on the **OK** button and then the quota will be created.

The following screenshot shows the window **New Quota**:



When the quota is determined it can be associated with a specific user:

1. Click on the **Data Centers** section in **Tree Pane** and choose the data center in which there was a quota.
2. Go to the **Quota** tab in resource pane and select the quota from the list.
3. Next select the **Consumers** tab in the details pane and click on **Add**.
4. In the **Assign User / Group to Quota** dialog box, select the authentication domain and click on the **OK** button to display the available users.
5. Select users who will be assigned to a quota and click on **OK**.

The users will be displayed in the list of consumers; this means that now the users are limited to a quota. If the quota is exceeded, the users will get a message about the impossibility of the use of resources.

Summary

In this chapter, we learned about oVirt's advanced features. It is quite possible that we do not need these functions nevertheless, they can be useful. These features allow you to flexibly configure and use oVirt. Some features such as live migration and high availability are integral to represent the whole of modern virtualization management platform. At the same time, oVirt provides functions such as power management and quotas, which allow more efficient use of available resources. Snapshots and pools of prestarted VMs are useful functions that you can use to quickly and effectively install and run virtual machines. oVirt is an intensive developed product, so we can say with certainty that in the future there will be new features that will make oVirt even more attractive. In the next chapter, we will look at how to troubleshoot problems, what could be the problem, and what approaches should be used to solve them.

6
oVirt Troubleshooting

oVirt is a very complex mechanism that consists of multiple components. In case of problems, it is very difficult to keep track of the specific level of the problem. Most problems can be divided into three categories:

- General problems or problems of interaction between components
- Problems associated with oVirt Engine
- Problems associated with virtualization hosts

Let us try to examine these categories and develop common approaches to solve such problems.

Common problems

Common problems are characterized by the fact that the problem affects multiple components of the infrastructure. Such problems are quite serious as they can damage any part of the infrastructure. In most cases, the Administrator Portal provides a more or less clear idea of the level at which the problem occurred. oVirt Engine constantly monitors the status of the attached storages and virtualization hosts. In the case that there are any problems with storages, virtualization hosts, or other components, it will be immediately displayed in the event pane. Having obtained the information about the error, we can begin troubleshooting.

General examples of such problems may include:

- **Loss of communication between the SPM host and storage**: In this case, you can see errors in `vdsm.log`. The following example will show a case with an iSCSI storage failure:

  ```
  WARNING: lvm vgs failed: 5 [] ['
  ```

```
  /dev/mapper/1IET_00010001: read failed after 0 of 4096 at
38650445824: Input/output error', '
  /dev/mapper/1IET_00010001: read failed after 0 of 4096 at
38650503168: Input/output error', '
  /dev/mapper/1IET_00010001: read failed after 0 of 4096 at 0:
Input/output error', '
  WARNING: Error counts reached a limit of 3. Device /dev/
mapper/1IET_00010001 was disabled', '
  Volume group "b2464090-45b2-40b2-b2e9-5677e8d855dc" not found' ]
```

- **Loss of communication between the oVirt Engine and a VDSM agent on a particular host**: If this occurs, the following messages in `engine.log` will be displayed:

```
ERROR Command GetCapabilitiesVDS execution failed. Exception:
VDSNetworkException: java.net.ConnectException: Connection refused
INFO  Server failed to respond, vds_id = 9de867d3-81cc-4530-
812c-35eda6a1a861, vds_name = node02, vm_count = 0, spm_status =
None, non-responsive_timeout (seconds) = 60, error = java.net.
ConnectException: Connection refused
WARN  Failed to refresh VDS , vds = 9de867d3-81cc-4530-812c-
35eda6a1a861 : node02, VDS Network Error, continuing.
```

  If this type of loss occurs, the algorithm is similar to the following steps:

  1. You need to ensure that the problem host or storage is running and responsible through the network.
  2. If connection to the virtualization host fails, the first thing to check is the VDSM service status by the command `service vdsmd status` (in CentOS/RHEL). Appropriate processes should be observed in the process list. If you find that any of the services are not running, start them. If you cannot start the services, you should focus on the system logs.
  3. If the problem is with the storage, make sure that the storage is available and ready for use. To check this, you can use ready-made tools. For example, to check the NFS storage utility, you can use `nfs-check.py` that comes with the source oVirt. For other types of storage, you can use its own utils to check for availability. For example, use the `iscsiadm` command line utility to check iSCSI storages. For example, `iscsiadm –m discovery –t sendtargets –p <storage_ip>`.

Security-related problems

Often, the cause of the problem may be an incorrectly configured packet filter, such as `firewalld` or `iptables`. Usually, these errors are accompanied by messages with timeouts or refused connections:

```
ERROR Command ListVDS execution failed. Exception: VDSNetworkException:
java.net.SocketTimeoutException: connect timed out
WARN Failed to refresh VDS, node02, VDS Network Error, continuing.
ERROR Command GetCapabilitiesVDS execution failed. Exception:
VDSNetworkException: java.net.SocketTimeoutException: connect timed out
```

If all the services are running but do not respond over the network, try stopping `firewalld`/`iptables` as a temporary measure. And then check if the problem persists. If after turning off packet filtering, the problem has ceased to manifest itself, it should be given a proper setup packet filter.

Also, you need to take SELinux into account. Developers and maintainers put in a lot of effort to support SELinux and to provide appropriate working of programs in the SELinux environment. But keep in mind that it is a tool that restricts access, and insufficient configuration can restrict access to resources. As a workaround, you can change the SELinux mode to permissive with the `setenforce` command-line utility. If SELinux does create problems in oVirt, you must reconfigure SELinux.

Problems with oVirt Engine

oVirt Engine problems usually occur during the setup phase and the preconfiguration phase. However, if you do not use nightly builds you can avoid serious problems. Before every release, oVirt is very thoroughly tested to avoid errors that crop up during use. But even after the release, bugs can be found, which are corrected in the later releases. You may observe the following problems:

- **Engine setup failure due to wrongly configured (or not configured) DNS**: This is a cautionary prerequisite for installing oVirt. Check the DNS forward and reverse lookup match using the command-line utility `nslookup` on Windows or the utility `nslookup/dig` on Linux. Additional information, explanation, and usage examples can be found on wiki pages `http://en.wikipedia.org/wiki/Dig_(command)` and `http://en.wikipedia.org/wiki/Nslookup`.

- **The problem of starting the services**: oVirt Engine is implemented as a set of services. Therefore, for it to work properly, services must be started, that is, at least services such as `httpd`, `postgresql`, and `ovirt-engine`. You need to monitor their condition and proper operation. In the event of a service crash, error information can be found in the system logs and appropriate action must be taken accordingly.
- **The problem of the misconfigured firewall**: When configuring oVirt by engine setup, the installer prompts you to import the settings for a firewall so if you are using a packet filter, it is a good choice to import these rules. However, the proposed configuration is generalized in the case of nonstandard settings and network environments for packet filter rules to be edited independently.
- **High Availability not working**: VM is marked as highly available but it isn't started or migrated automatically in case virtualization hosts fail. When such problems occur, we need to check the power management settings. In most cases, this means that the power management isn't configured or has been configured incorrectly. Power management is required for the High Availability feature.
- **VM's shutdown from oVirt Engine doesn't work**: This occurs when oVirt guest agent package isn't installed/running in VM. Instead of oVirt guest agent, you can set up and start the `acpid` service, which allows correct shutdown from oVirt Engine to be performed.

If you want to reset oVirt Engine, use the `engine-cleanup` utility before running the engine setup again.

Problems with virtualization hosts

Even in the event of failure of the virtualization host, oVirt can continue working as usual. Problems with the virtualization hosts are usually related with the VDSM agent or with cases where the host becomes unavailable. The search algorithm is similar to the general troubleshooting and problem-solving algorithm with a small addition of steps as follows:

1. Perform a general check of the operating system (such as CPU load, free space or inodes, network utilization, and connectivity, memory, and swap usage).
2. Check the hardware status.
3. Check the running VDSM service.
4. Check the VDSM service logs for errors and warnings.

Also, one of the most common problems is disabled hardware virtualization in BIOS. This subject will be covered in the *Virtualization Host Requirements* section in *Chapter 1*, *Before you Begin*. In this case, the host must be rebooted and the hardware virtualization support must be enabled in BIOS. When you are trying to add a host without enabled hardware virtualization support, you can see messages such as the following in `engine.log`:

```
ERROR Installation ovirt-node02.example.com: Failed to execute stage
'Setup validation': Hardware does not support virtualization
```

Another example in the list of the most common problems related to hardware equipment is the CPU family mismatch. This problem occurs in the case that a virtualization host is added into a cluster with no CPU family. For example, we have a cluster configured with the Intel Haswell family. Now, try to add a host with an old Conroe family into that cluster. This attempt will display the following message at the end in `engine.log`:

```
INFO  Message: Host node02 moved to Non-Operational state as host does
not meet the cluster's minimum CPU level. Missing CPU features : model_
Haswell
```

To avoid this problem, add the nodes to the clusters with corresponding CPU families.

Problems with storages

Storage is usually an external component, so you should make sure that the storage is successfully connected and available over the network. To do this, you must try manually mounting the storage to a temporary directory. For example, you can use the mount utility for NFS storages and for the iSCSI storage, `iscsiadm`.

Also, the usual problem with storages is hanging NFS mounts. This can happen when the ISO domain is located on the same host with oVirt Engine, and the NFS mount can hang when the host with the ISO domain and oVirt Engine is rebooted. In some cases, virtualization hosts can't reactivate the mount. A possible solution to this is to reboot the virtualization host causing the problem.

Logs

Logs are the most valuable sources of information, so logs are the key to solving most of the problems. Each component of oVirt usually writes its own log, which greatly facilitates the search for the problem's cause. oVirt or VDSM logs are placed in the default directory for the logs, `/var/log`.

oVirt Engine logs

oVirt Engine logs are stored on the host on which oVirt Engine runs:

- `ovirt-engine/boot.log`, `ovirt-engine/console.log`, and `ovirt-engine/server.log`: These logs are used to display the messages associated with the Jboss Application Service
- `ovirt-engine/engine.log`: This log is used by the oVirt Engine service and is one of the most important logfiles
- `ovirt-engine/engine-manage-domains.log`: This log is used to log messages related to the management of authentication domains
- `ovirt-engine/host-deploy/`: This log contains logs of operations performed when adding a new virtualization host in oVirt
- `ovirt-engine/setup/`:This log contains logs of the operations performed during engine setup

VDSM logs

The VDSM agent logs are placed in the default directory `/var/log` on each virtualization node, and they reflect the work of the VDSM agents.

- `vdsm/vdsm.log`: This is the main log of the VDSM agent
- `vdsm/spm-lock.log`: This log works in cases where the host is running as an SPM

The names of logfiles in different versions of oVirt may vary, but always keep in mind that logs are very good sources of information for solving problems.

Monitoring

In conclusion, I would like to say a few words about the monitoring systems. While configuring the virtualization infrastructure, considerable attention should be paid to the monitoring infrastructure. Monitoring systems are aware of what is happening within the infrastructure and respond quickly in the case there is an emergency. A good choice might be a monitoring system, such as ZABBIX or Nagios. These monitoring systems are flexible enough to set up and are very functional. These monitoring systems are well documented and have a very large community that allows you to quickly learn and start using them. Note that in the case of Nagios, monitoring is an available plugin, which can be integrated into oVirt. Information about this plugin and others can be found at `http://www.ovirt.org/Features/UIPlugins` in the section **oVirt Monitoring UI Plugin**.

You can monitor the status of hardware servers and running services. The oVirt infrastructure is placed on a set of physical servers, which can accordingly be subject to the classic problems associated with the hardware. It may be the failure of some components of the server or an excessive load. If you detect such faults, they should be dealt with as usual. When faulty hardware is replaced, the excess load is redistributed.

Summary

This is the final chapter; we looked at the potential sources of failures inside oVirt. At first glance, oVirt seems a complicated mechanism; so if a problem occurs, it is important to know in which direction to look for a solution. Here we have considered the main ways to find solutions that can solve most of the problems. However, to successfully address any problems that may arise, it is necessary to know what's inside oVirt and what components it consists of. In this case, finding a solution is fairly simple. Always try to look behind the screen. Good luck!

A
NFS Storage Setup on CentOS

oVirt supports NFS storage. This appendix will cover how to set up the NFS server with CentOS Linux.

Preparing the local storage

Local disk running **Logical Volume Manager** (**LVM**) is used as storage. LVM is a flexible and powerful disk manager. For additional information about LVM, please look at the following link: `https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/Logical_Volume_Manager_Administration/index.html`.

To create LVM volume, we need a whole disk. For a disk device, only the partition table must be erased, which will effectively destroy information about partitions on that disk. In the following command, we will focus on the first sector (be careful, this command will erase the partition table):

```
# dd if=/dev/zero of=PhysicalVolume bs=512 count=1
```

`PhysicalVolume` must be replaced by target disk device.

Now we can initialize a physical volume and a volume group. Logical volume will be created in the storage volume group. For logical volume, which is also named `storage`, we take 90 percent of the free space from the physical volume:

```
# pvcreate /dev/sdb
# vgcreate storage /dev/sdb
# lvcreate -l 90%FREE -n storage/storage
```

Now we can format the logical volume to the desired filesystem. For example, `ext4`. When formatting, we have the option to add labels to filesystems (`-L label_name`). After formatting, logical volume should create a mount point, for example `/mnt/data`, and mount the created filesystem.

```
# mkfs.ext4 -q -j -L nfs /dev/storage/storage
# mkdir -p /srv/data
# mount LABEL=nfs /srv/data
```

Also, we should save the new mount point configuration at `/etc/fstab`. This ensures that after the reboot, the filesystem will be mounted automatically.

```
# vi /etc/fstab
LABEL=nfs               /srv/data               ext4    defaults        1 1
```

Now we can install the packages and set up NFS.

Package installation and NFS setup

We need to install the `nfs-utils` package and configure the export of a previously created filesystem. Installation will be performed through the `yum`—CentOS/RHEL package manager:

```
# yum install -y nfs-utils
```

Now configure exports, edit `/etc/exports`, and add the following lines:

```
# vi /etc/exports
/srv/data       10.101.0.0/16(rw,sync,no_root_squash,no_subtree_check)
```

The next step is to set the ownership on the exported directory and edit SELinux settings:

```
# chown vdsm:kvm /srv/data
# setsebool -P virt_use_nfs 1
```

After that, we can start services and set them to autostart:

```
# service rpcbind restart && chkconfig rpcbind on
# service nfs restart && chkconfig nfs on
```

Check exports; in the output we must see our export directory:

```
# showmount -e
Export list for ovirt-node04.example.com:
/srv/data *
```

The installation is complete. NFS storage is ready to connect to oVirt.

B
iSCSI Storage Setup on CentOS

oVirt Engine supports iSCSI storage. This appendix will cover how to set up an iSCSI server with CentOS Linux.

Preparing the local storage

Local disk running with **Logical Volume Manager** (**LVM**) is used as the storage. LVM is a flexible and powerful disk manager. For additional info about LVM, please refer to the link `https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/Logical_Volume_Manager_Administration/index.html`.

To create LVM volume, we need a whole disk. For a disk device, only the partition table must be erased, which will effectively destroy information about partitions on that disk. In the following command, we will focus on the first sector (be careful, this command will erase the partition table):

```
# dd if=/dev/zero of=PhysicalVolume bs=512 count=1
```

`PhysicalVolume` must be replaced by target disk device.

Now we can initialize a physical volume and a volume group. Logical volume will be created in the storage volume group. For logical volume, which is also named as `storage`, we take 90 percent of free space from the physical volume:

```
# pvcreate /dev/sdb
# vgcreate storage /dev/sdb
# lvcreate -l 90%FREE -n storage/storage
```

Formatting the logical volume to filesystem is not required and we can proceed to the package installation.

Package installation and iSCSI setup

We need to install the `scsi-target-utils` package and configure the iSCSI target on the previously created logical volume. Installation will be performed through the `yum`—`CentOS/RHEL` package manager:

```
# yum install -y scsi-target-utils
```

After installation, configure the targets. At the end of `targets.conf`, add the `<target>-</target>` section and set the targets in same way with following naming rule: `[iqn.yaer-month.domain:any-name]`. In the initiator option, we should specify the client IP network:

```
# vi /etc/tgt/targets.conf
<target iqn.2013-10.com.example:iscsi-target0>
        backing-store /dev/storage/storage
        initiator-address 10.101.15.0/24
        incominguser ovirtuser ovirtpassword
</target>
```

Now, we can start the `tgtd` service and set the service to autostart:

```
# service tgtd start && chkconfig tgtd on
```

Check the created target using the following command. In the output, we must see **State: ready** and in LUN information, **Online: yes**. This state ensures that the iSCSI LUN is running and ready for use:

```
# tgtadm --mode target --op show
```

The installation is complete. iSCSI storage is ready to connect to oVirt.

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Thank you for buying

Getting Started with oVirt 3.3

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.

About Packt Open Source

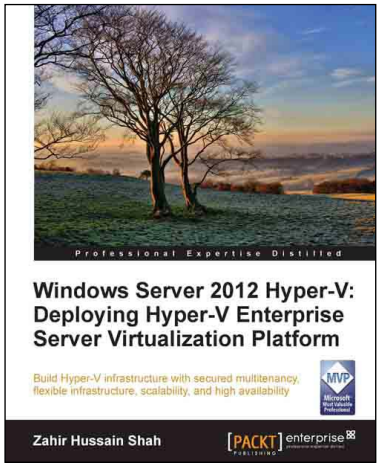
In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

[PACKT] open source
community experience distilled
PUBLISHING

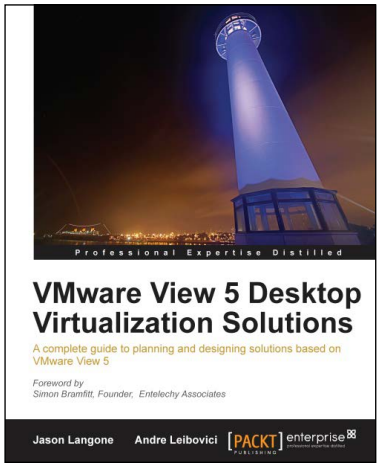


Windows Server 2012 Hyper-V: Deploying Hyper-V Enterprise Server Virtualization Platform

ISBN: 978-1-849688-34-5 Paperback: 410 pages

Build Hyper-V infrastructure with secured multitenancy, flexible infrasructure, scalability, and high availability

1. A complete step-by-step Hyper-V deployment guide, covering all Hyper-V features for configuration and management best practices
2. Understand multi-tenancy, flexible architecture, scalability, and high availability features of new Windows Server 2012 Hyper-V



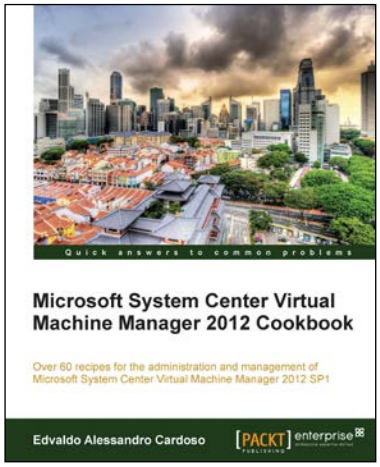
VMware View 5 Desktop Virtualization Solutions

ISBN: 978-1-849681-12-4 Paperback: 288 pages

A complete guide to planning and designing solutions based on VMware View 5

1. Written by VMware experts Jason Langone and Andre Leibovici, this book is a complete guide to planning and designing a solution based on VMware View 5
2. Secure your Visual Desktop Infrastructure (VDI) by having firewalls, antivirus, virtual enclaves, USB redirection and filtering and smart card authentication
3. Analyze the strategies and techniques used to migrate a user population from a physical desktop environment to a virtual desktop solution

[PACKT] open source
community experience distilled
PUBLISHING



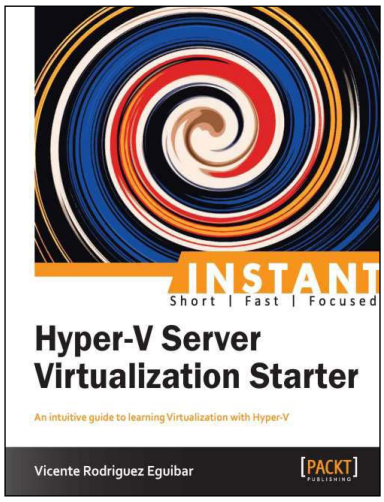
Microsoft System Center Virtual Machine Manager 2012 Cookbook

ISBN: 978-1-849686-32-7 Paperback: 342 pages

Master Spring's well-designed web frameworks to develop powerful web applications

1. Create, deploy, and manage Datacentres, Private and Hybrid Clouds with hybrid hypervisors by using VMM 2012 SP1, App Controller, and Operations Manager
2. Integrate and manage fabric (compute, storages, gateways, networking) services and resources. Deploy Clusters from bare metal servers

---



Instant Hyper-V Server Virtualization Starter

ISBN: 978-1-782179-97-9 Paperback: 58 pages

Building rigorously tested and bug-free Django applications

1. Learn something new in an Instant! A short, fast, focused guide delivering immediate results
2. Step-by-step, practical guide to understanding and implementing virtualization for an enterprise environment
3. Learn how to create a virtual machine in three steps