Proxmox VE Administration Guide

Release 4.3

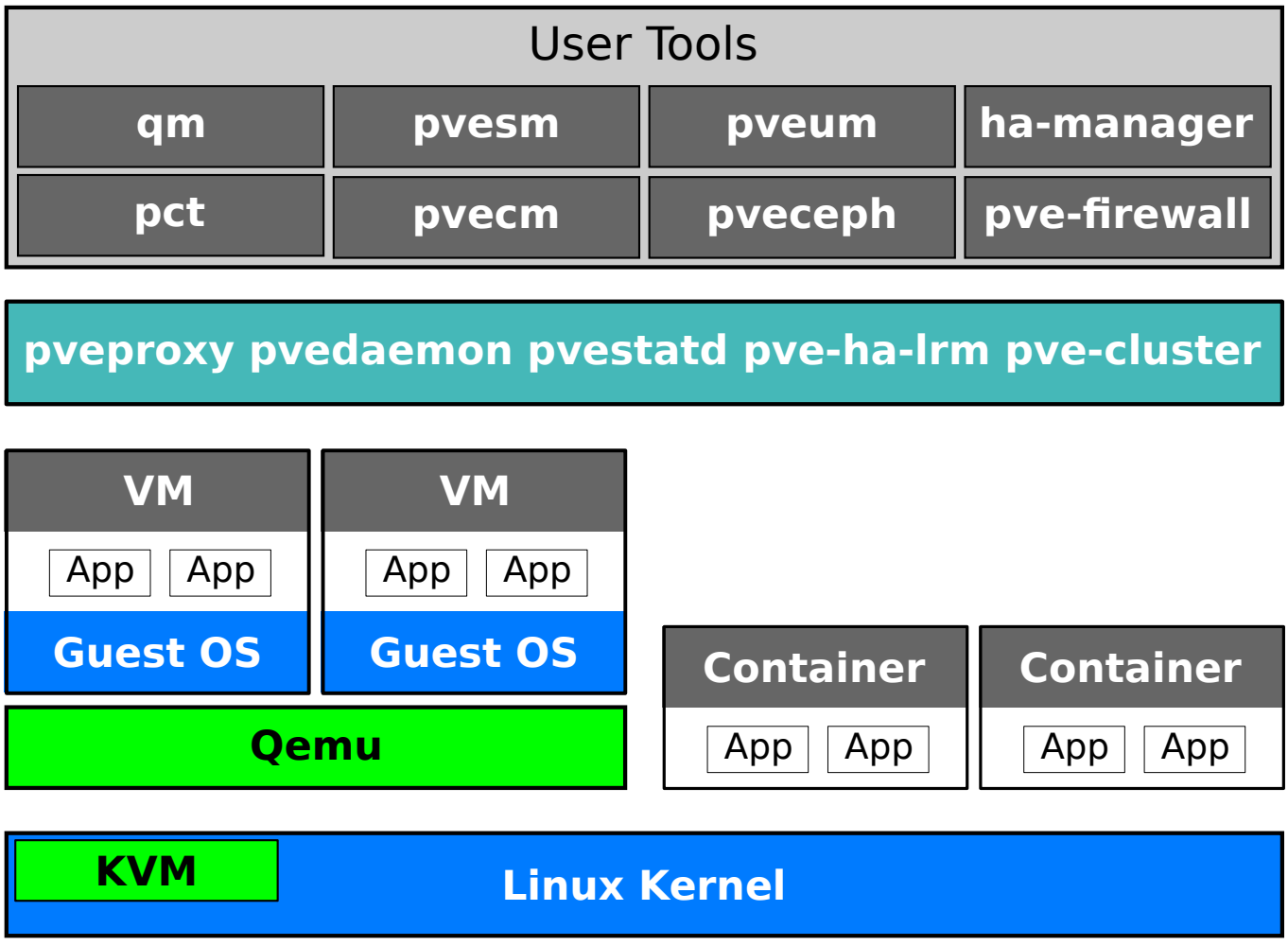
Contents

Chapter 1

Introduction

Proxmox VE is a platform to run virtual machines and containers. It is based on Debian Linux, and completely open source. For maximum flexibility, we implemented two virtualization technologies - Kernel-based Virtual Machine (KVM) and container-based virtualization (LXC).

One main design goal was to make administration as easy as possible. You can use Proxmox VE on a single node, or assemble a cluster of many nodes. All management task can be done using our web-based management interface, and even a novice user can setup and install Proxmox VE within minutes.



1.1 Central Management

While many people start with a single node, Proxmox VE can scale out to a large set of clustered nodes. The cluster stack is fully integrated and ships with the default installation.

Unique Multi-master Design

The integrated web-based management interface gives you a clean overview of all your KVM guests and Linux containers and even of your whole cluster. You can easily manage your VMs and containers, storage or cluster from the GUI. There is no need to install a separate, complex, and pricy management server.

Proxmox Cluster File System (pmxcfs)

Proxmox VE uses the unique Proxmox Cluster file system (pmxcfs), a database-driven file system for storing configuration files. This enables you to store the configuration of thousands of virtual machines. By using corosync, these files are replicated in real time on all cluster nodes. The file system stores all data inside a persistent database on disk, nonetheless, a copy of the data resides in RAM which provides a maximum storage size is 30MB - more than enough for thousands of VMs.

Proxmox VE is the only virtualization platform using this unique cluster file system.

Web-based Management Interface

Proxmox VE is simple to use. Management tasks can be done via the included web based managment interface - there is no need to install a separate management tool or any additional management node with huge databases. The multi-master tool allows you to manage your whole cluster from any node of your cluster. The central web-based management - based on the JavaScript Framework (ExtJS) - empowers you to control all functionalities from the GUI and overview history and syslogs of each single node. This includes running backup or restore jobs, live-migration or HA triggered activities.

Command Line

For advanced users who are used to the comfort of the Unix shell or Windows Powershell, Proxmox VE provides a command line interface to manage all the components of your virtual environment. This command line interface has intelligent tab completion and full documentation in the form of UNIX man pages.

REST API

Proxmox VE uses a RESTful API. We choose JSON as primary data format, and the whole API is formally defined using JSON Schema. This enables fast and easy integration for third party management tools like custom hosting environments.

Role-based Administration

You can define granular access for all objects (like VM´s, storages, nodes, etc.) by using the role based user- and permission management. This allows you to define privileges and helps you to control access to objects. This concept is also known as access control lists: Each permission specifies a subject (a user or group) and a role (set of privileges) on a specific path.

Authentication Realms

Proxmox VE supports multiple authentication sources like Microsoft Active Directory, LDAP, Linux PAM standard authentication or the built-in Proxmox VE authentication server.

1.2 Flexible Storage

The Proxmox VE storage model is very flexible. Virtual machine images can either be stored on one or several local storages or on shared storage like NFS and on SAN. There are no limits, you may configure as many storage definitions as you like. You can use all storage technologies available for Debian Linux.

One major benefit of storing VMs on shared storage is the ability to live-migrate running machines without any downtime, as all nodes in the cluster have direct access to VM disk images.

We currently support the following Network storage types:

- LVM Group (network backing with iSCSI targets)
- iSCSI target
- NFS Share
- Ceph RBD
- Directly use iSCSI LUNs
- GlusterFS

Local storage types supported are:

- LVM Group (local backing devices like block devices, FC devices, DRBD, etc.)
- Directory (storage on existing filesystem)
- ZFS

1.3 Integrated Backup and Restore

The integrated backup tool (vzdump) creates consistent snapshots of running Containers and KVM guests. It basically creates an archive of the VM or CT data which includes the VM/CT configuration files.

KVM live backup works for all storage types including VM images on NFS, iSCSI LUN, Ceph RBD or Sheepdog. The new backup format is optimized for storing VM backups fast and effective (sparse files, out of order data, minimized I/O).

1.4 High Availability Cluster

A multi-node Proxmox VE HA Cluster enables the definition of highly available virtual servers. The Proxmox VE HA Cluster is based on proven Linux HA technologies, providing stable and reliable HA services.

1.5 Flexible Networking

Proxmox VE uses a bridged networking model. All VMs can share one bridge as if virtual network cables from each guest were all plugged into the same switch. For connecting VMs to the outside world, bridges are attached to physical network cards assigned a TCP/IP configuration.

For further flexibility, VLANs (IEEE 802.1q) and network bonding/aggregation are possible. In this way it is possible to build complex, flexible virtual networks for the Proxmox VE hosts, leveraging the full power of the Linux network stack.

1.6 Integrated Firewall

The intergrated firewall allows you to filter network packets on any VM or Container interface. Common sets of firewall rules can be grouped into *security groups*.

1.7 Why Open Source

Proxmox VE uses a Linux kernel and is based on the Debian GNU/Linux Distribution. The source code of Proxmox VE is released under the GNU Affero General Public License, version 3. This means that you are free to inspect the source code at any time or contribute to the project yourself.

At Proxmox we are committed to use open source software whenever possible. Using open source software guarantees full access to all functionalities - as well as high security and reliability. We think that everybody should have the right to access the source code of a software to run it, build on it, or submit changes back to the project. Everybody is encouraged to contribute while Proxmox ensures the product always meets professional quality criteria.

Open source software also helps to keep your costs low and makes your core infrastructure independent from a single vendor.

1.8 Your benefit with Proxmox VE

- Open source software
- No vendor lock-in
- Linux kernel
- Fast installation and easy-to-use
- Web-based management interface
- REST API
- Huge active community
- Low administration costs and simple deployment

1.9 Project History

The project started in 2007, followed by a first stable version in 2008. By that time we used OpenVZ for containers, and KVM for virtual machines. The clustering features were limited, and the user interface was simple (server generated web page).

But we quickly developed new features using the Corosync cluster stack, and the introduction of the new Proxmox cluster file system (pmxcfs) was a big step forward, because it completely hides the cluster complexity from the user. Managing a cluster of 16 nodes is as simple as managing a single node.

We also introduced a new REST API, with a complete declarative spezification written in JSON-Schema. This enabled other people to integrate Proxmox VE into their infrastructur, and made it easy provide additional services.

Also, the new REST API made it possible to replace the original user interface with a modern HTML5 application using JavaScript. We also replaced the old Java based VNC console code with noVNC. So you only need a web browser to manage your VMs.

The support for various storage types is another big task. Notably, Proxmox VE was the first distribution to ship ZFS on Linux by default in 2014. Another milestone was the ability to run and manage Ceph storage on the hypervisor nodes. Such setups are extremely cost effective.

When we started we were among the first companies providing commercial support for KVM. The KVM project itself continuously evolved, and is now a widely used hypervisor. New features arrives with each release. We developed the KVM live backup feature, which makes it possible to create snapshot backups on any storage type.

The most notable change with version 4.0 was the move from OpenVZ to LXC. Containers are now deeply integrated, and they can use the same storage and network features as virtual machines.

Chapter 2

Host System Administration

Proxmox VE is based on the famous Debian Linux distribution. That means that you have access to the whole world of Debian packages, and the base system is well documented. The Debian Administrator's Handbook is available online, and provides a comprehensive introduction to the Debian operating system (see [Hertzog13]).

A standard Proxmox VE installation uses the default repositories from Debian, so you get bug fixes and security updates through that channel. In addition, we provide our own package repository to roll out all Proxmox VE related packages. This includes updates to some Debian packages when necessary.

We also deliver a specially optimized Linux kernel, where we enable all required virtualization and container features. That kernel includes drivers for ZFS, and several hardware drivers. For example, we ship Intel network card drivers to support their newest hardware.

The following sections will concentrate on virtualization related topics. They either explains things which are different on Proxmox VE, or tasks which are commonly used on Proxmox VE. For other topics, please refer to the standard Debian documentation.

2.1 System requirements

For production servers, high quality server equipment is needed. Keep in mind, if you run 10 Virtual Servers on one machine and you then experience a hardware failure, 10 services are lost. Proxmox VE supports clustering, this means that multiple Proxmox VE installations can be centrally managed thanks to the included cluster functionality.

Proxmox VE can use local storage (DAS), SAN, NAS and also distributed storage (Ceph RBD). For details see chapter storage Chapter 5.

2.1.1 Minimum requirements, for evaluation

- CPU: 64bit (Intel EMT64 or AMD64)
- RAM: 1 GB RAM
- Hard drive

- One NIC

2.1.2 Recommended system requirements

- CPU: 64bit (Intel EMT64 or AMD64), Multi core CPU recommended
- RAM: 8 GB is good, more is better
- Hardware RAID with batteries protected write cache (BBU) or flash based protection
- Fast hard drives, best results with 15k rpm SAS, Raid10
- At least two NIC´s, depending on the used storage technology you need more

2.2 Getting Help

2.2.1 Proxmox VE Wiki

The primary source of information is the Proxmox VE wiki. It combines the reference documentaion with user contributed content.

2.2.2 Community Support Forum

Proxmox VE itself is fully open source, so we always encourage our users to discuss and share their knowledge using the Community Support Forum. The forum is fully moderated by the Proxmox support team, and has a quite large user base around the whole world. Needless to say that such a large forum is a great place to get information.

2.2.3 Mailing Lists

This is a fast way to communicate via email with the Proxmox VE community

- Mailing list for users: PVE User List

The primary communication channel for developers is:

- Mailing list for developer: PVE development discussion

2.2.4 Commercial Support

Proxmox Server Solutions Gmbh also offers a commercial support channel. Proxmox VE server subscriptions can be ordered online, see Proxmox VE Shop. For all details see Proxmox VE Subscription Service Plans. Please contact the Proxmox sales team for commercial support requests or volume discounts.

2.2.5 Bug Tracker

We also run a public a public bug tracker at https://bugzilla.proxmox.com. If you ever detect a bug, you can file an bug entry there. This makes it easy to track the bug status, and you will get notified as soon as the bug is fixed.

2.3 Package Repositories

All Debian based systems use APT as package management tool. The list of repositories is defined in */etc/apt/sources.list* and *.list* files found inside */etc/apt/sources.d/*. Updates can be installed directly using *apt-get*, or via the GUI.

Apt *sources.list* files list one package repository per line, with the most preferred source listed first. Empty lines are ignored, and a # character anywhere on a line marks the remainder of that line as a comment. The information available from the configured sources is acquired by *apt-get update*.

File /etc/apt/sources.list

```
deb http://ftp.debian.org/debian jessie main contrib

# security updates
deb http://security.debian.org jessie/updates main contrib
```

In addition, Proxmox VE provides three different package repositories.

2.3.1 Proxmox VE Enterprise Repository

This is the default, stable and recommended repository, available for all Proxmox VE subscription users. It contains the most stable packages, and is suitable for production use. The `pve-enterprise` repository is enabled by default:

File /etc/apt/sources.list.d/pve-enterprise.list

```
deb https://enterprise.proxmox.com/debian jessie pve-enterprise
```

As soon as updates are available, the `root@pam` user is notified via email about the available new packages. On the GUI, the change-log of each package can be viewed (if available), showing all details of the update. So you will never miss important security fixes.

Please note that and you need a valid subscription key to access this repository. We offer different support levels, and you can find further details at http://www.proxmox.com/en/proxmox-ve/pricing.

Note
You can disable this repository by commenting out the above line using a # (at the start of the line). This prevents error messages if you do not have a subscription key. Please configure the `pve-no-subscription` repository in that case.

2.3.2 Proxmox VE No-Subscription Repository

As the name suggests, you do not need a subscription key to access this repository. It can be used for testing and non-production use. Its not recommended to run on production servers, as these packages are not always heavily tested and validated.

We recommend to configure this repository in */etc/apt/sources.list*.

File /etc/apt/sources.list

```
deb http://ftp.debian.org/debian jessie main contrib

# PVE pve-no-subscription repository provided by proxmox.com,
# NOT recommended for production use
deb http://download.proxmox.com/debian jessie pve-no-subscription

# security updates
deb http://security.debian.org jessie/updates main contrib
```

2.3.3 Proxmox VE Test Repository

Finally, there is a repository called `pvetest`. This one contains the latest packages and is heavily used by developers to test new features. As usual, you can configure this using */etc/apt/sources.list* by adding the following line:

sources.list entry for pvetest

```
deb http://download.proxmox.com/debian jessie pvetest
```

Warning
the `pvetest` repository should (as the name implies) only be used for testing new features or bug fixes.

2.3.4 SecureApt

We use GnuPG to sign the *Release* files inside those repositories, and APT uses that signatures to verify that all packages are from a trusted source.

The key used for verification is already installed if you install from our installation CD. If you install by other means, you can manually download the key with:

```
# wget http://download.proxmox.com/debian/key.asc
```

Please verify the fingerprint afterwards:

```
# gpg --with-fingerprint key.asc
pub  1024D/9887F95A 2008-10-28 Proxmox Release Key <proxmox-release@proxmox ↩
   .com>
      Key fingerprint = BE25 7BAA 5D40 6D01 157D  323E C23A C7F4 9887 F95A
sub  2048g/A87A1B00 2008-10-28
```

If this shows the exact same fingerprint, you can add the key to the list of trusted APT keys:

```
# apt-key add key.asc
```

2.4 Installing Proxmox VE

Proxmox VE ships as a set of Debian packages, so you can simply install it on top of a normal Debian installation. After configuring the repositories, you need to run:

```
apt-get update
apt-get install proxmox-ve
```

While this looks easy, it presumes that you have correctly installed the base system, and you know how you want to configure and use the local storage. Network configuration is also completely up to you.

In general, this is not trivial, especially when you use LVM or ZFS. This is why we provide an installation CD-ROM for Proxmox VE. That installer just ask you a few questions, then partitions the local disk(s), installs all required packages, and configures the system including a basic network setup. You can get a fully functional system within a few minutes, including the following:

- Complete operating system (Debian Linux, 64-bit)
- Partition the hard drive with ext4 (alternative ext3 or xfs) or ZFS
- Proxmox VE Kernel with LXC and KVM support
- Complete toolset
- Web based management interface

Note
By default, the complete server is used and all existing data is removed.

2.4.1 Using the Proxmox VE Installation CD-ROM

Please insert the installation CD-ROM, then boot from that drive. Immediately afterwards you can choose the following menu options:

Install Proxmox VE
Start normal installation.

Install Proxmox VE (Debug mode)
Start installation in debug mode. It opens a shell console at several installation steps, so that you can debug things if something goes wrong. Please press `CTRL-D` to exit those debug consoles and continue installation. This option is mostly for developers and not meant for general use.

Rescue Boot
: This option allows you to boot an existing installation. It searches all attached hard disks, and if it finds an existing installation, boots directly into that disk using the existing Linux kernel. This can be useful if there are problems with the boot block (grub), or the BIOS is unable to read the boot block from the disk.

Test Memory
: Runs *memtest86+*. This is useful to check if your memory if functional and error free.

You normally select **Install Proxmox VE** to start the installation. After that you get prompted to select the target hard disk(s). The `Options` button lets you select the target file system, which defaults to `ext4`. The installer uses LVM if you select *ext3*, *ext4* or *xfs* as file system, and offers additional option to restrict LVM space (see below)

If you have more than one disk, you can also use ZFS as file system. ZFS supports several software RAID levels, so this is specially useful if you do not have a hardware RAID controller. The `Options` button lets you select the ZFS RAID level, and you can choose disks there.

The next pages just asks for basic configuration options like time zone and keyboard layout. You also need to specify your email address and select a superuser password.

The last step is the network configuration. Please note that you can use either IPv4 or IPv6 here, but not both. If you want to configure a dual stack node, you can easily do that after installation.

If you press `Next` now, installation starts to format disks, and copies packages to the target. Please wait until that is finished, then reboot the server.

Further configuration is done via the Proxmox web interface. Just point your browser to the IP address given during installation (https://youripaddress:8006). Proxmox VE is tested for IE9, Firefox 10 and higher, and Google Chrome.

Advanced LVM configuration options

The installer creates a Volume Group (VG) called `pve`, and additional Logical Volumes (LVs) called `root`, `data` and `swap`. The size of those volumes can be controlled with:

`hdsize`
: Defines the total HD size to be used. This way you can save free space on the HD for further partitioning (i.e. for an additional PV and VG on the same hard disk that can be used for LVM storage).

`swapsize`
: To define the size of the `swap` volume. Default is the same size as installed RAM, with 4GB minimum and `hdsize/8` as maximum.

`maxroot`
: The `root` volume size. The `root` volume stores the whole operation system.

`maxvz`
: Define the size of the `data` volume, which is mounted at */var/lib/vz*.

minfree
To define the amount of free space left in LVM volume group `pve`. 16GB is the default if storage available > 128GB, `hdsize/8` otherwise.

Note
LVM requires free space in the VG for snapshot creation (not required for lvmthin snapshots).

ZFS Performance Tips

ZFS uses a lot of memory, so it is best to add additional 8-16GB RAM if you want to use ZFS.

ZFS also provides the feature to use a fast SSD drive as write cache. The write cache is called the ZFS Intent Log (ZIL). You can add that after installation using the following command:

```
zpool add <pool-name> log </dev/path_to_fast_ssd>
```

2.5 System Software Updates

We provide regular package updates on all repositories. You can install those update using the GUI, or you can directly run the CLI command *apt-get*:

```
apt-get update
apt-get dist-upgrade
```

Note
The *apt* package management system is extremely flexible and provides countless of feature - see `man apt-get` or [Hertzog13] for additional information.

You should do such updates at regular intervals, or when we release versions with security related fixes. Major system upgrades are announced at the Forum. Those announcement also contain detailed upgrade instructions.

Tip
We recommend to run regular upgrades, because it is important to get the latest security updates.

2.6 Network Configuration

Proxmox VE uses a bridged networking model. Each host can have up to 4094 bridges. Bridges are like physical network switches implemented in software. All VMs can share a single bridge, as if virtual network cables from each guest were all plugged into the same switch. But you can also create multiple bridges to separate network domains.

For connecting VMs to the outside world, bridges are attached to physical network cards. For further flexibility, you can configure VLANs (IEEE 802.1q) and network bonding, also known as "link aggregation". That way it is possible to build complex and flexible virtual networks.

Debian traditionally uses the *ifup* and *ifdown* commands to configure the network. The file */etc/network/interfaces* contains the whole network setup. Please refer to to manual page (*man interfaces*) for a complete format description.

Note
Proxmox VE does not write changes directly to */etc/network/interfaces*. Instead, we write into a temporary file called */etc/network/interfaces.new*, and commit those changes when you reboot the node.

It is worth mentioning that you can directly edit the configuration file. All Proxmox VE tools tries hard to keep such direct user modifications. Using the GUI is still preferable, because it protect you from errors.

2.6.1 Naming Conventions

We currently use the following naming conventions for device names:

- Ethernet devices: eth[N], where $0 \leq N$ (`eth0`, `eth1`, ...)
- Bridge names: vmbr[N], where $0 \leq N \leq 4094$ (`vmbr0` - `vmbr4094`)
- Bonds: bond[N], where $0 \leq N$ (`bond0`, `bond1`, ...)
- VLANs: Simply add the VLAN number to the device name, separated by a period (`eth0.50`, `bond1.30`)

This makes it easier to debug networks problems, because the device names implies the device type.

2.6.2 Default Configuration using a Bridge

The installation program creates a single bridge named `vmbr0`, which is connected to the first ethernet card `eth0`. The corresponding configuration in */etc/network/interfaces* looks like this:

```
auto lo
iface lo inet loopback

iface eth0 inet manual

auto vmbr0
iface vmbr0 inet static
        address 192.168.10.2
        netmask 255.255.255.0
        gateway 192.168.10.1
        bridge_ports eth0
        bridge_stp off
        bridge_fd 0
```

Virtual machines behave as if they were directly connected to the physical network. The network, in turn, sees each virtual machine as having its own MAC, even though there is only one network cable connecting all of these VMs to the network.

2.6.3 Routed Configuration

Most hosting providers do not support the above setup. For security reasons, they disable networking as soon as they detect multiple MAC addresses on a single interface.

Tip
Some providers allows you to register additional MACs on there management interface. This avoids the problem, but is clumsy to configure because you need to register a MAC for each of your VMs.

You can avoid the problem by "routing" all traffic via a single interface. This makes sure that all network packets use the same MAC address.

A common scenario is that you have a public IP (assume 192.168.10.2 for this example), and an additional IP block for your VMs (10.10.10.1/255.255.255.0). We recommend the following setup for such situations:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address  192.168.10.2
        netmask  255.255.255.0
        gateway  192.168.10.1
        post-up echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp


auto vmbr0
iface vmbr0 inet static
        address  10.10.10.1
        netmask  255.255.255.0
        bridge_ports none
        bridge_stp off
        bridge_fd 0
```

2.6.4 Masquerading (NAT) with iptables

In some cases you may want to use private IPs behind your Proxmox host's true IP, and masquerade the traffic using NAT:

```
auto lo
iface lo inet loopback

auto eth0
```

```
#real IP adress
iface eth0 inet static
        address  192.168.10.2
        netmask  255.255.255.0
        gateway  192.168.10.1

auto vmbr0
#private sub network
iface vmbr0 inet static
        address  10.10.10.1
        netmask  255.255.255.0
        bridge_ports none
        bridge_stp off
        bridge_fd 0

        post-up echo 1 > /proc/sys/net/ipv4/ip_forward
        post-up   iptables -t nat -A POSTROUTING -s '10.10.10.0/24' -o eth0 ↩
            -j MASQUERADE
        post-down iptables -t nat -D POSTROUTING -s '10.10.10.0/24' -o eth0 ↩
            -j MASQUERADE
```

2.7 Logical Volume Manager (LVM)

Most people install Proxmox VE directly on a local disk. The Proxmox VE installation CD offers several options for local disk management, and the current default setup uses LVM. The installer let you select a single disk for such setup, and uses that disk as physical volume for the **V**olume **G**roup (VG) *pve*. The following output is from a test installation using a small 8GB disk:

```
# pvs
  PV         VG   Fmt  Attr PSize PFree
  /dev/sda3  pve  lvm2 a--  7.87g 876.00m

# vgs
  VG   #PV #LV #SN Attr   VSize VFree
  pve    1   3   0 wz--n- 7.87g 876.00m
```

The installer allocates three **L**ogical **V**olumes (LV) inside this VG:

```
# lvs
  LV   VG   Attr       LSize   Pool Origin Data%  Meta%
  data pve  twi-a-tz--   4.38g             0.00   0.63
  root pve  -wi-ao----   1.75g
  swap pve  -wi-ao---- 896.00m
```

root
: Formatted as *ext4*, and contains the operation system.

swap
Swap partition

data
This volume uses LVM-thin, and is used to store VM images. LVM-thin is preferable for this task, because it offers efficient support for snapshots and clones.

2.7.1 Hardware

We highly recommend to use a hardware RAID controller (with BBU) for such setups. This increases performance, provides redundancy, and make disk replacements easier (hot-pluggable).

LVM itself does not need any special hardware, and memory requirements are very low.

2.7.2 Bootloader

We install two boot loaders by default. The first partition contains the standard GRUB boot loader. The second partition is an **E**FI **S**ystem **P**artition (ESP), which makes it possible to boot on EFI systems.

2.8 ZFS on Linux

ZFS is a combined file system and logical volume manager designed by Sun Microsystems. Starting with Proxmox VE 3.4, the native Linux kernel port of the ZFS file system is introduced as optional file-system and also as an additional selection for the root file-system. There is no need for manually compile ZFS modules - all packages are included.

By using ZFS, its possible to achieve maximal enterprise features with low budget hardware, but also high performance systems by leveraging SSD caching or even SSD only setups. ZFS can replace cost intense hardware raid cards by moderate CPU and memory load combined with easy management.

GENERAL ZFS ADVANTAGES

- Easy configuration and management with Proxmox VE GUI and CLI.
- Reliable
- Protection against data corruption
- Data compression on file-system level
- Snapshots
- Copy-on-write clone
- Various raid levels: RAID0, RAID1, RAID10, RAIDZ-1, RAIDZ-2 and RAIDZ-3
- Can use SSD for cache
- Self healing

- Continuous integrity checking
- Designed for high storage capacities
- Protection against data corruption
- Asynchronous replication over network
- Open Source
- Encryption
- ...

2.8.1 Hardware

ZFS depends heavily on memory, so you need at least 8GB to start. In practice, use as much you can get for your hardware/budget. To prevent data corruption, we recommend the use of high quality ECC RAM.

If you use a dedicated cache and/or log disk, you should use a enterprise class SSD (e.g. Intel SSD DC S3700 Series). This can increase the overall performance significantly.

Important
Do not use ZFS on top of hardware controller which has it's own cache management. ZFS needs to directly communicate with disks. An HBA adapter is the way to go, or something like LSI controller flashed in *IT* mode.

If you are experimenting with an installation of Proxmox VE inside a VM (Nested Virtualization), don't use *virtio* for disks of that VM, since they are not supported by ZFS. Use IDE or SCSI instead (works also with *virtio* SCSI controller type).

2.8.2 Installation as root file system

When you install using the Proxmox VE installer, you can choose ZFS for the root file system. You need to select the RAID type at installation time:

RAID0	Also called *striping*. The capacity of such volume is the sum of the capacity of all disks. But RAID0 does not add any redundancy, so the failure of a single drive makes the volume unusable.
RAID1	Also called mirroring. Data is written identically to all disks. This mode requires at least 2 disks with the same size. The resulting capacity is that of a single disk.
RAID10	A combination of RAID0 and RAID1. Requires at least 4 disks.
RAIDZ-1	A variation on RAID-5, single parity. Requires at least 3 disks.

RAIDZ-2 A variation on RAID-5, double parity. Requires at least 4 disks.

RAIDZ-3 A variation on RAID-5, triple parity. Requires at least 5 disks.

The installer automatically partitions the disks, creates a ZFS pool called *rpool*, and installs the root file system on the ZFS subvolume *rpool/ROOT/pve-1*.

Another subvolume called *rpool/data* is created to store VM images. In order to use that with the Proxmox VE tools, the installer creates the following configuration entry in */etc/pve/storage.cfg*:

```
zfspool: local-zfs
        pool rpool/data
        sparse
        content images,rootdir
```

After installation, you can view your ZFS pool status using the *zpool* command:

```
# zpool status
  pool: rpool
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        rpool       ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            sda2    ONLINE       0     0     0
            sdb2    ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            sdc     ONLINE       0     0     0
            sdd     ONLINE       0     0     0

errors: No known data errors
```

The *zfs* command is used configure and manage your ZFS file systems. The following command lists all file systems after installation:

```
# zfs list
NAME                USED  AVAIL  REFER  MOUNTPOINT
rpool              4.94G  7.68T    96K  /rpool
rpool/ROOT          702M  7.68T    96K  /rpool/ROOT
rpool/ROOT/pve-1    702M  7.68T   702M  /
rpool/data           96K  7.68T    96K  /rpool/data
rpool/swap         4.25G  7.69T    64K  -
```

2.8.3 Bootloader

The default ZFS disk partitioning scheme does not use the first 2048 sectors. This gives enough room to install a GRUB boot partition. The Proxmox VE installer automatically allocates that space, and installs the

GRUB boot loader there. If you use a redundant RAID setup, it installs the boot loader on all disk required for booting. So you can boot even if some disks fail.

Note
It is not possible to use ZFS as root partition with UEFI boot.

2.8.4 ZFS Administration

This section gives you some usage examples for common tasks. ZFS itself is really powerful and provides many options. The main commands to manage ZFS are *zfs* and *zpool*. Both commands comes with great manual pages, worth to read:

```
# man zpool
# man zfs
```

Create a new ZPool To create a new pool, at least one disk is needed. The *ashift* should have the same sector-size (2 power of *ashift*) or larger as the underlying disk.

```
zpool create -f -o ashift=12 <pool> <device>
```

To activate the compression

```
zfs set compression=lz4 <pool>
```

Create a new pool with RAID-0 Minimum 1 Disk

```
zpool create -f -o ashift=12 <pool> <device1> <device2>
```

Create a new pool with RAID-1 Minimum 2 Disks

```
zpool create -f -o ashift=12 <pool> mirror <device1> <device2>
```

Create a new pool with RAID-10 Minimum 4 Disks

```
zpool create -f -o ashift=12 <pool> mirror <device1> <device2>  ↩
   mirror <device3> <device4>
```

Create a new pool with RAIDZ-1 Minimum 3 Disks

```
zpool create -f -o ashift=12 <pool> raidz1 <device1> <device2> < ↩
   device3>
```

Create a new pool with RAIDZ-2 Minimum 4 Disks

```
zpool create -f -o ashift=12 <pool> raidz2 <device1> <device2> < ↩
   device3> <device4>
```

Create a new pool with Cache (L2ARC) It is possible to use a dedicated cache drive partition to increase the performance (use SSD).

As <*device*> it is possible to use more devices, like it's shown in "Create a new pool with RAID*".

```
zpool create -f -o ashift=12 <pool> <device> cache <cache_device>
```

Create a new pool with Log (ZIL) It is possible to use a dedicated cache drive partition to increase the performance(SSD).

As <*device*> it is possible to use more devices, like it's shown in "Create a new pool with RAID*".

```
zpool create -f -o ashift=12 <pool> <device> log <log_device>
```

Add Cache and Log to an existing pool If you have an pool without cache and log. First partition the SSD in 2 partition with parted or gdisk

Important
Always use GPT partition tables (gdisk or parted).

The maximum size of a log device should be about half the size of physical memory, so this is usually quite small. The rest of the SSD can be used to the cache.

```
zpool add -f <pool> log <device-part1> cache <device-part2>
```

Changing a failed Device

```
zpool replace -f <pool> <old device> <new-device>
```

2.8.5 Activate E-Mail Notification

ZFS comes with an event daemon, which monitors events generated by the ZFS kernel module. The daemon can also send E-Mails on ZFS event like pool errors.

To activate the daemon it is necessary to edit /etc/zfs/zed.d/zed.rc with your favored editor, and uncomment the *ZED_EMAIL_ADDR* setting:

ZED_EMAIL_ADDR="root"

Please note Proxmox VE forwards mails to *root* to the email address configured for the root user.

Important
the only settings that is required is ZED_EMAIL_ADDR. All other settings are optional.

2.8.6 Limit ZFS memory usage

It is good to use maximal 50 percent (which is the default) of the system memory for ZFS ARC to prevent performance shortage of the host. Use your preferred editor to change the configuration in /etc/modprobe.d/zfs.conf and insert:

```
options zfs zfs_arc_max=8589934592
```

This example setting limits the usage to 8GB.

Important
If your root fs is ZFS you must update your initramfs every time this value changes.

```
update-initramfs -u
```

SWAP on ZFS SWAP on ZFS on Linux may generate some troubles, like blocking the server or generating a high IO load, often seen when starting a Backup to an external Storage.

We strongly recommend to use enough memory, so that you normally do not run into low memory situations. Additionally, you can lower the *swappiness* value. A good value for servers is 10:

```
sysctl -w vm.swappiness=10
```

To make the swappiness persistence, open */etc/sysctl.conf* with an editor of your choice and add the following line:

```
vm.swappiness = 10
```

Table 2.1: Linux Kernel *swappiness* parameter values

Value	Strategy
`vm.swappiness =0`	The kernel will swap only to avoid an *out of memory* condition
`vm.swappiness =1`	Minimum amount of swapping without disabling it entirely.
`vm.swappiness =10`	This value is sometimes recommended to improve performance when sufficient memory exists in a system.
`vm.swappiness =60`	The default value.
`vm.swappiness =100`	The kernel will swap aggressively.

Chapter 3

Cluster Manager

The Proxmox VE cluster manager *pvecm* is a tool to create a group of physical servers. Such group is called a **cluster**. We use the Corosync Cluster Engine for reliable group communication, and such cluster can consists of up to 32 physical nodes (probably more, dependent on network latency).

pvecm can be used to create a new cluster, join nodes to a cluster, leave the cluster, get status information and do various other cluster related tasks. The Proxmox Cluster file system (pmxcfs) is used to transparently distribute the cluster configuration to all cluster nodes.

Grouping nodes into a cluster has the following advantages:

- Centralized, web based management
- Multi-master clusters: Each node can do all management task
- Proxmox Cluster file system (pmxcfs): Database-driven file system for storing configuration files, replicated in real-time on all nodes using corosync.
- Easy migration of Virtual Machines and Containers between physical hosts
- Fast deployment
- Cluster-wide services like firewall and HA

3.1 Requirements

- All nodes must be in the same network as corosync uses IP Multicast to communicate between nodes (also see Corosync Cluster Engine). Corosync uses UDP ports 5404 and 5405 for cluster communication.

  Note
  Some switches do not support IP multicast by default and must be manually enabled first.

- Date and time have to be synchronized.

- SSH tunnel on TCP port 22 between nodes is used.
- If you are interested in High Availability, you need to have at least three nodes for reliable quorum. All nodes should have the same version.
- We recommend a dedicated NIC for the cluster traffic, especially if you use shared storage.

Note
It is not possible to mix Proxmox VE 3.x and earlier with Proxmox VE 4.0 cluster nodes.

3.2 Preparing Nodes

First, install Proxmox VE on all nodes. Make sure that each node is installed with the final hostname and IP configuration. Changing the hostname and IP is not possible after cluster creation.

Currently the cluster creation has to be done on the console, so you need to login via *ssh*.

3.3 Create the Cluster

Login via *ssh* to the first Proxmox VE node. Use a unique name for your cluster. This name cannot be changed later.

```
hp1# pvecm create YOUR-CLUSTER-NAME
```

Caution
The cluster name is used to compute the default multicast address. Please use unique cluster names if you run more than one cluster inside your network.

To check the state of your cluster use:

```
hp1# pvecm status
```

3.4 Adding Nodes to the Cluster

Login via *ssh* to the node you want to add.

```
hp2# pvecm add IP-ADDRESS-CLUSTER
```

For `IP-ADDRESS-CLUSTER` use the IP from an existing cluster node.

Caution
A new node cannot hold any VM´s, because you would get conflicts about identical VM IDs. Also, all existing configuration in */etc/pve* is overwritten when you join a new node to the cluster. To workaround, use vzdump to backup and restore to a different VMID after adding the node to the cluster.

To check the state of cluster:

```
# pvecm status
```

Cluster status after adding 4 nodes

```
hp2# pvecm status
Quorum information
~~~~~~~~~~~~~~~~~~
Date:             Mon Apr 20 12:30:13 2015
Quorum provider:  corosync_votequorum
Nodes:            4
Node ID:          0x00000001
Ring ID:          1928
Quorate:          Yes

Votequorum information
~~~~~~~~~~~~~~~~~~~~~~
Expected votes:   4
Highest expected: 4
Total votes:      4
Quorum:           2
Flags:            Quorate

Membership information
~~~~~~~~~~~~~~~~~~~~~~
    Nodeid      Votes Name
0x00000001          1 192.168.15.91
0x00000002          1 192.168.15.92 (local)
0x00000003          1 192.168.15.93
0x00000004          1 192.168.15.94
```

If you only want the list of all nodes use:

```
# pvecm nodes
```

List Nodes in a Cluster

```
hp2# pvecm nodes

Membership information
~~~~~~~~~~~~~~~~~~~~~~
    Nodeid      Votes Name
         1          1 hp1
```

```
         2          1 hp2 (local)
         3          1 hp3
         4          1 hp4
```

3.5 Remove a Cluster Node

Caution
Read carefully the procedure before proceeding, as it could not be what you want or need.

Move all virtual machines from the node. Make sure you have no local data or backups you want to keep, or save them accordingly.

Log in to one remaining node via ssh. Issue a *pvecm nodes* command to identify the node ID:

```
hp1# pvecm status

Quorum information
~~~~~~~~~~~~~~~~~~
Date:             Mon Apr 20 12:30:13 2015
Quorum provider:  corosync_votequorum
Nodes:            4
Node ID:          0x00000001
Ring ID:          1928
Quorate:          Yes

Votequorum information
~~~~~~~~~~~~~~~~~~~~~~
Expected votes:   4
Highest expected: 4
Total votes:      4
Quorum:           2
Flags:            Quorate

Membership information
~~~~~~~~~~~~~~~~~~~~~~
    Nodeid      Votes Name
0x00000001          1 192.168.15.91 (local)
0x00000002          1 192.168.15.92
0x00000003          1 192.168.15.93
0x00000004          1 192.168.15.94
```

Important
at this point you must power off the node to be removed and make sure that it will not power on again (in the network) as it is.

```
hp1# pvecm nodes

Membership information
~~~~~~~~~~~~~~~~~~~~~~
    Nodeid      Votes Name
         1          1 hp1 (local)
         2          1 hp2
         3          1 hp3
         4          1 hp4
```

Log in to one remaining node via ssh. Issue the delete command (here deleting node hp4):

```
hp1# pvecm delnode hp4
```

If the operation succeeds no output is returned, just check the node list again with *pvecm nodes* or *pvecm status*. You should see something like:

```
hp1# pvecm status

Quorum information
~~~~~~~~~~~~~~~~~~
Date:             Mon Apr 20 12:44:28 2015
Quorum provider:  corosync_votequorum
Nodes:            3
Node ID:          0x00000001
Ring ID:          1992
Quorate:          Yes

Votequorum information
~~~~~~~~~~~~~~~~~~~~~~
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           3
Flags:            Quorate

Membership information
~~~~~~~~~~~~~~~~~~~~~~
    Nodeid      Votes Name
0x00000001          1 192.168.15.90 (local)
0x00000002          1 192.168.15.91
0x00000003          1 192.168.15.92
```

Important
as said above, it is very important to power off the node **before** removal, and make sure that it will **never** power on again (in the existing cluster network) as it is.

If you power on the node as it is, your cluster will be screwed up and it could be difficult to restore a clean cluster state.

If, for whatever reason, you want that this server joins the same cluster again, you have to

- reinstall pve on it from scratch
- then join it, as explained in the previous section.

3.6 Quorum

Proxmox VE use a quorum-based technique to provide a consistent state among all cluster nodes.

> A quorum is the minimum number of votes that a distributed transaction has to obtain in order to be allowed to perform an operation in a distributed system.
>
> — from Wikipedia *Quorum (distributed computing)*

In case of network partitioning, state changes requires that a majority of nodes are online. The cluster switches to read-only mode if it loose quorum.

Note
Proxmox VE assigns a single vote to each node by default.

3.7 Cluster Cold Start

It is obvious that a cluster is not quorate when all nodes are offline. This is a common case after a power failure.

Note
It is always a good idea to use an uninterruptible power supply (*UPS*, also called *battery backup*) to avoid this state. Especially if you want HA.

On node startup, service *pve-manager* is started and waits for quorum. Once quorate, it starts all guests which have the *onboot* flag set.

When you turn on nodes, or when power comes back after power failure, it is likely that some nodes boots faster than others. Please keep in mind that guest startup is delayed until you reach quorum.

Chapter 4

Proxmox Cluster File System (pmxcfs)

The Proxmox Cluster file system (pmxcfs) is a database-driven file system for storing configuration files, replicated in real time to all cluster nodes using corosync. We use this to store all PVE related configuration files.

Although the file system stores all data inside a persistent database on disk, a copy of the data resides in RAM. That imposes restriction on the maximal size, which is currently 30MB. This is still enough to store the configuration of several thousand virtual machines.

This system provides the following advantages:

- seamless replication of all configuration to all nodes in real time
- provides strong consistency checks to avoid duplicate VM IDs
- read-only when a node loses quorum
- automatic updates of the corosync cluster configuration to all nodes
- includes a distributed locking mechanism

4.1 POSIX Compatibility

The file system is based on FUSE, so the behavior is POSIX like. But some feature are simply not implemented, because we do not need them:

- you can just generate normal files and directories, but no symbolic links, ...
- you can't rename non-empty directories (because this makes it easier to guarantee that VMIDs are unique).
- you can't change file permissions (permissions are based on path)
- `O_EXCL` creates were not atomic (like old NFS)
- `O_TRUNC` creates are not atomic (FUSE restriction)

4.2 File access rights

All files and directories are owned by user *root* and have group *www-data*. Only root has write permissions, but group *www-data* can read most files. Files below the following paths:

```
/etc/pve/priv/
/etc/pve/nodes/${NAME}/priv/
```

are only accessible by root.

4.3 Technology

We use the Corosync Cluster Engine for cluster communication, and SQlite for the database file. The filesystem is implemented in user space using FUSE.

4.4 File system layout

The file system is mounted at:

```
/etc/pve
```

4.4.1 Files

`corosync.conf`	corosync cluster configuration file (previous to Proxmox VE 4.x this file was called cluster.conf)
`storage.cfg`	Proxmox VE storage configuration
`datacenter.cfg`	Proxmox VE datacenter wide configuration (keyboard layout, proxy, . . .)
`user.cfg`	Proxmox VE access control configuration (users/groups/. . .)
`domains.cfg`	Proxmox VE Authentication domains
`authkey.pub`	public key used by ticket system
`pve-root-ca.pem`	public certificate of cluster CA
`priv/shadow.cfg`	shadow password file
`priv/authkey.key`	private key used by ticket system
`priv/pve-root-ca.key`	private key of cluster CA
`nodes/<NAME>/pve-ssl.pem`	public ssl certificate for web server (signed by cluster CA)
`nodes/<NAME>/pve-ssl.key`	private ssl key for pve-ssl.pem
`nodes/<NAME>/pveproxy-ssl.pem`	public ssl certificate (chain) for web server (optional override for pve-ssl.pem)
`nodes/<NAME>/pveproxy-ssl.key`	private ssl key for pveproxy-ssl.pem (optional)

nodes/<NAME>/qemu-server/<VMID>.conf	VM configuration data for KVM VMs
nodes/<NAME>/lxc/<VMID>.conf	VM configuration data for LXC containers
firewall/cluster.fw	Firewall config applied to all nodes
firewall/<NAME>.fw	Firewall config for individual nodes
firewall/<VMID>.fw	Firewall config for VMs and Containers

4.4.2 Symbolic links

local	nodes/<LOCAL_HOST_NAME>
qemu-server	nodes/<LOCAL_HOST_NAME>/qemu-server/
lxc	nodes/<LOCAL_HOST_NAME>/lxc/

4.4.3 Special status files for debugging (JSON)

.version	file versions (to detect file modifications)
.members	Info about cluster members
.vmlist	List of all VMs
.clusterlog	Cluster log (last 50 entries)
.rrd	RRD data (most recent entries)

4.4.4 Enable/Disable debugging

You can enable verbose syslog messages with:

```
echo "1" >/etc/pve/.debug
```

And disable verbose syslog messages with:

```
echo "0" >/etc/pve/.debug
```

4.5 Recovery

If you have major problems with your Proxmox VE host, e.g. hardware issues, it could be helpful to just copy the pmxcfs database file /var/lib/pve-cluster/config.db and move it to a new Proxmox VE host. On the new host (with nothing running), you need to stop the pve-cluster service and replace the config.db file (needed permissions 0600). Second, adapt */etc/hostname* and */etc/hosts* according to the lost Proxmox VE host, then reboot and check. (And don´t forget your VM/CT data)

4.5.1 Remove Cluster configuration

The recommended way is to reinstall the node after you removed it from your cluster. This makes sure that all secret cluster/ssh keys and any shared configuration data is destroyed.

In some cases, you might prefer to put a node back to local mode without reinstall, which is described here:

- stop the cluster file system in */etc/pve/*

```
# systemctl stop pve-cluster
```

- start it again but forcing local mode

```
# pmxcfs -l
```

- remove the cluster config

```
# rm /etc/pve/cluster.conf
# rm /etc/cluster/cluster.conf
# rm /var/lib/pve-cluster/corosync.authkey
```

- stop the cluster file system again

```
# systemctl stop pve-cluster
```

- restart pve services (or reboot)

```
# systemctl start pve-cluster
# systemctl restart pvedaemon
# systemctl restart pveproxy
# systemctl restart pvestatd
```

Chapter 5

Proxmox VE Storage

The Proxmox VE storage model is very flexible. Virtual machine images can either be stored on one or several local storages, or on shared storage like NFS or iSCSI (NAS, SAN). There are no limits, and you may configure as many storage pools as you like. You can use all storage technologies available for Debian Linux.

One major benefit of storing VMs on shared storage is the ability to live-migrate running machines without any downtime, as all nodes in the cluster have direct access to VM disk images. There is no need to copy VM image data, so live migration is very fast in that case.

The storage library (package *libpve-storage-perl*) uses a flexible plugin system to provide a common interface to all storage types. This can be easily adopted to include further storage types in future.

5.1 Storage Types

There are basically two different classes of storage types:

Block level storage
Allows to store large *raw* images. It is usually not possible to store other files (ISO, backups, ..) on such storage types. Most modern block level storage implementations support snapshots and clones. RADOS, Sheepdog and DRBD are distributed systems, replicating storage data to different nodes.

File level storage
They allow access to a full featured (POSIX) file system. They are more flexible, and allows you to store any content type. ZFS is probably the most advanced system, and it has full support for snapshots and clones.

Table 5.1: Available storage types

Description	PVE type	Level	Shared	Snapshots	Stable
ZFS (local)	`zfspool`	file	no	yes	yes
Directory	`dir`	file	no	no	yes
NFS	`nfs`	file	yes	no	yes

Table 5.1: (continued)

Description	PVE type	Level	Shared	Snapshots	Stable
GlusterFS	`glusterfs`	file	yes	no	yes
LVM	`lvm`	block	no	no	yes
LVM-thin	`lvmthin`	block	no	yes	yes
iSCSI/kernel	`iscsi`	block	yes	no	yes
iSCSI/libiscsi	`iscsidirect`	block	yes	no	yes
Ceph/RBD	`rbd`	block	yes	yes	yes
Sheepdog	`sheepdog`	block	yes	yes	beta
DRBD9	`drbd`	block	yes	yes	beta
ZFS over iSCSI	`zfs`	block	yes	yes	yes

Tip
It is possible to use LVM on top of an iSCSI storage. That way you get a *shared* LVM storage.

5.1.1 Thin provisioning

A number of storages, and the Qemu image format `qcow2`, support *thin provisioning*. With thin provisioning activated, only the blocks that the guest system actually use will be written to the storage.

Say for instance you create a VM with a 32GB hard disk, and after installing the guest system OS, the root filesystem of the VM contains 3 GB of data. In that case only 3GB are written to the storage, even if the guest VM sees a 32GB hard drive. In this way thin provisioning allows you to create disk images which are larger than the currently available storage blocks. You can create large disk images for your VMs, and when the need arises, add more disks to your storage without resizing the VMs filesystems.

All storage types which have the *Snapshots* feature also support thin provisioning.

Caution
If a storage runs full, all guests using volumes on that storage receives IO error. This can cause file system inconsistencies and may corrupt your data. So it is advisable to avoid over-provisioning of your storage resources, or carefully observe free space to avoid such conditions.

5.2 Storage Configuration

All Proxmox VE related storage configuration is stored within a single text file at */etc/pve/storage.cfg*. As this file is within */etc/pve/*, it gets automatically distributed to all cluster nodes. So all nodes share the same storage configuration.

Sharing storage configuration make perfect sense for shared storage, because the same *shared* storage is accessible from all nodes. But is also useful for local storage types. In this case such local storage is available on all nodes, but it is physically different and can have totally different content.

5.2.1 Storage Pools

Each storage pool has a `<type>`, and is uniquely identified by its `<STORAGE_ID>`. A pool configuration looks like this:

```
<type>: <STORAGE_ID>
        <property> <value>
        <property> <value>
        ...
```

The `<type>:<STORAGE_ID>` line starts the pool definition, which is then followed by a list of properties. Most properties have values, but some of them come with reasonable default. In that case you can omit the value.

To be more specific, take a look at the default storage configuration after installation. It contains one special local storage pool named `local`, which refers to the directory */var/lib/vz* and is always available. The Proxmox VE installer creates additional storage entries depending on the storage type chosen at installation time.

Default storage configuration (/etc/pve/storage.cfg)

```
dir: local
        path /var/lib/vz
        content iso,vztmpl,backup

# default image store on LVM based installation
lvmthin: local-lvm
        thinpool data
        vgname pve
        content rootdir,images

# default image store on ZFS based installation
zfspool: local-zfs
        pool rpool/data
        sparse
        content images,rootdir
```

5.2.2 Common Storage Properties

A few storage properties are common among different storage types.

nodes
: List of cluster node names where this storage is usable/accessible. One can use this property to restrict storage access to a limited set of nodes.

content
A storage can support several content types, for example virtual disk images, cdrom iso images, container templates or container root directories. Not all storage types support all content types. One can set this property to select for what this storage is used for.

images
KVM-Qemu VM images.

rootdir
Allow to store container data.

vztmpl
Container templates.

backup
Backup files (*vzdump*).

iso
ISO images

shared
Mark storage as shared.

disable
You can use this flag to disable the storage completely.

maxfiles
Maximal number of backup files per VM. Use `0` for unlimted.

format
Default image format (`raw|qcow2|vmdk`)

Warning
It is not advisable to use the same storage pool on different Proxmox VE clusters. Some storage operation need exclusive access to the storage, so proper locking is required. While this is implemented within a cluster, it does not work between different clusters.

5.3 Volumes

We use a special notation to address storage data. When you allocate data from a storage pool, it returns such a volume identifier. A volume is identified by the `<STORAGE_ID>`, followed by a storage type dependent volume name, separated by colon. A valid `<VOLUME_ID>` looks like:

```
local:230/example-image.raw

local:iso/debian-501-amd64-netinst.iso

local:vztmpl/debian-5.0-joomla_1.5.9-1_i386.tar.gz
```

```
iscsi-storage:0.0.2.scsi-14 ↩
   f504e46494c4500494b5042546d2d646744372d31616d61
```

To get the filesystem path for a <VOLUME_ID> use:

```
pvesm path <VOLUME_ID>
```

5.3.1 Volume Ownership

There exists an ownership relation for *image* type volumes. Each such volume is owned by a VM or Container. For example volume `local:230/example-image.raw` is owned by VM 230. Most storage backends encodes this ownership information into the volume name.

When you remove a VM or Container, the system also removes all associated volumes which are owned by that VM or Container.

5.4 Using the Command Line Interface

It is recommended to familiarize yourself with the concept behind storage pools and volume identifiers, but in real life, you are not forced to do any of those low level operations on the command line. Normally, allocation and removal of volumes is done by the VM and Container management tools.

Nevertheless, there is a command line tool called *pvesm* (Proxmox VE storage manager), which is able to perform common storage management tasks.

5.4.1 Examples

Add storage pools

```
pvesm add <TYPE> <STORAGE_ID> <OPTIONS>
pvesm add dir <STORAGE_ID> --path <PATH>
pvesm add nfs <STORAGE_ID> --path <PATH> --server <SERVER> --export  ↩
   <EXPORT>
pvesm add lvm <STORAGE_ID> --vgname <VGNAME>
pvesm add iscsi <STORAGE_ID> --portal <HOST[:PORT]> --target <TARGET ↩
   >
```

Disable storage pools

```
pvesm set <STORAGE_ID> --disable 1
```

Enable storage pools

```
pvesm set <STORAGE_ID> --disable 0
```

Change/set storage options

```
pvesm set <STORAGE_ID> <OPTIONS>
pvesm set <STORAGE_ID> --shared 1
pvesm set local --format qcow2
pvesm set <STORAGE_ID> --content iso
```

Remove storage pools. This does not delete any data, and does not disconnect or unmount anything. It just removes the storage configuration.

```
pvesm remove <STORAGE_ID>
```

Allocate volumes

```
pvesm alloc <STORAGE_ID> <VMID> <name> <size> [--format <raw|qcow2>]
```

Allocate a 4G volume in local storage. The name is auto-generated if you pass an empty string as `<name>`

```
pvesm alloc local <VMID> '' 4G
```

Free volumes

```
pvesm free <VOLUME_ID>
```

Warning
This really destroys all volume data.

List storage status

```
pvesm status
```

List storage contents

```
pvesm list <STORAGE_ID> [--vmid <VMID>]
```

List volumes allocated by VMID

```
pvesm list <STORAGE_ID> --vmid <VMID>
```

List iso images

```
pvesm list <STORAGE_ID> --iso
```

List container templates

```
pvesm list <STORAGE_ID> --vztmpl
```

Show filesystem path for a volume

```
pvesm path <VOLUME_ID>
```

5.5 Directory Backend

Storage pool type: `dir`

Proxmox VE can use local directories or locally mounted shares for storage. A directory is a file level storage, so you can store any content type like virtual disk images, containers, templates, ISO images or backup files.

Note
You can mount additional storages via standard linux */etc/fstab*, and then define a directory storage for that mount point. This way you can use any file system supported by Linux.

This backend assumes that the underlying directory is POSIX compatible, but nothing else. This implies that you cannot create snapshots at the storage level. But there exists a workaround for VM images using the `qcow2` file format, because that format supports snapshots internally.

Tip
Some storage types do not support `O_DIRECT`, so you can't use cache mode `none` with such storages. Simply use cache mode `writeback` instead.

We use a predefined directory layout to store different content types into different sub-directories. This layout is used by all file level storage backends.

Table 5.2: Directory layout

Content type	Subdir
VM images	`images/<VMID>/`
ISO images	`template/iso/`
Container templates	`template/cache`
Backup files	`dump/`

5.5.1 Configuration

This backend supports all common storage properties, and adds an additional property called `path` to specify the directory. This needs to be an absolute file system path.

Configuration Example (/etc/pve/storage.cfg)

```
dir: backup
        path /mnt/backup
        content backup
        maxfiles 7
```

Above configuration defines a storage pool called `backup`. That pool can be used to store up to 7 backups (`maxfiles 7`) per VM. The real path for the backup files is */mnt/backup/dump/...*.

5.5.2 File naming conventions

This backend uses a well defined naming scheme for VM images:

`vm-<VMID>-<NAME>.<FORMAT>`

<VMID>
This specifies the owner VM.

<NAME>
This can be an arbitrary name (`ascii`) without white spaces. The backend uses `disk-[N]` as default, where `[N]` is replaced by an integer to make the name unique.

<FORMAT>
Species the image format (`raw|qcow2|vmdk`).

When you create a VM template, all VM images are renamed to indicate that they are now read-only, and can be uses as a base image for clones:

`base-<VMID>-<NAME>.<FORMAT>`

Note
Such base images are used to generate cloned images. So it is important that those files are read-only, and never get modified. The backend changes the access mode to `0444`, and sets the immutable flag (`chattr +i`) if the storage supports that.

5.5.3 Storage Features

As mentioned above, most file systems do not support snapshots out of the box. To workaround that problem, this backend is able to use `qcow2` internal snapshot capabilities.

Same applies to clones. The backend uses the `qcow2` base image feature to create clones.

Table 5.3: Storage features for backend `dir`

Content types	Image formats	Shared	Snapshots	Clones
`images rootdir vztempl iso backup`	`raw qcow2 vmdk subvol`	no	qcow2	qcow2

5.5.4 Examples

Please use the following command to allocate a 4GB image on storage `local`:

```
# pvesm alloc local 100 vm-100-disk10.raw 4G
Formatting '/var/lib/vz/images/100/vm-100-disk10.raw', fmt=raw size ↩
   =4294967296
successfully created 'local:100/vm-100-disk10.raw'
```

Note
The image name must conform to above naming conventions.

The real file system path is shown with:

```
# pvesm path local:100/vm-100-disk10.raw
/var/lib/vz/images/100/vm-100-disk10.raw
```

And you can remove the image with:

```
# pvesm free local:100/vm-100-disk10.raw
```

5.6 NFS Backend

Storage pool type: `nfs`

The NFS backend is based on the directory backend, so it shares most properties. The directory layout and the file naming conventions are the same. The main advantage is that you can directly configure the NFS server properties, so the backend can mount the share automatically. There is no need to modify */etc/fstab*. The backend can also test if the server is online, and provides a method to query the server for exported shares.

5.6.1 Configuration

The backend supports all common storage properties, except the shared flag, which is always set. Additionally, the following properties are used to configure the NFS server:

server
Server IP or DNS name. To avoid DNS lookup delays, it is usually preferrable to use an IP address instead of a DNS name - unless you have a very reliable DNS server, or list the server in the local `/etc/hosts` file.

export
NFS export path (as listed by `pvesm nfsscan`).

You can also set NFS mount options:

path
The local mount point (defaults to */mnt/pve/<STORAGE_ID>/*).

options
NFS mount options (see `man nfs`).

Configuration Example (/etc/pve/storage.cfg)

```
nfs: iso-templates
        path /mnt/pve/iso-templates
        server 10.0.0.10
        export /space/iso-templates
        options vers=3,soft
        content iso,vztmpl
```

Tip
After an NFS request times out, NFS request are retried indefinitely by default. This can lead to unexpected hangs on the client side. For read-only content, it is worth to consider the NFS `soft` option, which limits the number of retries to three.

5.6.2 Storage Features

NFS does not support snapshots, but the backend use `qcow2` features to implement snapshots and cloning.

Table 5.4: Storage features for backend `nfs`

Content types	Image formats	Shared	Snapshots	Clones
`images rootdir vztempl iso backup`	`raw qcow2 vmdk subvol`	yes	qcow2	qcow2

5.6.3 Examples

You can get a list of exported NFS shares with:

```
# pvesm nfsscan <server>
```

5.7 GlusterFS Backend

Storage pool type: `glusterfs`

GlusterFS is a salable network file system. The system uses a modular design, runs on commodity hardware, and can provide a highly available enterprise storage at low costs. Such system is capable of scaling to several petabytes, and can handle thousands of clients.

Note
After a node/brick crash, GlusterFS does a full *rsync* to make sure data is consistent. This can take a very long time with large files, so this backend is not suitable to store large VM images.

5.7.1 Configuration

The backend supports all common storage properties, and adds the following GlusterFS specific options:

`server`
GlusterFS volfile server IP or DNS name.

`server2`
Backup volfile server IP or DNS name.

`volume`
GlusterFS Volume.

`transport`
GlusterFS transport: `tcp`, `unix` or `rdma`

Configuration Example (/etc/pve/storage.cfg)

```
glusterfs: Gluster
        server 10.2.3.4
        server2 10.2.3.5
        volume glustervol
        content images,iso
```

5.7.2 File naming conventions

The directory layout and the file naming conventions are inhertited from the `dir` backend.

5.7.3 Storage Features

The storage provides a file level interface, but no native snapshot/clone implementation.

Table 5.5: Storage features for backend `glusterfs`

Content types	Image formats	Shared	Snapshots	Clones
`images vztempl iso backup`	`raw qcow2 vmdk`	yes	qcow2	qcow2

5.8 Local ZFS Pool Backend

Storage pool type: `zfspool`

This backend allows you to access local ZFS pools (or ZFS filesystems inside such pools).

5.8.1 Configuration

The backend supports the common storage properties `content`, `nodes`, `disable`, and the following ZFS specific properties:

pool
Select the ZFS pool/filesystem. All allocations are done within that pool.

blocksize
Set ZFS blocksize parameter.

sparse
Use ZFS thin-provisioning. A sparse volume is a volume whose reservation is not equal to the volume size.

Configuration Example (/etc/pve/storage.cfg)

```
zfspool: vmdata
        pool tank/vmdata
        content rootdir,images
        sparse
```

5.8.2 File naming conventions

The backend uses the following naming scheme for VM images:

```
vm-<VMID>-<NAME>       // normal VM images
base-<VMID>-<NAME>     // template VM image (read-only)
subvol-<VMID>-<NAME>   // subvolumes (ZFS filesystem for containers)
```

<VMID>
This specifies the owner VM.

<NAME>
This scan be an arbitrary name (`ascii`) without white spaces. The backend uses `disk[N]` as default, where `[N]` is replaced by an integer to make the name unique.

5.8.3 Storage Features

ZFS is probably the most advanced storage type regarding snapshot and cloning. The backend uses ZFS datasets for both VM images (format `raw`) and container data (format `subvol`). ZFS properties are inherited from the parent dataset, so you can simply set defaults on the parent dataset.

Table 5.6: Storage features for backend `zfs`

Content types	Image formats	Shared	Snapshots	Clones
`images rootdir`	`raw subvol`	no	yes	yes

5.8.4 Examples

It is recommended to create and extra ZFS filesystem to store your VM images:

```
# zfs create tank/vmdata
```

To enable compression on that newly allocated filesystem:

```
# zfs set compression=on tank/vmdata
```

You can get a list of available ZFS filesystems with:

```
# pvesm zfsscan
```

5.9 LVM Backend

Storage pool type: `lvm`

LVM is a thin software layer on top of hard disks and partitions. It can be used to split available disk space into smaller logical volumes. LVM is widely used on Linux and makes managing hard drives easier.

Another use case is to put LVM on top of a big iSCSI LUN. That way you can easily manage space on that iSCSI LUN, which would not be possible otherwise, because the iSCSI specification does not define a management interface for space allocation.

5.9.1 Configuration

The LVM backend supports the common storage properties `content`, `nodes`, `disable`, and the following LVM specific properties:

`vgname`
LVM volume group name. This must point to an existing volume group.

`base`
Base volume. This volume is automatically activated before accessing the storage. This is mostly useful when the LVM volume group resides on a remote iSCSI server.

`saferemove`
Zero-out data when removing LVs. When removing a volume, this makes sure that all data gets erased.

`saferemove_throughput`
Wipe throughput (*cstream -t* parameter value).

Configuration Example (/etc/pve/storage.cfg)

```
lvm: myspace
        vgname myspace
        content rootdir,images
```

5.9.2 File naming conventions

The backend use basically the same naming conventions as the ZFS pool backend.

```
vm-<VMID>-<NAME>       // normal VM images
```

5.9.3 Storage Features

LVM is a typical block storage, but this backend does not support snapshot and clones. Unfortunately, normal LVM snapshots are quite inefficient, because they interfere all writes on the whole volume group during snapshot time.

One big advantage is that you can use it on top of a shared storage, for example an iSCSI LUN. The backend itself implement proper cluster wide locking.

Tip
The newer LVM-thin backend allows snapshot and clones, but does not support shared storage.

Table 5.7: Storage features for backend `lvm`

Content types	Image formats	Shared	Snapshots	Clones
`images` `rootdir`	`raw`	possible	no	no

5.9.4 Examples

List available volume groups:

```
# pvesm lvmscan
```

5.10 LVM thin Backend

Storage pool type: `lvmthin`

LVM normally allocates blocks when you create a volume. LVM thin pools instead allocates blocks when they are written. This behaviour is called thin-provisioning, because volumes can be much larger than physically available space.

You can use the normal LVM command line tools to manage and create LVM thin pools (see *man lvmthin* for details). Assuming you already have a LVM volume group called `pve`, the following commands create a new LVM thin pool (size 100G) called `data`:

```
lvcreate -L 100G -n data pve
lvconvert --type thin-pool pve/data
```

5.10.1 Configuration

The LVM thin backend supports the common storage properties `content`, `nodes`, `disable`, and the following LVM specific properties:

`vgname`
LVM volume group name. This must point to an existing volume group.

`thinpool`
The name of the LVM thin pool.

Configuration Example (/etc/pve/storage.cfg)

```
lvmthin: local-lvm
        thinpool data
        vgname pve
        content rootdir,images
```

5.10.2 File naming conventions

The backend use basically the same naming conventions as the ZFS pool backend.

```
vm-<VMID>-<NAME>       // normal VM images
```

5.10.3 Storage Features

LVM thin is a block storage, but fully supports snapshots and clones efficiently. New volumes are automatically initialized with zero.

It must be mentioned that LVM thin pools cannot be shared across multiple nodes, so you can only use them as local storage.

Table 5.8: Storage features for backend `lvmthin`

Content types	Image formats	Shared	Snapshots	Clones
`images` `rootdir`	`raw`	no	yes	yes

5.10.4 Examples

List available LVM thin pools on volume group `pve`:

```
# pvesm lvmthinscan pve
```

5.11 Open-iSCSI initiator

Storage pool type: `iscsi`

iSCSI is a widely employed technology used to connect to storage servers. Almost all storage vendors support iSCSI. There are also open source iSCSI target solutions available, e.g. OpenMediaVault, which is based on Debian.

To use this backend, you need to install the *open-iscsi* package. This is a standard Debian package, but it is not installed by default to save resources.

```
# apt-get install open-iscsi
```

Low-level iscsi management task can be done using the *iscsiadm* tool.

5.11.1 Configuration

The backend supports the common storage properties `content`, `nodes`, `disable`, and the following iSCSI specific properties:

portal
iSCSI portal (IP or DNS name with optional port).

target
iSCSI target.

Configuration Example (/etc/pve/storage.cfg)

```
iscsi: mynas
     portal 10.10.10.1
     target iqn.2006-01.openfiler.com:tsn.dcb5aaaddd
     content none
```

Tip
If you want to use LVM on top of iSCSI, it make sense to set `content none`. That way it is not possible to create VMs using iSCSI LUNs directly.

5.11.2 File naming conventions

The iSCSI protocol does not define an interface to allocate or delete data. Instead, that needs to be done on the target side and is vendor specific. The target simply exports them as numbered LUNs. So Proxmox VE iSCSI volume names just encodes some information about the LUN as seen by the linux kernel.

5.11.3 Storage Features

iSCSI is a block level type storage, and provides no management interface. So it is usually best to export one big LUN, and setup LVM on top of that LUN. You can then use the LVM plugin to manage the storage on that iSCSI LUN.

Table 5.9: Storage features for backend `iscsi`

Content types	Image formats	Shared	Snapshots	Clones
`images none`	`raw`	yes	no	no

5.11.4 Examples

Scan a remote iSCSI portal, and returns a list of possible targets:

```
pvesm iscsiscan -portal <HOST[:PORT]>
```

5.12 User Mode iSCSI Backend

Storage pool type: `iscsidirect`

This backend provides basically the same functionality as the Open-iSCSI backed, but uses a user-level library (package *libiscsi2*) to implement it.

It should be noted that there are no kernel drivers involved, so this can be viewed as performance optimization. But this comes with the drawback that you cannot use LVM on top of such iSCSI LUN. So you need to manage all space allocations at the storage server side.

5.12.1 Configuration

The user mode iSCSI backend uses the same configuration options as the Open-iSCSI backed.

Configuration Example (/etc/pve/storage.cfg)

```
iscsidirect: faststore
     portal 10.10.10.1
     target iqn.2006-01.openfiler.com:tsn.dcb5aaaddd
```

5.12.2 Storage Features

Note
This backend works with VMs only. Containers cannot use this driver.

Table 5.10: Storage features for backend `iscsidirect`

Content types	Image formats	Shared	Snapshots	Clones
`images`	`raw`	yes	no	no

5.13 Ceph RADOS Block Devices (RBD)

Storage pool type: `rbd`

Ceph is a distributed object store and file system designed to provide excellent performance, reliability and scalability. RADOS block devices implement a feature rich block level storage, and you get the following advantages:

- thin provisioning
- resizable volumes
- distributed and redundant (striped over multiple OSDs)
- full snapshot and clone capabilities
- self healing
- no single point of failure
- scalable to the exabyte level
- kernel and unser space implementation available

Note
For smaller deployments, it is also possible to run Ceph services directly on your Proxmox VE nodes. Recent hardware has plenty of CPU power and RAM, so running storage services and VMs on same node is possible.

5.13.1 Configuration

This backend supports the common storage properties `nodes`, `disable`, `content`, and the following `rbd` specific properties:

monhost
List of monitor daemon IPs.

pool
Ceph pool name.

username
RBD user Id.

krbd
Access rbd through krbd kernel module. This is required if you want to use the storage for containers.

Configuration Example (/etc/pve/storage.cfg)

```
rbd: ceph3
        monhost 10.1.1.20 10.1.1.21 10.1.1.22
        pool ceph3
        content images
        username admin
```

Tip
You can use the *rbd* utility to do low-level management tasks.

5.13.2 Authentication

If you use cephx authentication, you need to copy the keyfile from Ceph to Proxmox VE host.

Create the directory */etc/pve/priv/ceph* with

```
mkdir /etc/pve/priv/ceph
```

Then copy the keyring

```
scp <cephserver>:/etc/ceph/ceph.client.admin.keyring /etc/pve/priv/ ↩
   ceph/<STORAGE_ID>.keyring
```

The keyring must be named to match your `<STORAGE_ID>`. Copying the keyring generally requires root privileges.

5.13.3 Storage Features

The `rbd` backend is a block level storage, and implements full snapshot and clone functionality.

Table 5.11: Storage features for backend `rbd`

Content types	Image formats	Shared	Snapshots	Clones
`images` `rootdir`	`raw`	yes	yes	yes

Chapter 6

Qemu/KVM Virtual Machines

Qemu (short form for Quick Emulator) is an opensource hypervisor that emulates a physical computer. From the perspective of the host system where Qemu is running, Qemu is a user program which has access to a number of local resources like partitions, files, network cards which are then passed to an emulated computer which sees them as if they were real devices.

A guest operating system running in the emulated computer accesses these devices, and runs as it were running on real hardware. For instance you can pass an iso image as a parameter to Qemu, and the OS running in the emulated computer will see a real CDROM inserted in a CD drive.

Qemu can emulates a great variety of hardware from ARM to Sparc, but Proxmox VE is only concerned with 32 and 64 bits PC clone emulation, since it represents the overwhelming majority of server hardware. The emulation of PC clones is also one of the fastest due to the availability of processor extensions which greatly speed up Qemu when the emulated architecture is the same as the host architecture.

Note
You may sometimes encounter the term *KVM* (Kernel-based Virtual Machine). It means that Qemu is running with the support of the virtualization processor extensions, via the Linux kvm module. In the context of Proxmox VE *Qemu* and *KVM* can be use interchangeably as Qemu in Proxmox VE will always try to load the kvm module.

Qemu inside Proxmox VE runs as a root process, since this is required to access block and PCI devices.

6.1 Emulated devices and paravirtualized devices

The PC hardware emulated by Qemu includes a mainboard, network controllers, scsi, ide and sata controllers, serial ports (the complete list can be seen in the `kvm(1)` man page) all of them emulated in software. All these devices are the exact software equivalent of existing hardware devices, and if the OS running in the guest has the proper drivers it will use the devices as if it were running on real hardware. This allows Qemu to runs *unmodified* operating systems.

This however has a performance cost, as running in software what was meant to run in hardware involves a lot of extra work for the host CPU. To mitigate this, Qemu can present to the guest operating system

paravirtualized devices, where the guest OS recognizes it is running inside Qemu and cooperates with the hypervisor.

Qemu relies on the virtio virtualization standard, and is thus able to presente paravirtualized virtio devices, which includes a paravirtualized generic disk controller, a paravirtualized network card, a paravirtualized serial port, a paravirtualized SCSI controller, etc ...

It is highly recommended to use the virtio devices whenever you can, as they provide a big performance improvement. Using the virtio generic disk controller versus an emulated IDE controller will double the sequential write throughput, as measured with `bonnie++(8)`. Using the virtio network interface can deliver up to three times the throughput of an emulated Intel E1000 network card, as measured with `iperf(1)`. [1]

6.2 Virtual Machines settings

Generally speaking Proxmox VE tries to choose sane defaults for virtual machines (VM). Make sure you understand the meaning of the settings you change, as it could incur a performance slowdown, or putting your data at risk.

6.2.1 General Settings

General settings of a VM include

- the **Node** : the physical server on which the VM will run
- the **VM ID**: a unique number in this Proxmox VE installation used to identify your VM
- **Name**: a free form text string you can use to describe the VM
- **Resource Pool**: a logical group of VMs

6.2.2 OS Settings

When creating a VM, setting the proper Operating System(OS) allows Proxmox VE to optimize some low level parameters. For instance Windows OS expect the BIOS clock to use the local time, while Unix based OS expect the BIOS clock to have the UTC time.

6.2.3 Hard Disk

Qemu can emulate a number of storage controllers:

- the **IDE** controller, has a design which goes back to the 1984 PC/AT disk controller. Even if this controller has been superseded by more more designs, each and every OS you can think has support for it, making it a great choice if you want to run an OS released before 2003. You can connect up to 4 devices on this controller.

[1] See this benchmark on the KVM wiki http://www.linux-kvm.org/page/Using_VirtIO_NIC

- the **SATA** (Serial ATA) controller, dating from 2003, has a more modern design, allowing higher throughput and a greater number of devices to be connected. You can connect up to 6 devices on this controller.
- the **SCSI** controller, designed in 1985, is commonly found on server grade hardware, and can connect up to 14 storage devices. Proxmox VE emulates by default a LSI 53C895A controller.
  A SCSI controller of type *Virtio* is the recommended setting if you aim for performance and is automatically selected for newly created Linux VMs since Proxmox VE 4.3. Linux distributions have support for this controller since 2012, and FreeBSD since 2014. For Windows OSes, you need to provide an extra iso containing the drivers during the installation.
- The **Virtio** controller, also called virtio-blk to distinguish from the Virtio SCSI controller, is an older type of paravirtualized controller which has been superseded in features by the Virtio SCSI Controller.

On each controller you attach a number of emulated hard disks, which are backed by a file or a block device residing in the configured storage. The choice of a storage type will determine the format of the hard disk image. Storages which present block devices (LVM, ZFS, Ceph) will require the **raw disk image format**, whereas files based storages (Ext4, NFS, GlusterFS) will let you to choose either the **raw disk image format** or the **QEMU image format**.

- the **QEMU image format** is a copy on write format which allows snapshots, and thin provisioning of the disk image.
- the **raw disk image** is a bit-to-bit image of a hard disk, similar to what you would get when executing the `dd` command on a block device in Linux. This format do not support thin provisioning or snapshotting by itself, requiring cooperation from the storage layer for these tasks. It is however 10% faster than the **QEMU image format**. [2]
- the **VMware image format** only makes sense if you intend to import/export the disk image to other hypervisors.

Setting the **Cache** mode of the hard drive will impact how the host system will notify the guest systems of block write completions. The **No cache** default means that the guest system will be notified that a write is complete when each block reaches the physical storage write queue, ignoring the host page cache. This provides a good balance between safety and speed.

If you want the Proxmox VE backup manager to skip a disk when doing a backup of a VM, you can set the **No backup** option on that disk.

If your storage supports *thin provisioning* (see the storage chapter in the Proxmox VE guide), and your VM has a **SCSI** controller you can activate the **Discard** option on the hard disks connected to that controller. With **Discard** enabled, when the filesystem of a VM marks blocks as unused after removing files, the emulated SCSI controller will relay this information to the storage, which will then shrink the disk image accordingly.

IO Thread The option **IO Thread** can only be enabled when using a disk with the **VirtIO** controller, or with the **SCSI** controller, when the emulated controller type is **VirtIO SCSI**. With this enabled, Qemu uses one thread per disk, instead of one thread for all, so it should increase performance when using multiple disks. Note that backups do not currently work with **IO Thread** enabled.

[2] See this benchmark for details http://events.linuxfoundation.org/sites/events/files/slides/-CloudOpen2013_Khoa_Huynh_v3.pdf

6.2.4 CPU

A **CPU socket** is a physical slot on a PC motherboard where you can plug a CPU. This CPU can then contain one or many **cores**, which are independent processing units. Whether you have a single CPU socket with 4 cores, or two CPU sockets with two cores is mostly irrelevant from a performance point of view. However some software is licensed depending on the number of sockets you have in your machine, in that case it makes sense to set the number of of sockets to what the license allows you, and increase the number of cores.
Increasing the number of virtual cpus (cores and sockets) will usually provide a performance improvement though that is heavily dependent on the use of the VM. Multithreaded applications will of course benefit from a large number of virtual cpus, as for each virtual cpu you add, Qemu will create a new thread of execution on the host system. If you're not sure about the workload of your VM, it is usually a safe bet to set the number of **Total cores** to 2.

Note
It is perfectly safe to set the *overall* number of total cores in all your VMs to be greater than the number of of cores you have on your server (ie. 4 VMs with each 4 Total cores running in a 8 core machine is OK) In that case the host system will balance the Qemu execution threads between your server cores just like if you were running a standard multithreaded application. However Proxmox VE will prevent you to allocate on a *single* machine more vcpus than physically available, as this will only bring the performance down due to the cost of context switches.

Qemu can emulate a number different of **CPU types** from 486 to the latest Xeon processors. Each new processor generation adds new features, like hardware assisted 3d rendering, random number generation, memory protection, etc ... Usually you should select for your VM a processor type which closely matches the CPU of the host system, as it means that the host CPU features (also called *CPU flags*) will be available in your VMs. If you want an exact match, you can set the CPU type to **host** in which case the VM will have exactly the same CPU flags as your host system.
This has a downside though. If you want to do a live migration of VMs between different hosts, your VM might end up on a new system with a different CPU type. If the CPU flags passed to the guest are missing, the qemu process will stop. To remedy this Qemu has also its own CPU type **kvm64**, that Proxmox VE uses by defaults. kvm64 is a Pentium 4 look a like CPU type, which has a reduced CPU flags set, but is guaranteed to work everywhere.
In short, if you care about live migration and moving VMs between nodes, leave the kvm64 default. If you don't care about live migration, set the CPU type to host, as in theory this will give your guests maximum performance.

You can also optionally emulate a **NUMA** architecture in your VMs. The basics of the NUMA architecture mean that instead of having a global memory pool available to all your cores, the memory is spread into local banks close to each socket. This can bring speed improvements as the memory bus is not a bottleneck anymore. If your system has a NUMA architecture [3] we recommend to activate the option, as this will allow proper distribution of the VM resources on the host system. This option is also required in Proxmox VE to allow hotplugging of cores and RAM to a VM.

If the NUMA option is used, it is recommended to set the number of sockets to the number of sockets of the host system.

[3] if the command `numactl --hardware | grep available` returns more than one node, then your host system has a NUMA architecture

6.2.5 Memory

For each VM you have the option to set a fixed size memory or asking Proxmox VE to dynamically allocate memory based on the current RAM usage of the host.

When choosing a **fixed size memory** Proxmox VE will simply allocate what you specify to your VM.

When choosing to **automatically allocate memory**, Proxmox VE will make sure that the minimum amount you specified is always available to the VM, and if RAM usage on the host is below 80%, will dynamically add memory to the guest up to the maximum memory specified.
When the host is becoming short on RAM, the VM will then release some memory back to the host, swapping running processes if needed and starting the oom killer in last resort. The passing around of memory between host and guest is done via a special `balloon` kernel driver running inside the guest, which will grab or release memory pages from the host. [4]

When multiple VMs use the autoallocate facility, it is possible to set a **Shares** coefficient which indicates the relative amount of the free host memory that each VM shoud take. Suppose for instance you have four VMs, three of them running a HTTP server and the last one is a database server. To cache more database blocks in the database server RAM, you would like to prioritize the database VM when spare RAM is available. For this you assign a Shares property of 3000 to the database VM, leaving the other VMs to the Shares default setting of 1000. The host server has 32GB of RAM, and is curring using 16GB, leaving 32 * 80/100 - 16 = 9GB RAM to be allocated to the VMs. The database VM will get 9 * 3000 / (3000 + 1000 + 1000 + 1000) = 4.5 GB extra RAM and each HTTP server will get 1/5 GB.

All Linux distributions released after 2010 have the balloon kernel driver included. For Windows OSes, the balloon driver needs to be added manually and can incur a slowdown of the guest, so we don't recommend using it on critical systems.

When allocating RAMs to your VMs, a good rule of thumb is always to leave 1GB of RAM available to the host.

6.2.6 Network Device

Each VM can have many *Network interface controllers* (NIC), of four different types:

- **Intel E1000** is the default, and emulates an Intel Gigabit network card.
- the **VirtIO** paravirtualized NIC should be used if you aim for maximum performance. Like all VirtIO devices, the guest OS should have the proper driver installed.
- the **Realtek 8139** emulates an older 100 MB/s network card, and should only be used when emulating older operating systems (released before 2002)
- the **vmxnet3** is another paravirtualized device, which should only be used when importing a VM from another hypervisor.

Proxmox VE will generate for each NIC a random **MAC address**, so that your VM is addressable on Ethernet networks.

The NIC you added to the VM can follow one of two differents models:

[4] A good explanation of the inner workings of the balloon driver can be found here https://rwmj.wordpress.com/2010/07/-17/virtio-balloon/

- in the default **Bridged mode** each virtual NIC is backed on the host by a *tap device*, (a software loopback device simulating an Ethernet NIC). This tap device is added to a bridge, by default vmbr0 in Proxmox VE. In this mode, VMs have direct access to the Ethernet LAN on which the host is located.
- in the alternative **NAT mode**, each virtual NIC will only communicate with the Qemu user networking stack, where a builting router and DHCP server can provide network access. This built-in DHCP will serve adresses in the private 10.0.2.0/24 range. The NAT mode is much slower than the bridged mode, and should only be used for testing.

You can also skip adding a network device when creating a VM by selecting **No network device**.

Multiqueue If you are using the VirtIO driver, you can optionally activate the **Multiqueue** option. This option allows the guest OS to process networking packets using multiple virtual CPUs, providing an increase in the total number of packets transfered.

When using the VirtIO driver with Proxmox VE, each NIC network queue is passed to the host kernel, where the queue will be processed by a kernel thread spawn by the vhost driver. With this option activated, it is possible to pass *multiple* network queues to the host kernel for each NIC.

When using Multiqueue, it is recommended to set it to a value equal to the number of Total Cores of your guest. You also need to set in the VM the number of multi-purpose channels on each VirtIO NIC with the ethtool command:

```
ethtool -L eth0 combined X
```

where X is the number of the number of vcpus of the VM.

You should note that setting the Multiqueue parameter to a value greater than one will increase the CPU load on the host and guest systems as the traffic increases. We recommend to set this option only when the VM has to process a great number of incoming connections, such as when the VM is running as a router, reverse proxy or a busy HTTP server doing long polling.

6.2.7 USB Passthrough

There are two different types of USB passthrough devices:

- Host USB passtrough
- SPICE USB passthrough

Host USB passthrough works by giving a VM a USB device of the host. This can either be done via the vendor- and product-id, or via the host bus and port.

The vendor/product-id looks like this: **0123:abcd**, where **0123** is the id of the vendor, and **abcd** is the id of the product, meaning two pieces of the same usb device have the same id.

The bus/port looks like this: **1-2.3.4**, where **1** is the bus and **2.3.4** is the port path. This represents the physical ports of your host (depending of the internal order of the usb controllers).

If a device is present in a VM configuration when the VM starts up, but the device is not present in the host, the VM can boot without problems. As soon as the device/port ist available in the host, it gets passed through.

Warning
Using this kind of USB passthrough, means that you cannot move a VM online to another host, since the hardware is only available on the host the VM is currently residing.

The second type of passthrough is SPICE USB passthrough. This is useful if you use a SPICE client which supports it. If you add a SPICE USB port to your VM, you can passthrough a USB device from where your SPICE client is, directly to the VM (for example an input device or hardware dongle).

6.2.8 BIOS and UEFI

In order to properly emulate a computer, QEMU needs to use a firmware. By default QEMU uses **SeaBIOS** for this, which is an open-source, x86 BIOS implementation. SeaBIOS is a good choice for most standard setups.

There are, however, some scenarios in which a BIOS is not a good firmware to boot from, e.g. if you want to do VGA passthrough. [5] In such cases, you should rather use **OVMF**, which is an open-source UEFI implemenation. [6]

If you want to use OVMF, there are several things to consider:

In order to save things like the **boot order**, there needs to be an EFI Disk. This disk will be included in backups and snapshots, and there can only be one.

You can create such a disk with the following command:

```
qm set <vmid> -efidisk0 <storage>:1,format=<format>
```

Where **<storage>** is the storage where you want to have the disk, and **<format>** is a format which the storage supports. Alternatively, you can create such a disk through the web interface with *Add* → *EFI Disk* in the hardware section of a VM.

When using OVMF with a virtual display (without VGA passthrough), you need to set the client resolution in the OVMF menu(which you can reach with a press of the ESC button during boot), or you have to choose SPICE as the display type.

6.3 Managing Virtual Machines with *qm*

qm is the tool to manage Qemu/Kvm virtual machines on Proxmox VE. You can create and destroy virtual machines, and control execution (start/stop/suspend/resume). Besides that, you can use qm to set parameters in the associated config file. It is also possible to create and delete virtual disks.

[5] Alex Williamson has a very good blog entry about this. http://vfio.blogspot.co.at/2014/08/primary-graphics-assignment-without-vga.html

[6] See the OVMF Project http://www.tianocore.org/ovmf/

6.3.1 CLI Usage Examples

Create a new VM with 4 GB IDE disk.

```
qm create 300 -ide0 4 -net0 e1000 -cdrom proxmox-mailgateway_2.1.iso
```

Start the new VM

```
qm start 300
```

Send a shutdown request, then wait until the VM is stopped.

```
qm shutdown 300 && qm wait 300
```

Same as above, but only wait for 40 seconds.

```
qm shutdown 300 && qm wait 300 -timeout 40
```

6.4 Configuration

All configuration files consists of lines in the form

```
PARAMETER: value
```

Configuration files are stored inside the Proxmox cluster file system, and can be accessed at */etc/pve/qemu-server/<VMID>.conf*.

6.4.1 Options

acpi: boolean (default=1)
Enable/disable ACPI.

agent: boolean (default=0)
Enable/disable Qemu GuestAgent.

args: string
Arbitrary arguments passed to kvm, for example:

args: -no-reboot -no-hpet

> **Note**
> this option is for experts only.

autostart: boolean (default=0)
Automatic restart after crash (currently ignored).

balloon: integer (0 -N)
Amount of target RAM for the VM in MB. Using zero disables the ballon driver.

bios: (ovmf | seabios) (default=seabios)
Select BIOS implementation.

boot: [acdn]{1,4} (default=cdn)
Boot on floppy (a), hard disk (c), CD-ROM (d), or network (n).

bootdisk: (ide|sata|scsi|virtio)\d+
Enable booting from specified disk.

cdrom: volume
This is an alias for option -ide2

cores: integer (1 -N) (default=1)
The number of cores per socket.

cpu: [cputype=]<cputype> [,hidden=<1|0>]
Emulated CPU type.

cputype=<cputype> (default=kvm64)
Emulated CPU type.

hidden=boolean (default=0)
Do not identify as a KVM virtual machine.

cpulimit: number (0 -128) (default=0)
Limit of CPU usage.

Note
If the computer has 2 CPUs, it has total of *2* CPU time. Value *0* indicates no CPU limit.

---

cpuunits: integer (0 -500000) (default=1000)
CPU weight for a VM. Argument is used in the kernel fair scheduler. The larger the number is, the more CPU time this VM gets. Number is relative to weights of all the other running VMs.

Note
You can disable fair-scheduler configuration by setting this to 0.

---

description: string
Description for the VM. Only used on the configuration web interface. This is saved as comment inside the configuration file.

freeze: boolean
Freeze CPU at startup (use *c* monitor command to start execution).

hostpci[n]: [host=]<HOSTPCIID[;HOSTPCIID2...]> [,pcie=<1|0>] [,rombar=<1|0>]

Map host PCI devices into guest.

Note
This option allows direct access to host hardware. So it is no longer possible to migrate such machines - use with special care.

Caution
Experimental! User reported problems with this option.

host=<HOSTPCIID[;HOSTPCIID2...]>
Host PCI device pass through. The PCI ID of a host's PCI device or a list of PCI virtual functions of the host. HOSTPCIID syntax is:

bus:dev.func (hexadecimal numbers)

You can us the *lspci* command to list existing PCI devices.

pcie=boolean (default=0)
Choose the PCI-express bus (needs the *q35* machine model).

rombar=boolean (default=1)
Specify whether or not the device's ROM will be visible in the guest's memory map.

x-vga=boolean (default=0)
Enable vfio-vga device support.

hotplug: string (default=network,disk,usb)
Selectively enable hotplug features. This is a comma separated list of hotplug features: *network*, *disk*, *cpu*, *memory* and *usb*. Use *0* to disable hotplug completely. Value *1* is an alias for the default *network,disk,usb*.

ide[n]: [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bps>]

Use volume as IDE hard disk or CD-ROM (n is 0 to 3).

aio=(native | threads)
AIO type to use.

backup=boolean
Whether the drive should be included when making backups.

bps=<bps>
Maximum r/w speed speed in bytes per second.

bps_rd=<bps>
Maximum read speed speed in bytes per second.

bps_wr=<bps>
Maximum write speed speed in bytes per second.

cache=(directsync | none | unsafe | writeback | writethrough)
The drive's cache mode

cyls=integer
Force the drive's physical geometry to have a specific cylinder count.

detect_zeroes=boolean
Controls whether to detect and try to optimize writes of zeroes.

discard=(ignore | on)
Controls whether to pass discard/trim requests to the underlying storage.

file=<volume>
The drive's backing volume.

format=<image format>
The drive's backing file's data format.

heads=integer
Force the drive's physical geometry to have a specific head count.

iops=<iops>
Maximum r/w I/O speed in operations per second.

iops_max=<iops>
Maximum unthrottled r/w I/O pool speed in operations per second.

iops_rd=<iops>
Maximum read I/O speed in operations per second.

iops_rd_max=<iops>
Maximum unthrottled read I/O pool speed in operations per second.

iops_wr=<iops>
Maximum write I/O speed in operations per second.

iops_wr_max=<iops>
Maximum unthrottled write I/O pool speed in operations per second.

mbps=<mbps>
Maximum r/w speed speed in megabytes per second.

mbps_max=<mbps>
Maximum unthrottled r/w pool speed in megabytes per second.

mbps_rd=<mbps>
Maximum read speed speed in megabytes per second.

mbps_rd_max=<mbps>
Maximum unthrottled read pool speed in megabytes per second.

mbps_wr=<mbps>
Maximum write speed speed in megabytes per second.

mbps_wr_max=<mbps>
Maximum unthrottled write pool speed in megabytes per second.

media=(cdrom | disk) (default=disk)
The drive's media type.

model=<model>
The drive's reported model name, url-encoded, up to 40 bytes long.

rerror=(ignore | report | stop)
Read error action.

secs=integer
Force the drive's physical geometry to have a specific sector count.

serial=<serial>
The drive's reported serial number, url-encoded, up to 20 bytes long.

size=<DiskSize>
Disk size. This is purely informational and has no effect.

snapshot=boolean
Whether the drive should be included when making snapshots.

trans=(auto | lba | none)
Force disk geometry bios translation mode.

werror=(enospc | ignore | report | stop)
Write error action.

keyboard: (da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | fr-ca |

Keybord layout for vnc server. Default is read from the */etc/pve/datacenter.conf* configuration file.

kvm: boolean (default=1)
Enable/disable KVM hardware virtualization.

localtime: boolean
Set the real time clock to local time. This is enabled by default if ostype indicates a Microsoft OS.

lock: (backup | migrate | rollback | snapshot)
Lock/unlock the VM.

machine: (pc|pc(-i440fx)?-\d+\.\d+(\.pxe)?|q35|pc-q35-\d+\.\d+(\.pxe)?)

Specific the Qemu machine type.

memory: integer (16 -N) (default=512)
Amount of RAM for the VM in MB. This is the maximum available memory when you use the balloon device.

migrate_downtime: number (0 -N) (default=0.1)
Set maximum tolerated downtime (in seconds) for migrations.

migrate_speed: integer (0 -N) (default=0)
Set maximum speed (in MB/s) for migrations. Value 0 is no limit.

name: string
Set a name for the VM. Only used on the configuration web interface.

net[n]: [model=]<model> [,bridge=<bridge>] [,firewall=<1|0>] [,link_down=<1|

Specify network devices.

bridge=<bridge>
Bridge to attach the network device to. The Proxmox VE standard bridge is called *vmbr0*.

If you do not specify a bridge, we create a kvm user (NATed) network device, which provides DHCP and DNS services. The following addresses are used:

```
10.0.2.2   Gateway
10.0.2.3   DNS Server
10.0.2.4   SMB Server
```

The DHCP server assign addresses to the guest starting from 10.0.2.15.

firewall=boolean
Whether this interface should be protected by the firewall.

link_down=boolean
Whether this interface should be disconnected (like pulling the plug).

macaddr=<XX:XX:XX:XX:XX:XX>
MAC address. That address must be unique withing your network. This is automatically generated if not specified.

model=<model>
Network Card Model. The *virtio* model provides the best performance with very low CPU overhead. If your guest does not support this driver, it is usually best to use *e1000*.

queues=integer (0 -16)
Number of packet queues to be used on the device.

rate=number (0 -N)
Rate limit in mbps (megabytes per second) as floating point number.

tag=integer (1 -4094)
VLAN tag to apply to packets on this interface.

trunks=<vlanid[;vlanid...]>
VLAN trunks to pass through this interface.

numa: boolean (**default=0**)
Enable/disable NUMA.

numa[n]: cpus=<id[-id];...> [,hostnodes=<id[-id];...>] [,memory=<number>] [,

NUMA topology.

cpus=<id[-id];...>
CPUs accessing this NUMA node.

hostnodes=<id[-id];...>
Host NUMA nodes to use.

memory=number
Amount of memory this NUMA node provides.

policy=(bind | interleave | preferred)
NUMA allocation policy.

onboot: boolean (default=0)
Specifies whether a VM will be started during system bootup.

ostype: (l24 | l26 | other | solaris | w2k | w2k3 | w2k8 | win7 | win8 | wvi

Specify guest operating system. This is used to enable special optimization/features for specific operating systems:

other	unspecified OS
wxp	Microsoft Windows XP
w2k	Microsoft Windows 2000
w2k3	Microsoft Windows 2003
w2k8	Microsoft Windows 2008
wvista	Microsoft Windows Vista
win7	Microsoft Windows 7
win8	Microsoft Windows 8/2012
l24	Linux 2.4 Kernel
l26	Linux 2.6/3.X Kernel
solaris	Solaris/OpenSolaris/OpenIndiania kernel

parallel[n]: /dev/parport\d+|/dev/usb/lp\d+
Map host parallel devices (n is 0 to 2).

Note
This option allows direct access to host hardware. So it is no longer possible to migrate such machines - use with special care.

Caution
Experimental! User reported problems with this option.

protection: boolean (default=0)
Sets the protection flag of the VM. This will disable the remove VM and remove disk operations.

reboot: boolean (default=1)
Allow reboot. If set to *0* the VM exit on reboot.

sata[n]: [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bps>

Use volume as SATA hard disk or CD-ROM (n is 0 to 5).

aio=(native | threads)
AIO type to use.

backup=boolean
Whether the drive should be included when making backups.

bps=<bps>
Maximum r/w speed speed in bytes per second.

bps_rd=<bps>
Maximum read speed speed in bytes per second.

bps_wr=<bps>
Maximum write speed speed in bytes per second.

cache=(directsync | none | unsafe | writeback | writethrough)
The drive's cache mode

cyls=integer
Force the drive's physical geometry to have a specific cylinder count.

detect_zeroes=boolean
Controls whether to detect and try to optimize writes of zeroes.

discard=(ignore | on)
Controls whether to pass discard/trim requests to the underlying storage.

file=<volume>
The drive's backing volume.

format=<image format>
The drive's backing file's data format.

heads=integer
Force the drive's physical geometry to have a specific head count.

iops=<iops>
Maximum r/w I/O speed in operations per second.

iops_max=<iops>
Maximum unthrottled r/w I/O pool speed in operations per second.

iops_rd=<iops>
Maximum read I/O speed in operations per second.

iops_rd_max=<iops>
Maximum unthrottled read I/O pool speed in operations per second.

iops_wr=<iops>
Maximum write I/O speed in operations per second.

iops_wr_max=<iops>
Maximum unthrottled write I/O pool speed in operations per second.

mbps=<mbps>
Maximum r/w speed speed in megabytes per second.

mbps_max=<mbps>
Maximum unthrottled r/w pool speed in megabytes per second.

mbps_rd=<mbps>
Maximum read speed speed in megabytes per second.

mbps_rd_max=<mbps>
Maximum unthrottled read pool speed in megabytes per second.

mbps_wr=<mbps>
Maximum write speed speed in megabytes per second.

mbps_wr_max=<mbps>
Maximum unthrottled write pool speed in megabytes per second.

media=(cdrom | disk) (**default=disk**)
The drive's media type.

rerror=(ignore | report | stop)
Read error action.

secs=integer
Force the drive's physical geometry to have a specific sector count.

serial=<serial>
The drive's reported serial number, url-encoded, up to 20 bytes long.

size=<DiskSize>
Disk size. This is purely informational and has no effect.

snapshot=boolean
Whether the drive should be included when making snapshots.

trans=(auto | lba | none)
Force disk geometry bios translation mode.

werror=(enospc | ignore | report | stop)
Write error action.

scsi[n]: [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bps>

Use volume as SCSI hard disk or CD-ROM (n is 0 to 13).

aio=(native | threads)
AIO type to use.

backup=boolean
Whether the drive should be included when making backups.

bps=<bps>
Maximum r/w speed speed in bytes per second.

bps_rd=<bps>
Maximum read speed speed in bytes per second.

bps_wr=<bps>
Maximum write speed speed in bytes per second.

cache=(directsync | none | unsafe | writeback | writethrough)
The drive's cache mode

cyls=integer
Force the drive's physical geometry to have a specific cylinder count.

detect_zeroes=boolean
Controls whether to detect and try to optimize writes of zeroes.

discard=(ignore | on)
Controls whether to pass discard/trim requests to the underlying storage.

file=<volume>
The drive's backing volume.

format=<image format>
The drive's backing file's data format.

heads=integer
Force the drive's physical geometry to have a specific head count.

iops=<iops>
Maximum r/w I/O speed in operations per second.

iops_max=<iops>
Maximum unthrottled r/w I/O pool speed in operations per second.

iops_rd=<iops>
Maximum read I/O speed in operations per second.

iops_rd_max=<iops>
Maximum unthrottled read I/O pool speed in operations per second.

iops_wr=<iops>
Maximum write I/O speed in operations per second.

iops_wr_max=<iops>
Maximum unthrottled write I/O pool speed in operations per second.

iothread=boolean
Whether to use iothreads for this drive

mbps=<mbps>
Maximum r/w speed speed in megabytes per second.

mbps_max=<mbps>
Maximum unthrottled r/w pool speed in megabytes per second.

mbps_rd=<mbps>
Maximum read speed speed in megabytes per second.

mbps_rd_max=<mbps>
Maximum unthrottled read pool speed in megabytes per second.

mbps_wr=<mbps>
Maximum write speed speed in megabytes per second.

mbps_wr_max=<mbps>
Maximum unthrottled write pool speed in megabytes per second.

media=(cdrom | disk) (default=disk)
The drive's media type.

queues=integer (2 -N)
Number of queues.

secs=integer
Force the drive's physical geometry to have a specific sector count.

serial=<serial>
The drive's reported serial number, url-encoded, up to 20 bytes long.

size=<DiskSize>
Disk size. This is purely informational and has no effect.

snapshot=boolean
Whether the drive should be included when making snapshots.

trans=(auto | lba | none)
Force disk geometry bios translation mode.

werror=(enospc | ignore | report | stop)
Write error action.

scsihw: (lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci | virtio-scsi-

SCSI controller model

serial[n]: (/dev/.+|socket)
Create a serial device inside the VM (n is 0 to 3), and pass through a host serial device (i.e. /dev/ttyS0), or create a unix socket on the host side (use *qm terminal* to open a terminal connection).

Note
If you pass through a host serial device, it is no longer possible to migrate such machines - use with special care.

Caution
Experimental! User reported problems with this option.

---

shares: integer (0 -50000) (default=1000)
Amount of memory shares for auto-ballooning. The larger the number is, the more memory this VM gets. Number is relative to weights of all other running VMs. Using zero disables auto-ballooning

smbios1: [family=<string>] [,manufacturer=<string>] [,product=<string>] [,se

Specify SMBIOS type 1 fields.

family=<string>
Set SMBIOS1 family string.

manufacturer=<string>
Set SMBIOS1 manufacturer.

product=<string>
Set SMBIOS1 product ID.

serial=<string>
Set SMBIOS1 serial number.

sku=<string>
Set SMBIOS1 SKU string.

uuid=<UUID>
Set SMBIOS1 UUID.

version=<string>
Set SMBIOS1 version.

smp: integer (1 -N) (default=1)
The number of CPUs. Please use option -sockets instead.

sockets: integer (1 -N) (default=1)
The number of CPU sockets.

startdate: (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (default=now)
Set the initial date of the real time clock. Valid format for date are: *now* or *2006-06-17T16:01:21* or *2006-06-17*.

startup: `[[order=]\d+] [,up=\d+] [,down=\d+] `
Startup and shutdown behavior. Order is a non-negative number defining the general startup order. Shutdown in done with reverse ordering. Additionally you can set the *up* or *down* delay in seconds, which specifies a delay to wait before the next VM is started or stopped.

tablet: boolean (default=1)
Enable/disable the USB tablet device. This device is usually needed to allow absolute mouse positioning with VNC. Else the mouse runs out of sync with normal VNC clients. If you're running lots of console-only guests on one host, you may consider disabling this to save some context switches. This is turned off by default if you use spice (-vga=qxl).

tdf: boolean (default=0)
Enable/disable time drift fix.

template: boolean (default=0)
Enable/disable Template.

unused[n]: string
Reference to unused volumes. This is used internally, and should not be modified manually.

usb[n]: [host=]<HOSTUSBDEVICE|spice> [,usb3=<1|0>]
Configure an USB device (n is 0 to 4).

host=<HOSTUSBDEVICE|spice>
The Host USB device or port or the value *spice*. HOSTUSBDEVICE syntax is:

```
'bus-port(.port)*' (decimal numbers) or
'vendor_id:product_id' (hexadeciaml numbers) or
'spice'
```

You can use the *lsusb -t* command to list existing usb devices.

Note
This option allows direct access to host hardware. So it is no longer possible to migrate such machines - use with special care.

The value *spice* can be used to add a usb redirection devices for spice.

usb3=boolean (default=0)
Specifies whether if given host option is a USB3 device or port (this does currently not work reliably with spice redirection and is then ignored).

vcpus: integer (1 -N) (default=0)
Number of hotplugged vcpus.

vga: (cirrus | qxl | qxl2 | qxl3 | qxl4 | serial0 | serial1 | serial2 | seri

Select the VGA type. If you want to use high resolution modes (>= 1280x1024x16) then you should use the options *std* or *vmware*. Default is *std* for win8/win7/w2k8, and *cirrus* for other OS types. The *qxl* option enables the SPICE display sever. For win* OS you can select how many independent displays you want, Linux guests can add displays them self. You can also run without any graphic card, using a serial device as terminal.

virtio[n]: [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bp

Use volume as VIRTIO hard disk (n is 0 to 15).

aio=(native | threads)
AIO type to use.

backup=boolean
Whether the drive should be included when making backups.

bps=<bps>
Maximum r/w speed speed in bytes per second.

bps_rd=<bps>
Maximum read speed speed in bytes per second.

bps_wr=<bps>
Maximum write speed speed in bytes per second.

cache=(directsync | none | unsafe | writeback | writethrough)
The drive's cache mode

cyls=integer
Force the drive's physical geometry to have a specific cylinder count.

detect_zeroes=boolean
Controls whether to detect and try to optimize writes of zeroes.

discard=(ignore | on)
Controls whether to pass discard/trim requests to the underlying storage.

file=<volume>
The drive's backing volume.

format=<image format>
The drive's backing file's data format.

heads=integer
Force the drive's physical geometry to have a specific head count.

iops=<iops>
Maximum r/w I/O speed in operations per second.

iops_max=<iops>
Maximum unthrottled r/w I/O pool speed in operations per second.

iops_rd=<iops>
Maximum read I/O speed in operations per second.

iops_rd_max=<iops>
Maximum unthrottled read I/O pool speed in operations per second.

iops_wr=<iops>
Maximum write I/O speed in operations per second.

iops_wr_max=<iops>
Maximum unthrottled write I/O pool speed in operations per second.

iothread=boolean
Whether to use iothreads for this drive

mbps=<mbps>
Maximum r/w speed speed in megabytes per second.

mbps_max=<mbps>
Maximum unthrottled r/w pool speed in megabytes per second.

mbps_rd=<mbps>
Maximum read speed speed in megabytes per second.

mbps_rd_max=<mbps>
Maximum unthrottled read pool speed in megabytes per second.

mbps_wr=<mbps>
Maximum write speed speed in megabytes per second.

mbps_wr_max=<mbps>
Maximum unthrottled write pool speed in megabytes per second.

media=(cdrom | disk) (default=disk)
The drive's media type.

rerror=(ignore | report | stop)
Read error action.

secs=integer
Force the drive's physical geometry to have a specific sector count.

serial=<serial>
The drive's reported serial number, url-encoded, up to 20 bytes long.

size=<DiskSize>
Disk size. This is purely informational and has no effect.

snapshot=boolean
Whether the drive should be included when making snapshots.

trans=(auto | lba | none)
Force disk geometry bios translation mode.

werror=(enospc | ignore | report | stop)
Write error action.

watchdog: [[model=]<i6300esb|ib700>] [,action=<reset|shutdown|poweroff|pause

Create a virtual hardware watchdog device. Once enabled (by a guest action), the watchdog must be periodically polled by an agent inside the guest or else the watchdog will reset the guest (or execute the respective action specified)

action=(debug | none | pause | poweroff | reset | shutdown)
The action to perform if after activation the guest fails to poll the watchdog in time.

model=(i6300esb | ib700) (**default=i6300esb**)
Watchdog type to emulate.

6.5 Locks

Online migrations and backups (*vzdump*) set a lock to prevent incompatible concurrent actions on the affected VMs. Sometimes you need to remove such a lock manually (e.g., after a power failure).

```
qm unlock <vmid>
```

Chapter 7

Proxmox Container Toolkit

Containers are a lightweight alternative to fully virtualized VMs. Instead of emulating a complete Operating System (OS), containers simply use the OS of the host they run on. This implies that all containers use the same kernel, and that they can access resources from the host directly.

This is great because containers do not waste CPU power nor memory due to kernel emulation. Container run-time costs are close to zero and usually negligible. But there are also some drawbacks you need to consider:

- You can only run Linux based OS inside containers, i.e. it is not possible to run FreeBSD or MS Windows inside.
- For security reasons, access to host resources needs to be restricted. This is done with AppArmor, SecComp filters and other kernel features. Be prepared that some syscalls are not allowed inside containers.

Proxmox VE uses LXC as underlying container technology. We consider LXC as low-level library, which provides countless options. It would be too difficult to use those tools directly. Instead, we provide a small wrapper called `pct`, the "Proxmox Container Toolkit".

The toolkit is tightly coupled with Proxmox VE. That means that it is aware of the cluster setup, and it can use the same network and storage resources as fully virtualized VMs. You can even use the Proxmox VE firewall, or manage containers using the HA framework.

Our primary goal is to offer an environment as one would get from a VM, but without the additional overhead. We call this "System Containers".

Note
If you want to run micro-containers (with docker, rkt, . . .), it is best to run them inside a VM.

7.1 Security Considerations

Containers use the same kernel as the host, so there is a big attack surface for malicious users. You should consider this fact if you provide containers to totally untrusted people. In general, fully virtualized VMs provide better isolation.

The good news is that LXC uses many kernel security features like AppArmor, CGroups and PID and user namespaces, which makes containers usage quite secure. We distinguish two types of containers:

7.1.1 Privileged containers

Security is done by dropping capabilities, using mandatory access control (AppArmor), SecComp filters and namespaces. The LXC team considers this kind of container as unsafe, and they will not consider new container escape exploits to be security issues worthy of a CVE and quick fix. So you should use this kind of containers only inside a trusted environment, or when no untrusted task is running as root in the container.

7.1.2 Unprivileged containers

This kind of containers use a new kernel feature called user namespaces. The root uid 0 inside the container is mapped to an unprivileged user outside the container. This means that most security issues (container escape, resource abuse, ...) in those containers will affect a random unprivileged user, and so would be a generic kernel security bug rather than an LXC issue. The LXC team thinks unprivileged containers are safe by design.

7.2 Configuration

The */etc/pve/lxc/<CTID>.conf* file stores container configuration, where *<CTID>* is the numeric ID of the given container. Like all other files stored inside */etc/pve/*, they get automatically replicated to all other cluster nodes.

Note
CTIDs < 100 are reserved for internal purposes, and CTIDs need to be unique cluster wide.

Example Container Configuration

```
ostype: debian
arch: amd64
hostname: www
memory: 512
swap: 512
net0: bridge=vmbr0,hwaddr=66:64:66:64:64:36,ip=dhcp,name=eth0,type=veth
rootfs: local:107/vm-107-disk-1.raw,size=7G
```

Those configuration files are simple text files, and you can edit them using a normal text editor (*vi*, *nano*, ...). This is sometimes useful to do small corrections, but keep in mind that you need to restart the container to apply such changes.

For that reason, it is usually better to use the *pct* command to generate and modify those files, or do the whole thing using the GUI. Our toolkit is smart enough to instantaneously apply most changes to running containers. This feature is called "hot plug", and there is no need to restart the container in that case.

7.2.1 File Format

Container configuration files use a simple colon separated key/value format. Each line has the following format:

```
# this is a comment
OPTION: value
```

Blank lines in those files are ignored, and lines starting with a # character are treated as comments and are also ignored.

It is possible to add low-level, LXC style configuration directly, for example:

```
lxc.init_cmd: /sbin/my_own_init
```

or

```
lxc.init_cmd = /sbin/my_own_init
```

Those settings are directly passed to the LXC low-level tools.

7.2.2 Snapshots

When you create a snapshot, *pct* stores the configuration at snapshot time into a separate snapshot section within the same configuration file. For example, after creating a snapshot called *testsnapshot*, your configuration file will look like this:

Container Configuration with Snapshot

```
memory: 512
swap: 512
parent: testsnaphot
...

[testsnaphot]
memory: 512
swap: 512
snaptime: 1457170803
...
```

There are a few snapshot related properties like *parent* and *snaptime*. The *parent* property is used to store the parent/child relationship between snapshots. *snaptime* is the snapshot creation time stamp (unix epoch).

7.2.3 Guest Operating System Configuration

We normally try to detect the operating system type inside the container, and then modify some files inside the container to make them work as expected. Here is a short list of things we do at container startup:

set /etc/hostname
: to set the container name

modify /etc/hosts
to allow lookup of the local hostname

network setup
pass the complete network setup to the container

configure DNS
pass information about DNS servers

adapt the init system
for example, fix the number of spawned getty processes

set the root password
when creating a new container

rewrite ssh_host_keys
so that each container has unique keys

randomize crontab
so that cron does not start at the same time on all containers

Changes made by Proxmox VE are enclosed by comment markers:

```
# --- BEGIN PVE ---
<data>
# --- END PVE ---
```

Those markers will be inserted at a reasonable location in the file. If such a section already exists, it will be updated in place and will not be moved.

Modification of a file can be prevented by adding a `.pve-ignore.` file for it. For instance, if the file `/etc/.pve-ignore.hosts` exists then the `/etc/hosts` file will not be touched. This can be a simple empty file creatd via:

```
# touch /etc/.pve-ignore.hosts
```

Most modifications are OS dependent, so they differ between different distributions and versions. You can completely disable modifications by manually setting the *ostype* to *unmanaged*.

OS type detection is done by testing for certain files inside the container:

Ubuntu
inspect /etc/lsb-release (*DISTRIB_ID=Ubuntu*)

Debian
test /etc/debian_version

Fedora
test /etc/fedora-release

RedHat or CentOS
test /etc/redhat-release

ArchLinux
test /etc/arch-release

Alpine
test /etc/alpine-release

Gentoo
test /etc/gentoo-release

Note
Container start fails if the configured *ostype* differs from the auto detected type.

7.2.4 Options

`arch`: `(amd64 | i386)` (default=`amd64`)
OS architecture type.

`cmode`: `(console | shell | tty)` (default=`tty`)
Console mode. By default, the console command tries to open a connection to one of the available tty devices. By setting cmode to *console* it tries to attach to /dev/console instead. If you set cmode to *shell*, it simply invokes a shell inside the container (no login).

`console`: `boolean` (default=`1`)
Attach a console device (/dev/console) to the container.

`cpulimit`: `number (0 -128)` (default=`0`)
Limit of CPU usage.

Note
If the computer has 2 CPUs, it has a total of *2* CPU time. Value *0* indicates no CPU limit.

`cpuunits`: `integer (0 -500000)` (default=`1024`)
CPU weight for a VM. Argument is used in the kernel fair scheduler. The larger the number is, the more CPU time this VM gets. Number is relative to the weights of all the other running VMs.

Note
You can disable fair-scheduler configuration by setting this to 0.

`description`: `string`
Container description. Only used on the configuration web interface.

`hostname`: `string`
Set a host name for the container.

lock: (backup | migrate | rollback | snapshot)
Lock/unlock the VM.

memory: integer (16 -N) (default=512)
Amount of RAM for the VM in MB.

mp[n]: [volume=]<volume> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>] [,quota=<1|

Use volume as container mount point.

acl=boolean
Explicitly enable or disable ACL support.

backup=boolean
Whether to include the mountpoint in backups.

mp=<Path>
Path to the mountpoint as seen from inside the container.

quota=boolean
Enable user quotas inside the container (not supported with zfs subvolumes)

ro=boolean
Read-only mountpoint (not supported with bind mounts)

size=<DiskSize>
Volume size (read only value).

volume=<volume>
Volume, device or directory to mount into the container.

nameserver: string
Sets DNS server IP address for a container. Create will automatically use the setting from the host if you neither set searchdomain nor nameserver.

net[n]: name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4

Specifies network interfaces for the container.

bridge=<bridge>
Bridge to attach the network device to.

firewall=boolean
Controls whether this interface's firewall rules should be used.

gw=<GatewayIPv4>
Default gateway for IPv4 traffic.

gw6=<GatewayIPv6>
Default gateway for IPv6 traffic.

hwaddr=<XX:XX:XX:XX:XX:XX>
The interface MAC address. This is dynamically allocated by default, but you can set that statically if needed, for example to always have the same link-local IPv6 address. (lxc.network.hwaddr)

ip=<IPv4Format/CIDR>
IPv4 address in CIDR format.

ip6=<IPv6Format/CIDR>
IPv6 address in CIDR format.

mtu=integer (64 -N)
Maximum transfer unit of the interface. (lxc.network.mtu)

name=<string>
Name of the network device as seen from inside the container. (lxc.network.name)

rate=<mbps>
Apply rate limiting to the interface

tag=integer (1 -4094)
VLAN tag for this interface.

trunks=<vlanid[;vlanid...]>
VLAN ids to pass through the interface

type=(veth)
Network interface type.

onboot: boolean (default=0)
Specifies whether a VM will be started during system bootup.

ostype: (alpine | archlinux | centos | debian | fedora | gentoo | opensuse |

OS type. This is used to setup configuration inside the container, and corresponds to lxc setup scripts in /usr/share/lxc/config/<ostype>.common.conf. Value *unmanaged* can be used to skip and OS specific setup.

protection: boolean (default=0)
Sets the protection flag of the container. This will prevent the CT or CT's disk remove/update operation.

rootfs: [volume=]<volume> [,acl=<1|0>] [,quota=<1|0>] [,ro=<1|0>] [,size=<Di

Use volume as container root.

acl=boolean
Explicitly enable or disable ACL support.

quota=boolean
Enable user quotas inside the container (not supported with zfs subvolumes)

ro=boolean
Read-only mountpoint (not supported with bind mounts)

size=<DiskSize>
Volume size (read only value).

volume=<volume>
Volume, device or directory to mount into the container.

searchdomain: string
Sets DNS search domains for a container. Create will automatically use the setting from the host if you neither set searchdomain nor nameserver.

startup: `[[order=]\d+] [,up=\d+] [,down=\d+] `
Startup and shutdown behavior. Order is a non-negative number defining the general startup order. Shutdown in done with reverse ordering. Additionally you can set the *up* or *down* delay in seconds, which specifies a delay to wait before the next VM is started or stopped.

swap: integer (0 -N) (default=512)
Amount of SWAP for the VM in MB.

template: boolean (default=0)
Enable/disable Template.

tty: integer (0 -6) (default=2)
Specify the number of tty available to the container

unprivileged: boolean (default=0)
Makes the container run as unprivileged user. (Should not be modified manually.)

unused[n]: string
Reference to unused volumes. This is used internally, and should not be modified manually.

7.3 Container Images

Container Images, sometimes also referred to as "templates" or "appliances", are *tar* archives which contain everything to run a container. You can think of it as a tidy container backup. Like most modern container toolkits, *pct* uses those images when you create a new container, for example:

```
pct create 999 local:vztmpl/debian-8.0-standard_8.0-1_amd64.tar.gz
```

Proxmox itself ships a set of basic templates for most common operating systems, and you can download them using the *pveam* (short for Proxmox VE Appliance Manager) command line utility. You can also download TurnKey Linux containers using that tool (or the graphical user interface).

Our image repositories contain a list of available images, and there is a cron job run each day to download that list. You can trigger that update manually with:

```
pveam update
```

After that you can view the list of available images using:

```
pveam available
```

You can restrict this large list by specifying the *section* you are interested in, for example basic *system* images:

List available system images

```
# pveam available --section system
system          archlinux-base_2015-24-29-1_x86_64.tar.gz
system          centos-7-default_20160205_amd64.tar.xz
system          debian-6.0-standard_6.0-7_amd64.tar.gz
system          debian-7.0-standard_7.0-3_amd64.tar.gz
```

```
system          debian-8.0-standard_8.0-1_amd64.tar.gz
system          ubuntu-12.04-standard_12.04-1_amd64.tar.gz
system          ubuntu-14.04-standard_14.04-1_amd64.tar.gz
system          ubuntu-15.04-standard_15.04-1_amd64.tar.gz
system          ubuntu-15.10-standard_15.10-1_amd64.tar.gz
```

Before you can use such a template, you need to download them into one of your storages. You can simply use storage *local* for that purpose. For clustered installations, it is preferred to use a shared storage so that all nodes can access those images.

```
pveam download local debian-8.0-standard_8.0-1_amd64.tar.gz
```

You are now ready to create containers using that image, and you can list all downloaded images on storage *local* with:

```
# pveam list local
local:vztmpl/debian-8.0-standard_8.0-1_amd64.tar.gz  190.20MB
```

The above command shows you the full Proxmox VE volume identifiers. They include the storage name, and most other Proxmox VE commands can use them. For examply you can delete that image later with:

```
pveam remove local:vztmpl/debian-8.0-standard_8.0-1_amd64.tar.gz
```

7.4 Container Storage

Traditional containers use a very simple storage model, only allowing a single mount point, the root file system. This was further restricted to specific file system types like *ext4* and *nfs*. Additional mounts are often done by user provided scripts. This turend out to be complex and error prone, so we try to avoid that now.

Our new LXC based container model is more flexible regarding storage. First, you can have more than a single mount point. This allows you to choose a suitable storage for each application. For example, you can use a relatively slow (and thus cheap) storage for the container root file system. Then you can use a second mount point to mount a very fast, distributed storage for your database application.

The second big improvement is that you can use any storage type supported by the Proxmox VE storage library. That means that you can store your containers on local *lvmthin* or *zfs*, shared *iSCSI* storage, or even on distributed storage systems like *ceph*. It also enables us to use advanced storage features like snapshots and clones. *vzdump* can also use the snapshot feature to provide consistent container backups.

Last but not least, you can also mount local devices directly, or mount local directories using bind mounts. That way you can access local storage inside containers with zero overhead. Such bind mounts also provide an easy way to share data between different containers.

7.4.1 Mount Points

The root mount point is configured with the `rootfs` property, and you can configure up to 10 additional mount points. The corresponding options are called `mp0` to `mp9`, and they can contain the following setting:

`rootfs: [volume=]<volume> [,acl=<1|0>] [,quota=<1|0>] [,ro=<1|0>] [,size=<DiskSize>]`

`mp[n]: [volume=]<volume> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>] [,quota=<1|0>] [,ro=<1|0>] [,size=<DiskSize>]`

acl: boolean
Explicitly enable or disable ACL support.

backup: boolean
Whether to include the mountpoint in backups.

mp: <Path>
Path to the mountpoint as seen from inside the container.

quota: boolean
Enable user quotas inside the container (not supported with zfs subvolumes)

ro: boolean
Read-only mountpoint (not supported with bind mounts)

size: <DiskSize>
Volume size (read only value).

volume: <volume>
Volume, device or directory to mount into the container.

Currently there are basically three types of mount points: storage backed mount points, bind mounts and device mounts.

Typical Container rootfs configuration

```
rootfs: thin1:base-100-disk-1,size=8G
```

Storage backed mount points

Storage backed mount points are managed by the Proxmox VE storage subsystem and come in three different flavors:

- Image based: These are raw images containing a single ext4 formatted file system.
- ZFS Subvolumes: These are technically bind mounts, but with managed storage, and thus allow resizing and snapshotting.
- Directories: passing `size=0` triggers a special case where instead of a raw image a directory is created.

Bind mount points

Bind mounts allow you to access arbitrary directories from your Proxmox VE host inside a container. Some potential use cases are:

- Accessing your home directory in the guest
- Accessing an USB device directory in the guest
- Accessing an NFS mount from the host in the guest

Bind mounts are considered to not be managed by the storage subsystem, so you cannot make snapshots or deal with quotas from inside the container. With unprivileged containers you might run into permission problems caused by the user mapping and cannot use ACLs.

Note
The contents of bind mount points are not backed up when using *vzdump*.

Warning
For security reasons, bind mounts should only be established using source directories especially reserved for this purpose, e.g., a directory hierarchy under `/mnt/bindmounts`. Never bind mount system directories like `/`, `/var` or `/etc` into a container - this poses a great security risk.

Note
The bind mount source path must not contain any symlinks.

For example, to make the directory `/mnt/bindmounts/shared` accessible in the container with ID `100` under the path `/shared`, use a configuration line like *mp0: /mnt/bindmounts/shared,mp=/shared* in */etc/pve/lxc/100.conf*. Alternatively, use *pct set 100 -mp0 /mnt/bindmounts/shared,mp=/shared* to achieve the same result.

Device mount points

Device mount points allow to mount block devices of the host directly into the container. Similar to bind mounts, device mounts are not managed by Proxmox VE's storage subsystem, but the `quota` and `acl` options will be honored.

Note
Device mount points should only be used under special circumstances. In most cases a storage backed mount point offers the same performance and a lot more features.

Note
The contents of device mount points are not backed up when using *vzdump*.

7.4.2 FUSE mounts

Warning
Because of existing issues in the Linux kernel's freezer subsystem the usage of FUSE mounts inside a container is strongly advised against, as containers need to be frozen for suspend or snapshot mode backups.

If FUSE mounts cannot be replaced by other mounting mechanisms or storage technologies, it is possible to establish the FUSE mount on the Proxmox host and use a bind mount point to make it accessible inside the container.

7.4.3 Using quotas inside containers

Quotas allow to set limits inside a container for the amount of disk space that each user can use. This only works on ext4 image based storage types and currently does not work with unprivileged containers.

Activating the `quota` option causes the following mount options to be used for a mount point: `usrjquota=aquota.user,grpjquota=aquota.group,jqfmt=vfsv0`

This allows quotas to be used like you would on any other system. You can initialize the `/aquota.user` and `/aquota.group` files by running

```
quotacheck -cmug /
quotaon /
```

and edit the quotas via the `edquota` command. Refer to the documentation of the distribution running inside the container for details.

Note
You need to run the above commands for every mount point by passing the mount point's path instead of just `/`.

7.4.4 Using ACLs inside containers

The standard Posix Access Control Lists are also available inside containers. ACLs allow you to set more detailed file ownership than the traditional user/ group/others model.

7.5 Container Network

You can configure up to 10 network interfaces for a single container. The corresponding options are called *net0* to *net9*, and they can contain the following setting:

```
net[n]: name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<Gateway
IPv4>] [,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:XX>] [,ip=<IPv4Format/
CIDR>] [,ip6=<IPv6Format/CIDR>] [,mtu=<integer>] [,rate=<mbps>] [,tag=<in
teger>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
```

bridge: <bridge>
Bridge to attach the network device to.

firewall: boolean
Controls whether this interface's firewall rules should be used.

gw: <GatewayIPv4>
Default gateway for IPv4 traffic.

gw6: <GatewayIPv6>
Default gateway for IPv6 traffic.

hwaddr: <XX:XX:XX:XX:XX:XX>
The interface MAC address. This is dynamically allocated by default, but you can set that statically if needed, for example to always have the same link-local IPv6 address. (lxc.network.hwaddr)

ip: <IPv4Format/CIDR>
IPv4 address in CIDR format.

ip6: <IPv6Format/CIDR>
IPv6 address in CIDR format.

mtu: integer (64 -N)
Maximum transfer unit of the interface. (lxc.network.mtu)

name: <string>
Name of the network device as seen from inside the container. (lxc.network.name)

rate: <mbps>
Apply rate limiting to the interface

tag: integer (1 -4094)
VLAN tag for this interface.

trunks: <vlanid[;vlanid...]>
VLAN ids to pass through the interface

type: (veth)
Network interface type.

7.6 Backup and Restore

7.6.1 Container Backup

It is possible to use the *vzdump* tool for container backup. Please refer to the *vzdump* manual page for details.

7.6.2 Restoring Container Backups

Restoring container backups made with *vzdump* is possible using the *pct restore* command. By default, *pct restore* will attempt to restore as much of the backed up container configuration as possible. It is possible to override the backed up configuration by manually setting container options on the command line (see the *pct* manual page for details).

Note
pvesm extractconfig can be used to view the backed up configuration contained in a vzdump archive.

There are two basic restore modes, only differing by their handling of mount points:

"Simple" restore mode

If neither the `rootfs` parameter nor any of the optional `mpX` parameters are explicitly set, the mount point configuration from the backed up configuration file is restored using the following steps:

1. Extract mount points and their options from backup
2. Create volumes for storage backed mount points (on storage provided with the `storage` parameter, or default local storage if unset)
3. Extract files from backup archive
4. Add bind and device mount points to restored configuration (limited to root user)

Note
Since bind and device mount points are never backed up, no files are restored in the last step, but only the configuration options. The assumption is that such mount points are either backed up with another mechanism (e.g., NFS space that is bind mounted into many containers), or not intended to be backed up at all.

This simple mode is also used by the container restore operations in the web interface.

"Advanced" restore mode

By setting the `rootfs` parameter (and optionally, any combination of `mpX` parameters), the *pct restore* command is automatically switched into an advanced mode. This advanced mode completely ignores the `rootfs` and `mpX` configuration options contained in the backup archive, and instead only uses the options explicitly provided as parameters.

This mode allows flexible configuration of mount point settings at restore time, for example:

- Set target storages, volume sizes and other options for each mount point individually
- Redistribute backed up files according to new mount point scheme
- Restore to device and/or bind mount points (limited to root user)

7.7 Managing Containers with *pct*

pct is the tool to manage Linux Containers on Proxmox VE. You can create and destroy containers, and control execution (start, stop, migrate, . . .). You can use pct to set parameters in the associated config file, like network configuration or memory limits.

7.7.1 CLI Usage Examples

Create a container based on a Debian template (provided you have already downloaded the template via the webgui)

```
pct create 100 /var/lib/vz/template/cache/debian-8.0-standard_8.0-1 ↩
   _amd64.tar.gz
```

Start container 100

```
pct start 100
```

Start a login session via getty

```
pct console 100
```

Enter the LXC namespace and run a shell as root user

```
pct enter 100
```

Display the configuration

```
pct config 100
```

Add a network interface called eth0, bridged to the host bridge vmbr0, set the address and gateway, while it's running

```
pct set 100 -net0 name=eth0,bridge=vmbr0,ip=192.168.15.147/24,gw ↩
   =192.168.15.1
```

Reduce the memory of the container to 512MB

```
pct set 100 -memory 512
```

7.8 Files

/etc/pve/lxc/<CTID>.conf
: Configuration file for the container <*CTID*>.

7.9 Container Advantages

- Simple, and fully integrated into Proxmox VE. Setup looks similar to a normal VM setup.
  - Storage (ZFS, LVM, NFS, Ceph, . . .)
  - Network
  - Authentification
  - Cluster
- Fast: minimal overhead, as fast as bare metal
- High density (perfect for idle workloads)
- REST API
- Direct hardware access

7.10 Technology Overview

- Integrated into Proxmox VE graphical user interface (GUI)
- LXC (https://linuxcontainers.org/)
- cgmanager for cgroup management
- lxcfs to provive containerized /proc file system
- apparmor
- CRIU: for live migration (planned)
- We use latest available kernels (4.4.X)
- Image based deployment (templates)
- Container setup from host (Network, DNS, Storage, . . .)

Chapter 8

Proxmox VE Firewall

Proxmox VE Firewall provides an easy way to protect your IT infrastructure. You can setup firewall rules for all hosts inside a cluster, or define rules for virtual machines and containers. Features like firewall macros, security groups, IP sets and aliases helps to make that task easier.

While all configuration is stored on the cluster file system, the iptables based firewall runs on each cluster node, and thus provides full isolation between virtual machines. The distributed nature of this system also provides much higher bandwidth than a central firewall solution.

The firewall has full support for IPv4 and IPv6. IPv6 support is fully transparent, and we filter traffic for both protocols by default. So there is no need to maintain a different set of rules for IPv6.

8.1 Zones

The Proxmox VE firewall groups the network into the following logical zones:

Host
: Traffic from/to a cluster node

VM
: Traffic from/to a specific VM

For each zone, you can define firewall rules for incoming and/or outgoing traffic.

8.2 Configuration Files

All firewall related configuration is stored on the proxmox cluster file system. So those files are automatically distributed to all cluster nodes, and the *pve-firewall* service updates the underlying iptables rules automatically on changes.

You can configure anything using the GUI (i.e. Datacenter → Firewall, or on a Node → Firewall), or you can edit the configuration files directly using your preferred editor.

Firewall configuration files contains sections of key-value pairs. Lines beginning with a # and blank lines are considered comments. Sections starts with a header line containing the section name enclosed in *[* and *]*.

8.2.1 Cluster Wide Setup

The cluster wide firewall configuration is stored at:

`/etc/pve/firewall/cluster.fw`

The configuration can contain the following sections:

[OPTIONS]
This is used to set cluster wide firewall options.

`enable: integer (0 -N)`
Enable or disable the firewall cluster wide.

`policy_in: (ACCEPT | DROP | REJECT)`
Input policy.

`policy_out: (ACCEPT | DROP | REJECT)`
Output policy.

[RULES]
This sections contains cluster wide firewall rules for all nodes.

[IPSET <name>]
Cluster wide IP set definitions.

[GROUP <name>]
Cluster wide security group definitions.

[ALIASES]
Cluster wide Alias definitions.

Enabling the Firewall

The firewall is completely disabled by default, so you need to set the enable option here:

```
[OPTIONS]
# enable firewall (cluster wide setting, default is disabled)
enable: 1
```

Important
If you enable the firewall, traffic to all hosts is blocked by default. Only exceptions is WebGUI(8006) and ssh(22) from your local network.

If you want to administrate your Proxmox VE hosts from remote, you need to create rules to allow traffic from those remote IPs to the web GUI (port 8006). You may also want to allow ssh (port 22), and maybe SPICE (port 3128).

Tip
Please open a SSH connection to one of your Proxmox VE hosts before enabling the firewall. That way you still have access to the host if something goes wrong .

To simplify that task, you can instead create an IPSet called *management*, and add all remote IPs there. This creates all required firewall rules to access the GUI from remote.

8.2.2 Host specific Configuration

Host related configuration is read from:

`/etc/pve/nodes/<nodename>/host.fw`

This is useful if you want to overwrite rules from *cluster.fw* config. You can also increase log verbosity, and set netfilter related options. The configuration can contain the following sections:

[OPTIONS]
This is used to set host related firewall options.

enable: boolean
Enable host firewall rules.

log_level_in: (alert | crit | debug | emerg | err | info | nolog | notice |

Log level for incoming traffic.

log_level_out: (alert | crit | debug | emerg | err | info | nolog | notice |

Log level for outgoing traffic.

ndp: boolean
Enable NDP.

nf_conntrack_max: integer (32768 -N)
Maximum number of tracked connections.

nf_conntrack_tcp_timeout_established: integer (7875 -N)
Conntrack established timeout.

nosmurfs: boolean
Enable SMURFS filter.

smurf_log_level: (alert | crit | debug | emerg | err | info | nolog | notice

Log level for SMURFS filter.

tcp_flags_log_level: (alert | crit | debug | emerg | err | info | nolog | no

Log level for illegal tcp flags filter.

tcpflags: boolean
Filter illegal combinations of TCP flags.

[RULES]
This sections contains host specific firewall rules.

8.2.3 VM/Container configuration

VM firewall configuration is read from:

`/etc/pve/firewall/<VMID>.fw`

and contains the following data:

[OPTIONS]
This is used to set VM/Container related firewall options.

dhcp: boolean
Enable DHCP.

enable: boolean
Enable/disable firewall rules.

ipfilter: boolean
Enable default IP filters. This is equivalent to adding an empty ipfilter-net<id> ipset for every interface. Such ipsets implicitly contain sane default restrictions such as restricting IPv6 link local addresses to the one derived from the interface's MAC address. For containers the configured IP addresses will be implicitly added.

log_level_in: (alert | crit | debug | emerg | err | info | nolog | notice |

Log level for incoming traffic.

log_level_out: (alert | crit | debug | emerg | err | info | nolog | notice |

Log level for outgoing traffic.

macfilter: boolean
Enable/disable MAC address filter.

ndp: boolean
Enable NDP.

policy_in: (ACCEPT | DROP | REJECT)
Input policy.

policy_out: (ACCEPT | DROP | REJECT)
Output policy.

radv: boolean
Allow sending Router Advertisement.

[RULES]
This sections contains VM/Container firewall rules.

[IPSET <name>]
IP set definitions.

[ALIASES]
IP Alias definitions.

Enabling the Firewall for VMs and Containers

Each virtual network device has its own firewall enable flag. So you can selectively enable the firewall for each interface. This is required in addition to the general firewall *enable* option.

The firewall requires a special network device setup, so you need to restart the VM/container after enabling the firewall on a network interface.

8.3 Firewall Rules

Firewall rules consists of a direction (`IN` or `OUT`) and an action (`ACCEPT`, `DENY`, `REJECT`). You can also specify a macro name. Macros contain predifined sets of rules and options. Rules can be disabled by prefixing them with `|`.

Firewall rules syntax

```
[RULES]

DIRECTION ACTION [OPTIONS]
|DIRECTION ACTION [OPTIONS] # disabled rule

DIRECTION MACRO(ACTION) [OPTIONS] # use predefined macro
```

The following options can be used to refine rule matches.

`-dest string`
Restrict packet destination address. This can refer to a single IP address, an IP set (*+ipsetname*) or an IP alias definition. You can also specify an address range like *20.34.101.207-201.3.9.99*, or a list of IP addresses and networks (entries are separated by comma). Please do not mix IPv4 and IPv6 addresses inside such lists.

`-dport string`
Restrict TCP/UDP destination port. You can use service names or simple numbers (0-65535), as defined in */etc/services*. Port ranges can be specified with *\d+:\d+*, for example *80:85*, and you can use comma separated list to match several ports or ranges.

`-iface string`
Network interface name. You have to use network configuration key names for VMs and containers (*net\d+*). Host related rules can use arbitrary strings.

-proto string
IP protocol. You can use protocol names (*tcp/udp*) or simple numbers, as defined in */etc/protocols*.

-source string
Restrict packet source address. This can refer to a single IP address, an IP set (*+ipsetname*) or an IP alias definition. You can also specify an address range like *20.34.101.207-201.3.9.99*, or a list of IP addresses and networks (entries are separated by comma). Please do not mix IPv4 and IPv6 addresses inside such lists.

-sport string
Restrict TCP/UDP source port. You can use service names or simple numbers (0-65535), as defined in */etc/services*. Port ranges can be specified with *\d+:\d+*, for example *80:85*, and you can use comma separated list to match several ports or ranges.

Here are some examples:

```
[RULES]
IN SSH(ACCEPT) -i net0
IN SSH(ACCEPT) -i net0 # a comment
IN SSH(ACCEPT) -i net0 -source 192.168.2.192 # only allow SSH from ↩
   192.168.2.192
IN SSH(ACCEPT) -i net0 -source 10.0.0.1-10.0.0.10 # accept SSH for ip range
IN SSH(ACCEPT) -i net0 -source 10.0.0.1,10.0.0.2,10.0.0.3 #accept ssh for ↩
   ip list
IN SSH(ACCEPT) -i net0 -source +mynetgroup # accept ssh for ipset ↩
   mynetgroup
IN SSH(ACCEPT) -i net0 -source myserveralias #accept ssh for alias ↩
   myserveralias

|IN SSH(ACCEPT) -i net0 # disabled rule

IN  DROP # drop all incoming packages
OUT ACCEPT # accept all outgoing packages
```

8.4 Security Groups

A security group is a collection of rules, defined at cluster level, which can be used in all VMs' rules. For example you can define a group named `webserver` with rules to open the http and https ports.

```
# /etc/pve/firewall/cluster.fw

[group webserver]
IN  ACCEPT -p tcp -dport 80
IN  ACCEPT -p tcp -dport 443
```

Then, you can add this group to a VM's firewall

```
# /etc/pve/firewall/<VMID>.fw
```

```
[RULES]
GROUP webserver
```

8.5 IP Aliases

IP Aliases allow you to associate IP addresses of networks with a name. You can then refer to those names:

- inside IP set definitions
- in `source` and `dest` properties of firewall rules

8.5.1 Standard IP alias `local_network`

This alias is automatically defined. Please use the following command to see assigned values:

```
# pve-firewall localnet
local hostname: example
local IP address: 192.168.2.100
network auto detect: 192.168.0.0/20
using detected local_network: 192.168.0.0/20
```

The firewall automatically sets up rules to allow everything needed for cluster communication (corosync, API, SSH) using this alias.

The user can overwrite these values in the cluster.fw alias section. If you use a single host on a public network, it is better to explicitly assign the local IP address

```
#  /etc/pve/firewall/cluster.fw
[ALIASES]
local_network 1.2.3.4 # use the single ip address
```

8.6 IP Sets

IP sets can be used to define groups of networks and hosts. You can refer to them with '+name` in the firewall rules' `source` and `dest` properties.

The following example allows HTTP traffic from the `management` IP set.

```
IN HTTP(ACCEPT) -source +management
```

8.6.1 Standard IP set `management`

This IP set applies only to host firewalls (not VM firewalls). Those ips are allowed to do normal management tasks (PVE GUI, VNC, SPICE, SSH).

The local cluster network is automatically added to this IP set (alias `cluster_network`), to enable inter-host cluster communication. (multicast,ssh,...)

```
# /etc/pve/firewall/cluster.fw

[IPSET management]
192.168.2.10
192.168.2.10/24
```

8.6.2 Standard IP set *blacklist*

Traffic from these ips is dropped by every host's and VM's firewall.

```
# /etc/pve/firewall/cluster.fw

[IPSET blacklist]
77.240.159.182
213.87.123.0/24
```

8.6.3 Standard IP set *ipfilter-net**

These filters belong to a VM's network interface and are mainly used to prevent IP spoofing. If such a set exists for an interface then any outgoing traffic with a source IP not matching its interface's corresponding ipfilter set will be dropped.

For containers with configured IP addresses these sets, if they exist (or are activated via the general `IP Filter` option in the VM's firewall's *options* tab), implicitly contain the associated IP addresses.

For both virtual machines and containers they also implicitly contain the standard MAC-derived IPv6 link-local address in order to allow the neighbor discovery protocol to work.

```
/etc/pve/firewall/<VMID>.fw

[IPSET ipfilter-net0] # only allow specified IPs on net0
192.168.2.10
```

8.7 Services and Commands

The firewall runs two service daemons on each node:

- pvefw-logger: NFLOG daemon (ulogd replacement).

- pve-firewall: updates iptables rules

There is also a CLI command named *pve-firewall*, which can be used to start and stop the firewall service:

```
# pve-firewall start
# pve-firewall stop
```

To get the status use:

```
# pve-firewall status
```

The above command reads and compiles all firewall rules, so you will see warnings if your firewall configuration contains any errors.

If you want to see the generated iptables rules you can use:

```
# iptables-save
```

8.8 Tips and Tricks

8.8.1 How to allow FTP

FTP is an old style protocol which uses port 21 and several other dynamic ports. So you need a rule to accept port 21. In addition, you need to load the *ip_conntrack_ftp* module. So please run:

```
modprobe ip_conntrack_ftp
```

and add `ip_conntrack_ftp` to */etc/modules* (so that it works after a reboot) .

8.8.2 Suricata IPS integration

If you want to use the Suricata IPS (Intrusion Prevention System), it's possible.

Packets will be forwarded to the IPS only after the firewall ACCEPTed them.

Rejected/Dropped firewall packets don't go to the IPS.

Install suricata on proxmox host:

```
# apt-get install suricata
# modprobe nfnetlink_queue
```

Don't forget to add `nfnetlink_queue` to */etc/modules* for next reboot.

Then, enable IPS for a specific VM with:

```
# /etc/pve/firewall/<VMID>.fw

[OPTIONS]
ips: 1
ips_queues: 0
```

`ips_queues` will bind a specific cpu queue for this VM.

Available queues are defined in

```
# /etc/default/suricata
NFQUEUE=0
```

8.8.3 Avoiding link-local addresses on tap and veth devices

With IPv6 enabled by default every interface gets a MAC-derived link local address. However, most devices on a typical Proxmox VE setup are connected to a bridge and so the bridge is the only interface which really needs one.

To disable a link local address on an interface you can set the interface's `disable_ipv6` sysconf variable. Despite the name, this does not prevent IPv6 traffic from passing through the interface when routing or bridging, so the only noticeable effect will be the removal of the link local address.

The easiest method of achieving this setting for all newly started VMs is to set it for the `default` interface configuration and enabling it explicitly on the interfaces which need it. This is also the case for other settings such as `forwarding`, `accept_ra` or `autoconf`.

Here's a possible setup:

```
# /etc/sysconf.d/90-ipv6.conf

net.ipv6.conf.default.forwarding = 0
net.ipv6.conf.default.proxy_ndp = 0
net.ipv6.conf.default.autoconf = 0
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.default.accept_ra = 0

net.ipv6.conf.lo.disable_ipv6 = 0
```

```
# /etc/network/interfaces
(...)
# Dual stack:
iface vmbr0 inet static
    address 1.2.3.4
    netmask 255.255.255.128
    gateway 1.2.3.5
iface vmbr0 inet6 static
    address fc00::31
    netmask 16
    gateway fc00::1
    accept_ra 0
    pre-up echo 0 > /proc/sys/net/ipv6/conf/$IFACE/disable_ipv6

# With IPv6-only 'pre-up' is too early and 'up' is too late.
# Work around this by creating the bridge manually
iface vmbr1 inet manual
    pre-up ip link add $IFACE type bridge
```

```
    up echo 0 > /proc/sys/net/ipv6/conf/$IFACE/disable_ipv6
iface vmbr1 inet6 static
    address fc00:b:3::1
    netmask 96
    bridge_ports none
    bridge_stp off
    bridge_fd 0
    bridge_vlan_aware yes
    accept_ra 0
(...)
```

8.9 Notes on IPv6

The firewall contains a few IPv6 specific options. One thing to note is that IPv6 does not use the ARP protocol anymore, and instead uses NDP (Neighbor Discovery Protocol) which works on IP level and thus needs IP addresses to succeed. For this purpose link-local addresses derived from the interface's MAC address are used. By default the *NDP* option is enabled on both host and VM level to allow neighbor discovery (NDP) packets to be sent and received.

Beside neighbor discovery NDP is also used for a couple of other things, like autoconfiguration and advertising routers.

By default VMs are allowed to send out router solicitation messages (to query for a router), and to receive router advetisement packets. This allows them to use stateless auto configuration. On the other hand VMs cannot advertise themselves as routers unless the *Allow Router Advertisement* (`radv:1`) option is set.

As for the link local addresses required for NDP, there's also an *IP Filter* (`ipfilter:1`) option which can be enabled which has the same effect as adding an `ipfilter-net*` ipset for each of the VM's network interfaces containing the corresponding link local addresses. (See the Standard IP set ipfilter-net* section for details.)

8.10 Ports used by Proxmox VE

- Web interface: 8006
- VNC Web console: 5900-5999
- SPICE proxy: 3128
- sshd (used for cluster actions): 22
- rpcbind: 111
- corosync multicast (if you run a cluster): 5404, 5405 UDP

Chapter 9

User Management

Proxmox VE supports multiple authentication sources, e.g. Microsoft Active Directory, LDAP, Linux PAM or the integrated Proxmox VE authentication server.

By using the role based user- and permission management for all objects (VM´s, storages, nodes, etc.) granular access can be defined.

9.1 Authentication Realms

Proxmox VE stores all user attributes in */etc/pve/user.cfg*. So there must be an entry for each user in that file. The password is not stored, instead you can use configure several realms to verify passwords.

Microsoft Active Directory , LDAP , Linux PAM standard authentication
: You need to create the system users first with *adduser* (e.g. adduser heinz) and possibly the group as well. After that you can create the user on the GUI!

```
useradd heinz
passwd heinz
groupadd watchman
usermod -a -G watchman heinz
```

Proxmox VE authentication server
: This is a unix like password store (*/etc/pve/priv/shadow.cfg*). Password are encrypted using the SHA-256 hash method. Users are allowed to change passwords.

9.2 Terms and Definitions

9.2.1 Users

A Proxmox VE user name consists of two parts: `<userid>@<realm>`. The login screen on the GUI shows them a separate items, but it is internally used as single string.

We store the following attribute for users (*/etc/pve/user.cfg*):

- first name
- last name
- email address
- expiration date
- flag to enable/disable account
- comment

Superuser

The traditional unix superuser account is called *root@pam*. All system mails are forwarded to the email assigned to that account.

9.2.2 Groups

Each user can be member of several groups. Groups are the preferred way to organize access permissions. You should always grant permission to groups instead of using individual users. That way you will get a much shorter access control list which is easier to handle.

9.2.3 Objects and Paths

Access permissions are assigned to objects, such as a virtual machines (*/vms/{vmid}*) or a storage (*/storage/{storeid}*) or a pool of resources (*/pool/{poolname}*). We use filesystem like paths to address those objects. Those paths form a natural tree, and permissions can be inherited down that hierarchy.

9.2.4 Privileges

A privilege is the right to perform a specific action. To simplify management, lists of privileges are grouped into roles, which can then be uses to set permissions.

We currently use the following privileges:

Node / System related privileges

- `Permissions.Modify`: modify access permissions
- `Sys.PowerMgmt`: Node power management (start, stop, reset, shutdown, ...)
- `Sys.Console`: console access to Node
- `Sys.Syslog`: view Syslog
- `Sys.Audit`: view node status/config
- `Sys.Modify`: create/remove/modify node network parameters
- `Group.Allocate`: create/remove/modify groups

- `Pool.Allocate`: create/remove/modify a pool
- `Realm.Allocate`: create/remove/modify authentication realms
- `Realm.AllocateUser`: assign user to a realm
- `User.Modify`: create/remove/modify user access and details.

Virtual machine related privileges

- `VM.Allocate`: create/remove new VM to server inventory
- `VM.Migrate`: migrate VM to alternate server on cluster
- `VM.PowerMgmt`: power management (start, stop, reset, shutdown, ...)
- `VM.Console`: console access to VM
- `VM.Monitor`: access to VM monitor (kvm)
- `VM.Backup`: backup/restore VMs
- `VM.Audit`: view VM config
- `VM.Clone`: clone/copy a VM
- `VM.Config.Disk`: add/modify/delete Disks
- `VM.Config.CDROM`: eject/change CDROM
- `VM.Config.CPU`: modify CPU settings
- `VM.Config.Memory`: modify Memory settings
- `VM.Config.Network`: add/modify/delete Network devices
- `VM.Config.HWType`: modify emulated HW type
- `VM.Config.Options`: modify any other VM configuration
- `VM.Snapshot`: create/remove VM snapshots

Storage related privileges

- `Datastore.Allocate`: create/remove/modify a data store, delete volumes
- `Datastore.AllocateSpace`: allocate space on a datastore
- `Datastore.AllocateTemplate`: allocate/upload templates and iso images
- `Datastore.Audit`: view/browse a datastore

9.2.5 Roles

A role is simply a list of privileges. Proxmox VE comes with a number of predefined roles which satisfies most needs.

- `Administrator`: has all privileges
- `NoAccess`: has no privileges (used to forbid access)
- `PVEAdmin`: can do most things, but miss rights to modify system settings (`Sys.PowerMgmt`, `Sys.Modify`, `Realm.Allocate`).

- PVEAuditor: read only access
- PVEDatastoreAdmin: create and allocate backup space and templates
- PVEDatastoreUser: allocate backup space and view storage
- PVEPoolAdmin: allocate pools
- PVESysAdmin: User ACLs, audit, system console and system logs
- PVETemplateUser: view and clone templates
- PVEUserAdmin: user administration
- PVEVMAdmin: fully administer VMs
- PVEVMUser: view, backup, config CDROM, VM console, VM power management

You can see the whole set of predefined roles on the GUI.

Adding new roles using the CLI:

```
pveum roleadd PVE_Power-only -privs "VM.PowerMgmt VM.Console"
pveum roleadd Sys_Power-only -privs "Sys.PowerMgmt Sys.Console"
```

9.2.6 Permissions

Permissions are the way we control access to objects. In technical terms they are simply a triple containing <path,user,role>. This concept is also known as access control lists. Each permission specifies a subject (user or group) and a role (set of privileges) on a specific path.

When a subject requests an action on an object, the framework looks up the roles assigned to that subject (using the object path). The set of roles defines the granted privileges.

Inheritance

As mentioned earlier, object paths forms a filesystem like tree, and permissions can be inherited down that tree (the propagate flag is set by default). We use the following inheritance rules:

- permission for individual users always overwrite group permission.
- permission for groups apply when the user is member of that group.
- permission set at higher level always overwrites inherited permissions.

What permission do I need?

The required API permissions are documented for each individual method, and can be found at http://pve.proxmox pve-docs/api-viewer/

9.2.7 Pools

Pools can be used to group a set of virtual machines and data stores. You can then simply set permissions on pools (*/pool/{poolid}*), which are inherited to all pool members. This is a great way simplify access control.

9.3 Command Line Tool

Most users will simply use the GUI to manage users. But there is also a full featured command line tool called *pveum* (short for *Proxmox VE User Manager*). I will use that tool in the following examples. Please note that all Proxmox VE command line tools are wrappers around the API, so you can also access those function through the REST API.

Here are some simple usage examples. To show help type:

```
pveum
```

or (to show detailed help about a specific command)

```
pveum help useradd
```

Create a new user:

```
pveum useradd testuser@pve -comment "Just a test"
```

Set or Change the password (not all realms support that):

```
pveum passwd testuser@pve
```

Disable a user:

```
pveum usermod testuser@pve -enable 0
```

Create a new group:

```
pveum groupadd testgroup
```

Create a new role:

```
pveum roleadd PVE_Power-only -privs "VM.PowerMgmt VM.Console"
```

9.4 Real World Examples

9.4.1 Administrator Group

One of the most wanted features was the ability to define a group of users with full administartor rights (without using the root account).

Define the group:

```
pveum groupadd admin -comment "System Administrators"
```

Then add the permission:

```
pveum aclmod / -group admin -role Administrator
```

You can finally add users to the new *admin* group:

```
pveum usermod testuser@pve -group admin
```

9.4.2 Auditors

You can give read only access to users by assigning the `PVEAuditor` role to users or groups.

Example1: Allow user *joe@pve* to see everything

```
pveum aclmod / -user joe@pve -role PVEAuditor
```

Example1: Allow user *joe@pve* to see all virtual machines

```
pveum aclmod /vms -user joe@pve -role PVEAuditor
```

9.4.3 Delegate User Management

If you want to delegate user managenent to user *joe@pve* you can do that with:

```
pveum aclmod /access -user joe@pve -role PVEUserAdmin
```

User *joe@pve* can now add and remove users, change passwords and other user attributes. This is a very powerful role, and you most likely want to limit that to selected realms and groups. The following example allows *joe@pve* to modify users within realm *pve* if they are members of group *customers*:

```
pveum aclmod /access/realm/pve -user joe@pve -role PVEUserAdmin
pveum aclmod /access/groups/customers -user joe@pve -role PVEUserAdmin
```

Note
The user is able to add other users, but only if they are members of group *customers* and within realm *pve*.

9.4.4 Pools

An enterprise is usually structured into several smaller departments, and it is common that you want to assign resources to them and delegate management tasks. A pool is simply a set of virtual machines and data stores. You can create pools on the GUI. After that you can add resources to the pool (VMs, Storage).

You can also assign permissions to the pool. Those permissions are inherited to all pool members.

Lets assume you have a software development department, so we first create a group

```
pveum groupadd developers -comment "Our software developers"
```

Now we create a new user which is a member of that group

```
pveum useradd developer1@pve -group developers -password
```

Note
The -password parameter will prompt you for a password

I assume we already created a pool called *dev-pool* on the GUI. So we can now assign permission to that pool:

```
pveum aclmod /pool/dev-pool/ -group developers -role PVEAdmin
```

Our software developers can now administrate the resources assigned to that pool.

Chapter 10

High Availability

Our modern society depends heavily on information provided by computers over the network. Mobile devices amplified that dependency, because people can access the network any time from anywhere. If you provide such services, it is very important that they are available most of the time.

We can mathematically define the availability as the ratio of (A) the total time a service is capable of being used during a given interval to (B) the length of the interval. It is normally expressed as a percentage of uptime in a given year.

Table 10.1: Availability - Downtime per Year

Availability %	**Downtime per year**
99	3.65 days
99.9	8.76 hours
99.99	52.56 minutes
99.999	5.26 minutes
99.9999	31.5 seconds
99.99999	3.15 seconds

There are several ways to increase availability. The most elegant solution is to rewrite your software, so that you can run it on several host at the same time. The software itself need to have a way to detect errors and do failover. This is relatively easy if you just want to serve read-only web pages. But in general this is complex, and sometimes impossible because you cannot modify the software yourself. The following solutions works without modifying the software:

- Use reliable "server" components

Note
Computer components with same functionality can have varying reliability numbers, depending on the component quality. Most vendors sell components with higher reliability as "server" components - usually at higher price.

- Eliminate single point of failure (redundant components)
  - use an uninterruptible power supply (UPS)
  - use redundant power supplies on the main boards
  - use ECC-RAM
  - use redundant network hardware
  - use RAID for local storage
  - use distributed, redundant storage for VM data
- Reduce downtime
  - rapidly accessible administrators (24/7)
  - availability of spare parts (other nodes in a Proxmox VE cluster)
  - automatic error detection (*ha-manager*)
  - automatic failover (*ha-manager*)

Virtualization environments like Proxmox VE make it much easier to reach high availability because they remove the "hardware" dependency. They also support to setup and use redundant storage and network devices. So if one host fail, you can simply start those services on another host within your cluster.

Even better, Proxmox VE provides a software stack called *ha-manager*, which can do that automatically for you. It is able to automatically detect errors and do automatic failover.

Proxmox VE *ha-manager* works like an "automated" administrator. First, you configure what resources (VMs, containers, ...) it should manage. *ha-manager* then observes correct functionality, and handles service failover to another node in case of errors. *ha-manager* can also handle normal user requests which may start, stop, relocate and migrate a service.

But high availability comes at a price. High quality components are more expensive, and making them redundant duplicates the costs at least. Additional spare parts increase costs further. So you should carefully calculate the benefits, and compare with those additional costs.

Tip

Increasing availability from 99% to 99.9% is relatively simply. But increasing availability from 99.9999% to 99.99999% is very hard and costly. *ha-manager* has typical error detection and failover times of about 2 minutes, so you can get no more than 99.999% availability.

10.1 Requirements

- at least three cluster nodes (to get reliable quorum)
- shared storage for VMs and containers
- hardware redundancy (everywhere)
- hardware watchdog - if not available we fall back to the linux kernel software watchdog (*softdog*)
- optional hardware fencing devices

10.2 Resources

We call the primary management unit handled by *ha-manager* a resource. A resource (also called "service") is uniquely identified by a service ID (SID), which consists of the resource type and an type specific ID, e.g.: *vm:100*. That example would be a resource of type *vm* (virtual machine) with the ID 100.

For now we have two important resources types - virtual machines and containers. One basic idea here is that we can bundle related software into such VM or container, so there is no need to compose one big service from other services, like it was done with *rgmanager*. In general, a HA enabled resource should not depend on other resources.

10.3 How It Works

This section provides an in detail description of the Proxmox VE HA-manager internals. It describes how the CRM and the LRM work together.

To provide High Availability two daemons run on each node:

pve-ha-lrm
: The local resource manager (LRM), it controls the services running on the local node. It reads the requested states for its services from the current manager status file and executes the respective commands.

pve-ha-crm
: The cluster resource manager (CRM), it controls the cluster wide actions of the services, processes the LRM results and includes the state machine which controls the state of each service.

Note
Locks are provided by our distributed configuration file system (pmxcfs). They are used to guarantee that each LRM is active once and working. As a LRM only executes actions when it holds its lock we can mark a failed node as fenced if we can acquire its lock. This lets us then recover any failed HA services securely without any interference from the now unknown failed Node. This all gets supervised by the CRM which holds currently the manager master lock.

10.3.1 Local Resource Manager

The local resource manager (*pve-ha-lrm*) is started as a daemon on boot and waits until the HA cluster is quorate and thus cluster wide locks are working.

It can be in three states:

- **wait for agent lock**: the LRM waits for our exclusive lock. This is also used as idle sate if no service is configured
- **active**: the LRM holds its exclusive lock and has services configured

- **lost agent lock**: the LRM lost its lock, this means a failure happened and quorum was lost.

After the LRM gets in the active state it reads the manager status file in */etc/pve/ha/manager_status* and determines the commands it has to execute for the services it owns. For each command a worker gets started, this workers are running in parallel and are limited to maximal 4 by default. This default setting may be changed through the datacenter configuration key "max_worker". When finished the worker process gets collected and its result saved for the CRM.

Note
The default value of 4 maximal concurrent Workers may be unsuited for a specific setup. For example may 4 live migrations happen at the same time, which can lead to network congestions with slower networks and/or big (memory wise) services. Ensure that also in the worst case no congestion happens and lower the "max_worker" value if needed. In the contrary, if you have a particularly powerful high end setup you may also want to increase it.

Each command requested by the CRM is uniquely identifiable by an UID, when the worker finished its result will be processed and written in the LRM status file */etc/pve/nodes/<nodename>/lrm_status*. There the CRM may collect it and let its state machine - respective the commands output - act on it.

The actions on each service between CRM and LRM are normally always synced. This means that the CRM requests a state uniquely marked by an UID, the LRM then executes this action **one time** and writes back the result, also identifiable by the same UID. This is needed so that the LRM does not executes an outdated command. With the exception of the *stop* and the *error* command, those two do not depend on the result produced and are executed always in the case of the stopped state and once in the case of the error state.

Note
The HA Stack logs every action it makes. This helps to understand what and also why something happens in the cluster. Here its important to see what both daemons, the LRM and the CRM, did. You may use `journalctl -u pve-ha-lrm` on the node(s) where the service is and the same command for the pve-ha-crm on the node which is the current master.

10.3.2 Cluster Resource Manager

The cluster resource manager (*pve-ha-crm*) starts on each node and waits there for the manager lock, which can only be held by one node at a time. The node which successfully acquires the manager lock gets promoted to the CRM master.

It can be in three states:

- **wait for agent lock**: the LRM waits for our exclusive lock. This is also used as idle sate if no service is configured
- **active**: the LRM holds its exclusive lock and has services configured
- **lost agent lock**: the LRM lost its lock, this means a failure happened and quorum was lost.

It main task is to manage the services which are configured to be highly available and try to always enforce them to the wanted state, e.g.: a enabled service will be started if its not running, if it crashes it will be started again. Thus it dictates the LRM the actions it needs to execute.

When an node leaves the cluster quorum, its state changes to unknown. If the current CRM then can secure the failed nodes lock, the services will be *stolen* and restarted on another node.

When a cluster member determines that it is no longer in the cluster quorum, the LRM waits for a new quorum to form. As long as there is no quorum the node cannot reset the watchdog. This will trigger a reboot after the watchdog then times out, this happens after 60 seconds.

10.4 Configuration

The HA stack is well integrated in the Proxmox VE API2. So, for example, HA can be configured via *ha-manager* or the PVE web interface, which both provide an easy to use tool.

The resource configuration file can be located at */etc/pve/ha/resources.cfg* and the group configuration file at */etc/pve/ha/groups.cfg*. Use the provided tools to make changes, there shouldn't be any need to edit them manually.

10.5 Node Power Status

If a node needs maintenance you should migrate and or relocate all services which are required to run always on another node first. After that you can stop the LRM and CRM services. But note that the watchdog triggers if you stop it with active services.

10.6 Package Updates

When updating the ha-manager you should do one node after the other, never all at once for various reasons. First, while we test our software thoughtfully, a bug affecting your specific setup cannot totally be ruled out. Upgrading one node after the other and checking the functionality of each node after finishing the update helps to recover from an eventual problems, while updating all could render you in a broken cluster state and is generally not good practice.

Also, the Proxmox VE HA stack uses a request acknowledge protocol to perform actions between the cluster and the local resource manager. For restarting, the LRM makes a request to the CRM to freeze all its services. This prevents that they get touched by the Cluster during the short time the LRM is restarting. After that the LRM may safely close the watchdog during a restart. Such a restart happens on a update and as already stated a active master CRM is needed to acknowledge the requests from the LRM, if this is not the case the update process can be too long which, in the worst case, may result in a watchdog reset.

10.7 Fencing

10.7.1 What Is Fencing

Fencing secures that on a node failure the dangerous node gets will be rendered unable to do any damage and that no resource runs twice when it gets recovered from the failed node. This is a really important task and one of the base principles to make a system Highly Available.

If a node would not get fenced it would be in an unknown state where it may have still access to shared resources, this is really dangerous! Imagine that every network but the storage one broke, now while not reachable from the public network the VM still runs and writes on the shared storage. If we would not fence the node and just start up this VM on another Node we would get dangerous race conditions, atomicity violations the whole VM could be rendered unusable. The recovery could also simply fail if the storage protects from multiple mounts and thus defeat the purpose of HA.

10.7.2 How Proxmox VE Fences

There are different methods to fence a node, for example fence devices which cut off the power from the node or disable their communication completely.

Those are often quite expensive and bring additional critical components in a system, because if they fail you cannot recover any service.

We thus wanted to integrate a simpler method in the HA Manager first, namely self fencing with watchdogs.

Watchdogs are widely used in critical and dependable systems since the beginning of micro controllers, they are often independent and simple integrated circuit which programs can use to watch them. After opening they need to report periodically. If, for whatever reason, a program becomes unable to do so the watchdogs triggers a reset of the whole server.

Server motherboards often already include such hardware watchdogs, these need to be configured. If no watchdog is available or configured we fall back to the Linux Kernel softdog while still reliable it is not independent of the servers Hardware and thus has a lower reliability then a hardware watchdog.

10.7.3 Configure Hardware Watchdog

By default all watchdog modules are blocked for security reasons as they are like a loaded gun if not correctly initialized. If you have a hardware watchdog available remove its kernel module from the blacklist, load it with insmod and restart the *watchdog-mux* service or reboot the node.

10.7.4 Recover Fenced Services

After a node failed and its fencing was successful we start to recover services to other available nodes and restart them there so that they can provide service again.

The selection of the node on which the services gets recovered is influenced by the users group settings, the currently active nodes and their respective active service count. First we build a set out of the intersection between user selected nodes and available nodes. Then the subset with the highest priority of those nodes

gets chosen as possible nodes for recovery. We select the node with the currently lowest active service count as a new node for the service. That minimizes the possibility of an overload, which else could cause an unresponsive node and as a result a chain reaction of node failures in the cluster.

10.8 Groups

A group is a collection of cluster nodes which a service may be bound to.

10.8.1 Group Settings

nodes
List of group node members where a priority can be given to each node. A service bound to this group will run on the nodes with the highest priority available. If more nodes are in the highest priority class the services will get distributed to those node if not already there. The priorities have a relative meaning only.

restricted
resources bound to this group may only run on nodes defined by the group. If no group node member is available the resource will be placed in the stopped state.

nofailback
the resource won't automatically fail back when a more preferred node (re)joins the cluster.

10.9 Start Failure Policy

The start failure policy comes in effect if a service failed to start on a node once ore more times. It can be used to configure how often a restart should be triggered on the same node and how often a service should be relocated so that it gets a try to be started on another node. The aim of this policy is to circumvent temporary unavailability of shared resources on a specific node. For example, if a shared storage isn't available on a quorate node anymore, e.g. network problems, but still on other nodes, the relocate policy allows then that the service gets started nonetheless.

There are two service start recover policy settings which can be configured specific for each resource.

max_restart
maximal number of tries to restart an failed service on the actual node. The default is set to one.

max_relocate
maximal number of tries to relocate the service to a different node. A relocate only happens after the max_restart value is exceeded on the actual node. The default is set to one.

Note
The relocate count state will only reset to zero when the service had at least one successful start. That means if a service is re-enabled without fixing the error only the restart policy gets repeated.

10.10 Error Recovery

If after all tries the service state could not be recovered it gets placed in an error state. In this state the service won't get touched by the HA stack anymore. To recover from this state you should follow these steps:

- bring the resource back into an safe and consistent state (e.g: killing its process)
- disable the ha resource to place it in an stopped state
- fix the error which led to this failures
- **after** you fixed all errors you may enable the service again

10.11 Service Operations

This are how the basic user-initiated service operations (via *ha-manager*) work.

enable
: the service will be started by the LRM if not already running.

disable
: the service will be stopped by the LRM if running.

migrate/relocate
: the service will be relocated (live) to another node.

remove
: the service will be removed from the HA managed resource list. Its current state will not be touched.

start/stop
: start and stop commands can be issued to the resource specific tools (like *qm* or *pct*), they will forward the request to the *ha-manager* which then will execute the action and set the resulting service state (enabled, disabled).

10.12 Service States

stopped
: Service is stopped (confirmed by LRM), if detected running it will get stopped again.

request_stop
: Service should be stopped. Waiting for confirmation from LRM.

started
: Service is active an LRM should start it ASAP if not already running. If the Service fails and is detected to be not running the LRM restarts it.

fence

Wait for node fencing (service node is not inside quorate cluster partition). As soon as node gets fenced successfully the service will be recovered to another node, if possible.

freeze

Do not touch the service state. We use this state while we reboot a node, or when we restart the LRM daemon.

migrate

Migrate service (live) to other node.

error

Service disabled because of LRM errors. Needs manual intervention.

Chapter 11

Backup and Restore

Backups are a requirements for any sensible IT deployment, and Proxmox VE provides a fully integrated solution, using the capabilities of each storage and each guest system type. This allows the system administrator to fine tune via the `mode` option between consistency of the backups and downtime of the guest system.

Proxmox VE backups are always full backups - containing the VM/CT configuration and all data. Backups can be started via the GUI or via the `vzdump` command line tool.

Backup Storage Before a backup can run, a backup storage must be defined. Refer to the Storage documentation on how to add a storage. A backup storage must be a file level storage, as backups are stored as regular files. In most situations, using a NFS server is a good way to store backups. You can save those backups later to a tape drive, for off-site archiving.

Scheduled Backup Backup jobs can be scheduled so that they are executed automatically on specific days and times, for selectable nodes and guest systems. Configuration of scheduled backups is done at the Datacenter level in the GUI, which will generate a cron entry in /etc/cron.d/vzdump.

11.1 Backup modes

There are several ways to provide consistency (option `mode`), depending on the guest type.

BACKUP MODES FOR VMS:

`stop` mode
: This mode provides the highest consistency of the backup, at the cost of a downtime in the VM operation. It works by executing an orderly shutdown of the VM, and then runs a background Qemu process to backup the VM data. After the backup is complete, the Qemu process resumes the VM to full operation mode if it was previously running.

`suspend` mode
: This mode is provided for compatibility reason, and suspends the VM before calling the `snapshot` mode. Since suspending the VM results in a longer downtime and does not necessarily improve the data consistency, the use of the `snapshot` mode is recommended instead.

`snapshot` mode
: This mode provides the lowest operation downtime, at the cost of a small inconstancy risk. It works by performing a Proxmox VE live backup, in which data blocks are copied while the VM is running. If the guest agent is enabled (`agent: 1`) and running, it calls *guest-fsfreeze-freeze* and *guest-fsfreeze-thaw* to improve consistency.

A technical overview of the Proxmox VE live backup for QemuServer can be found online here.

Note
Proxmox VE live backup provides snapshot-like semantics on any storage type. It does not require that the underlying storage supports snapshots.

BACKUP MODES FOR CONTAINERS:

`stop` mode
: Stop the container for the duration of the backup. This potentially results in a very long downtime.

`suspend` mode
: This mode uses rsync to copy the container data to a temporary location (see option `--tmpdir`). Then the container is suspended and a second rsync copies changed files. After that, the container is started (resumed) again. This results in minimal downtime, but needs additional space to hold the container copy.

  When the container is on a local filesystem and the target storage of the backup is an NFS server, you should set `--tmpdir` to reside on a local filesystem too, as this will result in a many fold performance improvement. Use of a local `tmpdir` is also required if you want to backup a local container using ACLs in suspend mode if the backup storage is an NFS server.

`snapshot` mode
: This mode uses the snapshotting facilities of the underlying storage. First, the container will be suspended to ensure data consistency. A temporary snapshot of the container's volumes will be made and the snapshot content will be archived in a tar file. Finally, the temporary snapshot is deleted again.

Note
`snapshot` mode requires that all backed up volumes are on a storage that supports snapshots. Using the `backup=no` mountpoint option individual volumes can be excluded from the backup (and thus this requirement).

Note
bind and device mountpoints are skipped during backup operations, like volume mountpoints with the backup option disabled.

11.2 Backup File Names

Newer versions of vzdump encode the guest type and the backup time into the filename, for example

`vzdump-lxc-105-2009_10_09-11_04_43.tar`

That way it is possible to store several backup in the same directory. The parameter `maxfiles` can be used to specify the maximum number of backups to keep.

11.3 Restore

The resulting archive files can be restored with the following programs.

`pct restore`
: Container restore utility

`qmrestore`
: QemuServer restore utility

For details see the corresponding manual pages.

11.4 Configuration

Global configuration is stored in */etc/vzdump.conf*. The file uses a simple colon separated key/value format. Each line has the following format:

`OPTION: value`

Blank lines in the file are ignored, and lines starting with a # character are treated as comments and are also ignored. Values from this file are used as default, and can be overwritten on the command line.

We currently support the following options:

`bwlimit: integer (0 -N)` **(default=0)**
: Limit I/O bandwidth (KBytes per second).

`compress: (0 | 1 | gzip | lzo)` **(default=0)**
: Compress dump file.

`dumpdir: string`
: Store resulting files to specified directory.

`exclude-path: string`
: Exclude certain files/directories (shell globs).

`ionice: integer (0 -8)` **(default=7)**
: Set CFQ ionice priority.

lockwait: integer (0 -N) (default=180)
Maximal time to wait for the global lock (minutes).

mailnotification: (always | failure) (default=always)
Specify when to send an email

mailto: string
Comma-separated list of email addresses that should receive email notifications.

maxfiles: integer (1 -N) (default=1)
Maximal number of backup files per guest system.

mode: (snapshot | stop | suspend) (default=snapshot)
Backup mode.

pigz: integer (default=0)
Use pigz instead of gzip when N>0. N=1 uses half of cores, N>1 uses N as thread count.

remove: boolean (default=1)
Remove old backup files if there are more than *maxfiles* backup files.

script: string
Use specified hook script.

stdexcludes: boolean (default=1)
Exclude temporary files and logs.

stopwait: integer (0 -N) (default=10)
Maximal time to wait until a guest system is stopped (minutes).

storage: string
Store resulting file to this storage.

tmpdir: string
Store temporary files to specified directory.

Example vzdump.conf Configuration

```
tmpdir: /mnt/fast_local_disk
storage: my_backup_storage
mode: snapshot
bwlimit: 10000
```

11.5 Hook Scripts

You can specify a hook script with option `--script`. This script is called at various phases of the backup process, with parameters accordingly set. You can find an example in the documentation directory (*vzdump-hook-script.pl*).

11.6 File Exclusions

Note
this option is only available for container backups.

vzdump skips the following files by default (disable with the option `--stdexcludes 0`)

```
/tmp/?*
/var/tmp/?*
/var/run/?*pid
```

You can also manually specify (additional) exclude paths, for example:

```
# vzdump 777 --exclude-path /tmp/ --exclude-path '/var/foo*'
```

(only excludes tmp directories)

Configuration files are also stored inside the backup archive (in `./etc/vzdump/`) and will be correctly restored.

11.7 Examples

Simply dump guest 777 - no snapshot, just archive the guest private area and configuration files to the default dump directory (usually */var/lib/vz/dump/*).

```
# vzdump 777
```

Use rsync and suspend/resume to create a snapshot (minimal downtime).

```
# vzdump 777 --mode suspend
```

Backup all guest systems and send notification mails to root and admin.

```
# vzdump --all --mode suspend --mailto root --mailto admin
```

Use snapshot mode (no downtime) and non-default dump directory.

```
# vzdump 777 --dumpdir /mnt/backup --mode snapshot
```

Backup more than one guest (selectively)

```
# vzdump 101 102 103 --mailto root
```

Backup all guests excluding 101 and 102

```
# vzdump --mode suspend --exclude 101,102
```

Restore a container to a new CT 600

```
# pct restore 600 /mnt/backup/vzdump-lxc-777.tar
```

Restore a QemuServer VM to VM 601

```
# qmrestore /mnt/backup/vzdump-qemu-888.vma 601
```

Clone an existing container 101 to a new container 300 with a 4GB root file system, using pipes

```
# vzdump 101 --stdout | pct restore --rootfs 4 300 -
```

Chapter 12

Important Service Daemons

12.1 Proxmox VE API Daemon

This daemom exposes the whole Proxmox VE API on 127.0.0.1:85. It runs as *root* and has permission to do all priviledged operations.

Note
The daemon listens to a local address only, so you cannot access it from outside. The *pveproxy* daemon exposes the API to the outside world.

12.2 Proxmox VE API Proxy Daemon

This daemon exposes the whole Proxmox VE API on TCP port 8006 using HTTPS. It runs as user *www-data* and has very limited permissions. Operation requiring more permissions are forwarded to the local *pvedaemon*.

Requests targeted for other nodes are automatically forwarded to those nodes. This means that you can manage your whole cluster by connecting to a single Proxmox VE node.

12.2.1 Host based Access Control

It is possible to configure "apache2" like access control lists. Values are read from file */etc/default/pveproxy*. For example:

```
ALLOW_FROM="10.0.0.1-10.0.0.5,192.168.0.0/22"
DENY_FROM="all"
POLICY="allow"
```

IP addresses can be specified using any syntax understood by `Net::IP`. The name *all* is an alias for *0/0*.

The default policy is *allow*.

Match	POLICY=deny	POLICY=allow
Match Allow only	allow	allow
Match Deny only	deny	deny
No match	deny	allow
Match Both Allow & Deny	deny	allow

12.2.2 SSL Cipher Suite

You can define the cipher list in */etc/default/pveproxy*, for example

```
CIPHERS="HIGH:MEDIUM:!aNULL:!MD5"
```

Above is the default. See the ciphers(1) man page from the openssl package for a list of all available options.

12.2.3 Diffie-Hellman Parameters

You can define the used Diffie-Hellman parameters in */etc/default/pveproxy* by setting `DHPARAMS` to the path of a file containing DH parameters in PEM format, for example

```
DHPARAMS="/path/to/dhparams.pem"
```

If this option is not set, the built-in *skip2048* parameters will be used.

Note
DH parameters are only used if a cipher suite utilizing the DH key exchange algorithm is negotiated.

12.2.4 Alternative HTTPS certificate

By default, pveproxy uses the certificate */etc/pve/local/pve-ssl.pem* (and private key */etc/pve/local/pve-ssl.key*) for HTTPS connections. This certificate is signed by the cluster CA certificate, and therefor not trusted by browsers and operating systems by default.

In order to use a different certificate and private key for HTTPS, store the server certificate and any needed intermediate / CA certificates in PEM format in the file */etc/pve/local/pveproxy-ssl.pem* and the associated private key in PEM format without a password in the file */etc/pve/local/pveproxy-ssl.key*.

Warning
Do not replace the automatically generated node certificate files in */etc/pve/local/pve-ssl.pem*/*etc/pve/local/pve-ssl.key* or the cluster CA files in */etc/pve/pve-root-ca.pem*//*etc/pve/priv/pve-root-ca.key*.

12.3 Proxmox VE Status Daemon

This daemom queries the status of VMs, storages and containers at regular intervals. The result is sent to all nodes in the cluster.

12.4 SPICE Proxy Service

SPICE (the Simple Protocol for Independent Computing Environments) is an open remote computing solution, providing client access to remote displays and devices (e.g. keyboard, mouse, audio). The main use case is to get remote access to virtual machines and container.

This daemon listens on TCP port 3128, and implements an HTTP proxy to forward *CONNECT* request from the SPICE client to the correct Proxmox VE VM. It runs as user *www-data* and has very limited permissions.

12.4.1 Host based Access Control

It is possible to configure "apache2" like access control lists. Values are read from file */etc/default/pveproxy*. See *pveproxy* documentation for details.

Chapter 13

Useful Command Line Tools

13.1 Manage CEPH Services on Proxmox VE Nodes

Tool to manage CEPH services on Proxmox VE nodes.

13.2 Subscription Management

This tool is used to handle pve subscriptions.

Chapter 14

Frequently Asked Questions

Note
New FAQs are appended to the bottom of this section.

1. *What distribution is Proxmox VE based on?*

   Proxmox VE is based on Debian GNU/Linux

2. *What license does the Proxmox VE project use?*

   Proxmox VE code is licensed under the GNU Affero General Public License, version 3.

3. *Will Proxmox VE run on a 32bit processor?*

   Proxmox VE works only on 64-bit CPU´s (AMD or Intel). There is no plan for 32-bit for the platform.

   Note
   VMs and Containers can be both 32-bit and/or 64-bit.

4. *Does my CPU support virtualization?*

   To check if your CPU is virtualization compatible, check for the "vmx" or "svm" tag in this command output:

   ```
   egrep '(vmx|svm)' /proc/cpuinfo
   ```

5. *Supported Intel CPUs*

   64-bit processors with Intel Virtualization Technology (Intel VT-x) support. (List of processors with Intel VT and 64-bit)

6. *Supported AMD CPUs*

   64-bit processors with AMD Virtualization Technology (AMD-V) support.

7. *What is a container, CT, VE, Virtual Private Server, VPS?*

   Operating-system-level virtualization is a server-virtualization method where the kernel of an operating system allows for multiple isolated user-space instances, instead of just one. We call such instances containers. As containers use the host's kernel they are limited to Linux guests.

8. *What is a QEMU/KVM guest (or VM)?*

   A QEMU/KVM guest (or VM) is a guest system running virtualized under Proxmox VE using QEMU and the Linux KVM kernel module.

9. *What is QEMU?*

   QEMU is a generic and open source machine emulator and virtualizer. QEMU uses the Linux KVM kernel module to achieve near native performance by executing the guest code directly on the host CPU. It is not limited to Linux guests but allows arbitrary operating systems to run.

10. *How long will my Proxmox VE version be supported?*

    Proxmox VE versions are supported at least as long as the corresponding Debian Version is oldstable. Proxmox VE uses a rolling release model and using the latest stable version is always recommended.

Proxmox VE Version	**Debian Version**	**First Release**	**Debian EOL**	**Proxmox EOL**
Proxmox VE 4.x	Debian 8 (Jessie)	2015-10	2018-05	tba
Proxmox VE 3.x	Debian 7 (Wheezy)	2013-05	2016-04	2017-02
Proxmox VE 2.x	Debian 6 (Squeeze)	2012-04	2014-05	2014-05
Proxmox VE 1.x	Debian 5 (Lenny)	2008-10	2012-03	2013-01

11. *LXC vs LXD vs Proxmox Containers vs Docker*

    LXC is a userspace interface for the Linux kernel containment features. Through a powerful API and simple tools, it lets Linux users easily create and manage system containers. LXC, as well as the former OpenVZ, aims at **system virtualization**, i.e. allows you to run a complete OS inside a container, where you log in as ssh, add users, run apache, etc...

    LXD is building on top of LXC to provide a new, better user experience. Under the hood, LXD uses LXC through *liblxc* and its Go binding to create and manage the containers. It's basically an alternative to LXC's tools and distribution template system with the added features that come from being controllable over the network.

    Proxmox Containers also aims at **system virtualization**, and thus uses LXC as the basis of its own container offer. The Proxmox Container Toolkit is called *pct*, and is tightly coupled with Proxmox VE. That means that it is aware of the cluster setup, and it can use the same network and storage resources as fully virtualized VMs. You can even use the Proxmox VE firewall, create and restore backups, or manage containers using the HA framework. Everything can be controlled over the network using the Proxmox VE API.

    Docker aims at running a **single** application running in a contained environment. Hence you're man-

aging a docker instance from the host with the docker toolkit. It is not recommended to run docker directly on your Proxmox VE host.

Note
You can however perfectly install and use docker inside a Proxmox Qemu VM, and thus getting the benefit of software containerization with the very strong isolation that VMs provide.

Chapter 15

Bibliography

15.1 Books about Proxmox VE

[1] [Ahmed16] Wasim Ahmed. Mastering Proxmox - Second Edition. Packt Publishing, 2016. ISBN 978-1785888243

[2] [Ahmed15] Wasim Ahmed. Proxmox Cookbook. Packt Publishing, 2015. ISBN 978-1783980901

[3] [Cheng14] Simon M.C. Cheng. Proxmox High Availability. Packt Publishing, 2014. ISBN 978-1783980888

[4] [Goldman16] Rik Goldman. Learning Proxmox VE. Packt Publishing, 2016. ISBN 978-1783981786

[5] [Surber16]] Lee R. Surber. Virtualization Complete: Business Basic Edition. Linux Solutions (LRS-TEK), 2016. ASIN B01BBVQZT6

15.2 Books about related Technology

[6] [Hertzog13] Raphaël Hertzog & Roland Mas. The Debian Administrator's Handbook: Debian Jessie from Discovery to Mastery, Freexian, 2013. ISBN 979-1091414050

[7] [Bir96] Kenneth P. Birman. Building Secure and Reliable Network Applications. Manning Publications Co, 1996. ISBN 978-1884777295

[8] [Walsh10] Norman Walsh. DocBook 5: The Definitive Guide. O'Reilly & Associates, 2010. ISBN 978-0596805029

[9] [Richardson07] Leonard Richardson & Sam Ruby. RESTful Web Services. O'Reilly Media, 2007. ISBN 978-0596529260

[10] [Singh15] Karan Singh. Learning Ceph. Packt Publishing, 2015. ISBN 978-1783985623

[11] [Mauerer08] Wolfgang Mauerer. Professional Linux Kernel Architecture. John Wiley & Sons, 2008. ISBN 978-0470343432

[12] [Loshin03] Pete Loshin, IPv6: Theory, Protocol, and Practice, 2nd Edition. Morgan Kaufmann, 2003. ISBN 978-1558608108

[13] [Loeliger12] Jon Loeliger & Matthew McCullough. Version Control with Git: Powerful tools and techniques for collaborative software development. O'Reilly and Associates, 2012. ISBN 978-1449316389

[14] [Kreibich10] Jay A. Kreibich. Using SQLite, O'Reilly and Associates, 2010. ISBN 978-0596521189

15.3 Books about related Topics

[15] [Bessen09] James Bessen & Michael J. Meurer, Patent Failure: How Judges, Bureaucrats, and Lawyers Put Innovators at Risk. Princeton Univ Press, 2009. ISBN 978-0691143217

Appendix A

Command Line Interface

A.1 pvesm - Proxmox VE Storage Manager

pvesm `<COMMAND> [ARGS] [OPTIONS]`

pvesm add `<type> <storage> [OPTIONS]`

Create a new storage.

<type> (dir | drbd | glusterfs | iscsi | iscsidirect | lvm | lvmthin | nfs

Storage type.

<storage> string
The storage identifier.

-authsupported string
Authsupported.

-base string
Base volume. This volume is automatically activated.

-blocksize string
block size

-comstar_hg string
host group for comstar views

-comstar_tg string
target group for comstar views

-content string
Allowed content types.

> **Note**
> the value *rootdir* is used for Containers, and value *images* for VMs.

-disable boolean
Flag to disable the storage.

-export string
NFS export path.

-format string
Default image format.

-iscsiprovider string
iscsi provider

-krbd boolean
Access rbd through krbd kernel module.

-maxfiles integer (0 -N)
Maximal number of backup files per VM. Use *0* for unlimted.

-monhost string
Monitors daemon ips.

-nodes string
List of cluster node names.

-nowritecache boolean
disable write caching on the target

-options string
NFS mount options (see *man nfs*)

-path string
File system path.

-pool string
Pool.

-portal string
iSCSI portal (IP or DNS name with optional port).

-redundancy integer (1 -16) (default=2)
The redundancy count specifies the number of nodes to which the resource should be deployed. It must be at least 1 and at most the number of nodes in the cluster.

-saferemove boolean
Zero-out data when removing LVs.

-saferemove_throughput string
Wipe throughput (cstream -t parameter value).

-server string
Server IP or DNS name.

-server2 string
Backup volfile server IP or DNS name.

> **Note**
> Requires option(s): `server`

-shared boolean
Mark storage as shared.

-sparse boolean
use sparse volumes

-target string
iSCSI target.

-thinpool string
LVM thin pool LV name.

-transport (rdma | tcp | unix)
Gluster transport: tcp or rdma

-username string
RBD Id.

-vgname string
Volume group name.

-volume string
Glusterfs Volume.

pvesm alloc `<storage> <vmid> <filename> <size> [OPTIONS]`

Allocate disk images.

<storage> string
The storage identifier.

<vmid> integer (1 -N)
Specify owner VM

<filename> string
The name of the file to create.

<size> \d+[MG]?
Size in kilobyte (1024 bytes). Optional suffixes *M* (megabyte, 1024K) and *G* (gigabyte, 1024M)

-format (qcow2 | raw | subvol)
no description available

Note
Requires option(s): `size`

pvesm free `<volume> [OPTIONS]`

Delete volume

`<volume>` `string`
Volume identifier

`-storage` `string`
The storage identifier.

pvesm glusterfsscan `<server>`

Scan remote GlusterFS server.

`<server>` `string`
no description available

pvesm help `[<cmd>] [OPTIONS]`

Get help about specified command.

`<cmd>` `string`
Command name

`-verbose` `boolean`
Verbose output format.

pvesm iscsiscan `-portal <string> [OPTIONS]`

Scan remote iSCSI server.

`-portal` `string`
no description available

pvesm list `<storage> [OPTIONS]`

List storage content.

`<storage>` `string`
The storage identifier.

`-content` `string`
Only list content of this type.

`-vmid` `integer (1 -N)`
Only list images for this VM

pvesm lvmscan

List local LVM volume groups.

pvesm lvmthinscan <vg>

List local LVM Thin Pools.

<vg> [a-zA-Z0-9\.\+_][a-zA-Z0-9\.\+_\-]+
no description available

pvesm nfsscan <server>

Scan remote NFS server.

<server> string
no description available

pvesm path <volume>

Get filesystem path for specified volume

<volume> string
Volume identifier

pvesm remove <storage>

Delete storage configuration.

<storage> string
The storage identifier.

pvesm set <storage> [OPTIONS]

Update storage configuration.

<storage> string
The storage identifier.

-blocksize string
block size

-comstar_hg string
host group for comstar views

-comstar_tg string
target group for comstar views

-content string
Allowed content types.

Note
the value *rootdir* is used for Containers, and value *images* for VMs.

-delete string
A list of settings you want to delete.

-digest string
Prevent changes if current configuration file has different SHA1 digest. This can be used to prevent concurrent modifications.

-disable boolean
Flag to disable the storage.

-format string
Default image format.

-krbd boolean
Access rbd through krbd kernel module.

-maxfiles integer (0 -N)
Maximal number of backup files per VM. Use *0* for unlimted.

-nodes string
List of cluster node names.

-nowritecache boolean
disable write caching on the target

-options string
NFS mount options (see *man nfs*)

-pool string
Pool.

-redundancy integer (1 -16) (default=2)
The redundancy count specifies the number of nodes to which the resource should be deployed. It must be at least 1 and at most the number of nodes in the cluster.

-saferemove boolean
Zero-out data when removing LVs.

-saferemove_throughput string
Wipe throughput (cstream -t parameter value).

-server string
Server IP or DNS name.

-server2 string
Backup volfile server IP or DNS name.

Note
Requires option(s): `server`

-shared boolean
Mark storage as shared.

-sparse boolean
use sparse volumes

-transport (rdma | tcp | unix)
Gluster transport: tcp or rdma

-username string
RBD Id.

pvesm status `[OPTIONS]`

Get status for all datastores.

-content string
Only list stores which support this content type.

-enabled boolean (default=0)
Only list stores which are enabled (not disabled in config).

-storage string
Only list status for specified storage

-target string
If target is different to *node*, we only lists shared storages which content is accessible on this *node* and the specified *target* node.

pvesm zfsscan

Scan zfs pool list on local node.

A.2 pvesubscription - Proxmox VE Subscription Manager

pvesubscription `<COMMAND> [ARGS] [OPTIONS]`

pvesubscription get

Read subscription info.

pvesubscription help `[<cmd>] [OPTIONS]`

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

pvesubscription set <key>

Set subscription key.

<key> string
Proxmox VE subscription key

pvesubscription update [OPTIONS]

Update subscription info.

-force boolean (default=0)
Always connect to server, even if we have up to date info inside local cache.

A.3 pveceph - Manage CEPH Services on Proxmox VE Nodes

pveceph <COMMAND> [ARGS] [OPTIONS]

pveceph createmon

Create Ceph Monitor

pveceph createosd <dev> [OPTIONS]

Create OSD

<dev> string
Block device name.

-fstype (btrfs | ext4 | xfs) (default=xfs)
File system type.

-journal_dev string
Block device name for journal.

pveceph createpool <name> [OPTIONS]

Create POOL

<name> string
The name of the pool. It must be unique.

-crush_ruleset integer (0 -32768) (default=0)
The ruleset to use for mapping object placement in the cluster.

-min_size integer (1 -3) (default=1)
Minimum number of replicas per object

-pg_num integer (8 -32768) (default=64)
Number of placement groups.

-size integer (1 -3) (default=2)
Number of replicas per object

pveceph destroymon <monid>

Destroy Ceph monitor.

<monid> integer
Monitor ID

pveceph destroyosd <osdid> [OPTIONS]

Destroy OSD

<osdid> integer
OSD ID

-cleanup boolean (default=0)
If set, we remove partition table entries.

pveceph destroypool <name>

Destroy pool

<name> string
The name of the pool. It must be unique.

pveceph help [<cmd>] [OPTIONS]

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

pveceph init [OPTIONS]

Create initial ceph default configuration and setup symlinks.

-network string
Use specific network for all ceph related traffic

-pg_bits integer (6 -14) (default=6)
Placement group bits, used to specify the default number of placement groups.

Note
osd pool default pg num does not work for default pools.

-size integer (1 -3) (default=2)
Number of replicas per object

pveceph install [OPTIONS]

Install ceph related packages.

-version (hammer)
no description available

pveceph lspools

List all pools.

pveceph purge

Destroy ceph related data and configuration files.

pveceph start [<service>]

Start ceph services.

<service> (mon|mds|osd)\.[A-Za-z0-9]{1,32}
Ceph service name.

pveceph status

Get ceph status.

pveceph stop [<service>]

Stop ceph services.

<service> (mon|mds|osd)\.[A-Za-z0-9]{1,32}
Ceph service name.

A.4 qm - Qemu/KVM Virtual Machine Manager

qm <COMMAND> [ARGS] [OPTIONS]

qm clone <vmid> <newid> [OPTIONS]

Create a copy of virtual machine/template.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<newid> integer (1 -N)
VMID for the clone.

-description string
Description for the new VM.

-format (qcow2 | raw | vmdk)
Target format for file storage.

> **Note**
> Requires option(s): `full`

-full boolean (default=0)
Create a full copy of all disk. This is always done when you clone a normal VM. For VM templates, we try to create a linked clone by default.

-name string
Set a name for the new VM.

-pool string
Add the new VM to the specified pool.

-snapname string
The name of the snapshot.

-storage string
Target storage for full clone.

> **Note**
> Requires option(s): `full`

-target string
Target node. Only allowed if the original VM is on shared storage.

qm config `<vmid> [OPTIONS]`

Get current virtual machine configuration. This does not include pending configuration changes (see *pending* API).

<vmid> integer (1 -N)
The (unique) ID of the VM.

-current boolean (default=0)
Get current values (instead of pending values).

qm create `<vmid> [OPTIONS]`

Create or restore a virtual machine.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-acpi boolean (default=1)
Enable/disable ACPI.

-agent boolean (default=0)
Enable/disable Qemu GuestAgent.

-archive string
The backup file.

-args string
Arbitrary arguments passed to kvm.

-autostart boolean (default=0)
Automatic restart after crash (currently ignored).

-balloon integer (0 -N)
Amount of target RAM for the VM in MB. Using zero disables the ballon driver.

-bios (ovmf | seabios) (default=seabios)
Select BIOS implementation.

-boot [acdn]{1,4} (default=cdn)
Boot on floppy (a), hard disk (c), CD-ROM (d), or network (n).

-bootdisk (ide|sata|scsi|virtio)\d+
Enable booting from specified disk.

-cdrom volume
This is an alias for option -ide2

-cores integer (1 -N) (default=1)
The number of cores per socket.

-cpu [cputype=]<cputype> [,hidden=<1|0>]
Emulated CPU type.

-cpulimit number (0 -128) (default=0)
Limit of CPU usage.

-cpuunits integer (0 -500000) (default=1000)
CPU weight for a VM.

-description string
Description for the VM. Only used on the configuration web interface. This is saved as comment inside the configuration file.

-force boolean
Allow to overwrite existing VM.

Note
Requires option(s): `archive`

-freeze boolean
Freeze CPU at startup (use *c* monitor command to start execution).

-hostpci[n] [host=]<HOSTPCIID[;HOSTPCIID2...]> [,pcie=<1|0>] [,rombar=<1|0>

Map host PCI devices into guest.

-hotplug string (default=network,disk,usb)
Selectively enable hotplug features. This is a comma separated list of hotplug features: *network*, *disk*, *cpu*, *memory* and *usb*. Use *0* to disable hotplug completely. Value *1* is an alias for the default *network,disk,usb*.

-ide[n] [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bps>

Use volume as IDE hard disk or CD-ROM (n is 0 to 3).

-keyboard (da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | fr-ca |

Keybord layout for vnc server. Default is read from the */etc/pve/datacenter.conf* configuration file.

-kvm boolean (default=1)
Enable/disable KVM hardware virtualization.

-localtime boolean
Set the real time clock to local time. This is enabled by default if ostype indicates a Microsoft OS.

-lock (backup | migrate | rollback | snapshot)
Lock/unlock the VM.

-machine (pc|pc(-i440fx)?-\d+\.\d+(\.pxe)?|q35|pc-q35-\d+\.\d+(\.pxe)?)

Specific the Qemu machine type.

-memory integer (16 -N) (default=512)
Amount of RAM for the VM in MB. This is the maximum available memory when you use the balloon device.

-migrate_downtime number (0 -N) (default=0.1)
Set maximum tolerated downtime (in seconds) for migrations.

-migrate_speed integer (0 -N) (default=0)
Set maximum speed (in MB/s) for migrations. Value 0 is no limit.

-name string
Set a name for the VM. Only used on the configuration web interface.

-net[n] [model=]<model> [,bridge=<bridge>] [,firewall=<1|0>] [,link_down=<1

Specify network devices.

-numa boolean (default=0)
Enable/disable NUMA.

-numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>] [,memory=<number>] [

NUMA topology.

-onboot boolean (default=0)
Specifies whether a VM will be started during system bootup.

-ostype (l24 | l26 | other | solaris | w2k | w2k3 | w2k8 | win7 | win8 | wv

Specify guest operating system.

-parallel[n] /dev/parport\d+|/dev/usb/lp\d+
Map host parallel devices (n is 0 to 2).

-pool string
Add the VM to the specified pool.

-protection boolean (default=0)
Sets the protection flag of the VM. This will disable the remove VM and remove disk operations.

-reboot boolean (default=1)
Allow reboot. If set to *0* the VM exit on reboot.

-sata[n] [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bps

Use volume as SATA hard disk or CD-ROM (n is 0 to 5).

-scsi[n] [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bps

Use volume as SCSI hard disk or CD-ROM (n is 0 to 13).

-scsihw (lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci | virtio-scsi

SCSI controller model

-serial[n] (/dev/.+|socket)
Create a serial device inside the VM (n is 0 to 3)

-shares integer (0 -50000) (default=1000)
Amount of memory shares for auto-ballooning. The larger the number is, the more memory this VM gets. Number is relative to weights of all other running VMs. Using zero disables auto-ballooning

-smbios1 [family=<string>] [,manufacturer=<string>] [,product=<string>] [,s

Specify SMBIOS type 1 fields.

-smp integer (1 -N) (default=1)
The number of CPUs. Please use option -sockets instead.

-sockets integer (1 -N) (default=1)
The number of CPU sockets.

-startdate (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (default=now)
Set the initial date of the real time clock. Valid format for date are: *now* or *2006-06-17T16:01:21* or *2006-06-17*.

-startup `[[order=]\d+] [,up=\d+] [,down=\d+] `
Startup and shutdown behavior. Order is a non-negative number defining the general startup order. Shutdown in done with reverse ordering. Additionally you can set the *up* or *down* delay in seconds, which specifies a delay to wait before the next VM is started or stopped.

-storage string
Default storage.

-tablet boolean (default=1)
Enable/disable the USB tablet device.

-tdf boolean (default=0)
Enable/disable time drift fix.

-template boolean (default=0)
Enable/disable Template.

-unique boolean
Assign a unique random ethernet address.

> **Note**
> Requires option(s): `archive`

-unused[n] string
Reference to unused volumes. This is used internally, and should not be modified manually.

-usb[n] [host=]<HOSTUSBDEVICE|spice> [,usb3=<1|0>]
Configure an USB device (n is 0 to 4).

-vcpus integer (1 -N) (default=0)
Number of hotplugged vcpus.

-vga (cirrus | qxl | qxl2 | qxl3 | qxl4 | serial0 | serial1 | serial2 | ser

Select the VGA type.

-virtio[n] [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<b

Use volume as VIRTIO hard disk (n is 0 to 15).

-watchdog [[model=]<i6300esb|ib700>] [,action=<reset|shutdown|poweroff|paus

Create a virtual hardware watchdog device.

qm delsnapshot `<vmid> <snapname> [OPTIONS]`

Delete a VM snapshot.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<snapname> string
The name of the snapshot.

-force boolean
For removal from config file, even if removing disk snapshots fails.

qm destroy `<vmid> [OPTIONS]`

Destroy the vm (also delete all used/owned volumes).

<vmid> integer (1 -N)
The (unique) ID of the VM.

-skiplock boolean
Ignore locks - only root is allowed to use this option.

qm help `[<cmd>] [OPTIONS]`

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

qm list `[OPTIONS]`

Virtual machine index (per node).

-full boolean
Determine the full status of active VMs.

qm migrate `<vmid> <target> [OPTIONS]`

Migrate virtual machine. Creates a new migration task.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<target> string
Target node.

-force boolean
Allow to migrate VMs which use local devices. Only root may use this option.

-online boolean
Use online/live migration.

qm monitor <vmid>

Enter Qemu Monitor interface.

<vmid> integer (1 -N)
The (unique) ID of the VM.

qm move_disk <vmid> <disk> <storage> [OPTIONS]

Move volume to different storage.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<disk> (ide0 | ide1 | ide2 | ide3 | sata0 | sata1 | sata2 | sata3 | sata4 |

The disk you want to move.

<storage> string
Target storage.

-delete boolean (default=0)
Delete the original disk after successful copy. By default the original disk is kept as unused disk.

-digest string
Prevent changes if current configuration file has different SHA1 digest. This can be used to prevent concurrent modifications.

-format (qcow2 | raw | vmdk)
Target Format.

qm mtunnel

Used by qmigrate - do not use manually.

qm pending <vmid>

Get virtual machine configuration, including pending changes.

<vmid> integer (1 -N)
The (unique) ID of the VM.

qm rescan [OPTIONS]

Rescan all storages and update disk sizes and unused disk images.

-vmid integer (1 -N)
The (unique) ID of the VM.

qm reset <vmid> [OPTIONS]

Reset virtual machine.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-skiplock boolean
Ignore locks - only root is allowed to use this option.

qm resize <vmid> <disk> <size> [OPTIONS]

Extend volume size.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<disk> (ide0 | ide1 | ide2 | ide3 | sata0 | sata1 | sata2 | sata3 | sata4 |

The disk you want to resize.

<size> \+?\d+(\.\d+)?[KMGT]?
The new size. With the + sign the value is added to the actual size of the volume and without it, the value is taken as an absolute one. Shrinking disk size is not supported.

-digest string
Prevent changes if current configuration file has different SHA1 digest. This can be used to prevent concurrent modifications.

-skiplock boolean
Ignore locks - only root is allowed to use this option.

qm resume <vmid> [OPTIONS]

Resume virtual machine.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-nocheck boolean
no description available

-skiplock boolean
Ignore locks - only root is allowed to use this option.

qm rollback <vmid> <snapname>

Rollback VM state to specified snapshot.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<snapname> string
The name of the snapshot.

qm sendkey <vmid> <key> [OPTIONS]

Send key event to virtual machine.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<key> string
The key (qemu monitor encoding).

-skiplock boolean
Ignore locks - only root is allowed to use this option.

qm set <vmid> [OPTIONS]

Set virtual machine options (synchrounous API) - You should consider using the POST method instead for any actions involving hotplug or storage allocation.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-acpi boolean (default=1)
Enable/disable ACPI.

-agent boolean (default=0)
Enable/disable Qemu GuestAgent.

-args string
Arbitrary arguments passed to kvm.

-autostart boolean (default=0)
Automatic restart after crash (currently ignored).

-balloon integer (0 -N)
Amount of target RAM for the VM in MB. Using zero disables the ballon driver.

-bios (ovmf | seabios) (default=seabios)
Select BIOS implementation.

-boot [acdn]{1,4} (default=cdn)
Boot on floppy (a), hard disk (c), CD-ROM (d), or network (n).

-bootdisk (ide|sata|scsi|virtio)\d+
Enable booting from specified disk.

-cdrom volume
This is an alias for option -ide2

-cores integer (1 -N) (default=1)
The number of cores per socket.

-cpu [cputype=]<cputype> [,hidden=<1|0>]
Emulated CPU type.

-cpulimit number (0 -128) (default=0)
Limit of CPU usage.

-cpuunits integer (0 -500000) (default=1000)
CPU weight for a VM.

-delete string
A list of settings you want to delete.

-description string
Description for the VM. Only used on the configuration web interface. This is saved as comment inside the configuration file.

-digest string
Prevent changes if current configuration file has different SHA1 digest. This can be used to prevent concurrent modifications.

-force boolean
Force physical removal. Without this, we simple remove the disk from the config file and create an additional configuration entry called *unused[n]*, which contains the volume ID. Unlink of unused[n] always cause physical removal.

Note
Requires option(s): `delete`

-freeze boolean
Freeze CPU at startup (use *c* monitor command to start execution).

-hostpci[n] [host=]<HOSTPCIID[;HOSTPCIID2...]> [,pcie=<1|0>] [,rombar=<1|0>

Map host PCI devices into guest.

-hotplug string (default=network,disk,usb)
Selectively enable hotplug features. This is a comma separated list of hotplug features: *network*, *disk*, *cpu*, *memory* and *usb*. Use *0* to disable hotplug completely. Value *1* is an alias for the default *network,disk,usb*.

-ide[n] [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bps>

Use volume as IDE hard disk or CD-ROM (n is 0 to 3).

-keyboard (da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | fr-ca |

Keybord layout for vnc server. Default is read from the */etc/pve/datacenter.conf* configuration file.

-kvm boolean (default=1)
Enable/disable KVM hardware virtualization.

-localtime boolean
Set the real time clock to local time. This is enabled by default if ostype indicates a Microsoft OS.

-lock (backup | migrate | rollback | snapshot)
Lock/unlock the VM.

-machine (pc|pc(-i440fx)?-\d+\.\d+(\.pxe)?|q35|pc-q35-\d+\.\d+(\.pxe)?)

Specific the Qemu machine type.

-memory integer (16 -N) (default=512)
Amount of RAM for the VM in MB. This is the maximum available memory when you use the balloon device.

-migrate_downtime number (0 -N) (default=0.1)
Set maximum tolerated downtime (in seconds) for migrations.

-migrate_speed integer (0 -N) (default=0)
Set maximum speed (in MB/s) for migrations. Value 0 is no limit.

-name string
Set a name for the VM. Only used on the configuration web interface.

-net[n] [model=]<model> [,bridge=<bridge>] [,firewall=<1|0>] [,link_down=<1

Specify network devices.

-numa boolean (default=0)
Enable/disable NUMA.

-numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>] [,memory=<number>] [

NUMA topology.

-onboot boolean (default=0)
Specifies whether a VM will be started during system bootup.

-ostype (l24 | l26 | other | solaris | w2k | w2k3 | w2k8 | win7 | win8 | wv

Specify guest operating system.

-parallel[n] /dev/parport\d+|/dev/usb/lp\d+
Map host parallel devices (n is 0 to 2).

-protection boolean (default=0)
Sets the protection flag of the VM. This will disable the remove VM and remove disk operations.

-reboot boolean (default=1)
Allow reboot. If set to *0* the VM exit on reboot.

-revert string
Revert a pending change.

-sata[n] [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bps>

Use volume as SATA hard disk or CD-ROM (n is 0 to 5).

-scsi[n] [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<bps>

Use volume as SCSI hard disk or CD-ROM (n is 0 to 13).

-scsihw (lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci | virtio-scsi-

SCSI controller model

-serial[n] (/dev/.+|socket)
Create a serial device inside the VM (n is 0 to 3)

-shares integer (0 -50000) (default=1000)
Amount of memory shares for auto-ballooning. The larger the number is, the more memory this VM gets. Number is relative to weights of all other running VMs. Using zero disables auto-ballooning

-skiplock boolean
Ignore locks - only root is allowed to use this option.

-smbios1 [family=<string>] [,manufacturer=<string>] [,product=<string>] [,se

Specify SMBIOS type 1 fields.

-smp integer (1 -N) (default=1)
The number of CPUs. Please use option -sockets instead.

-sockets integer (1 -N) (default=1)
The number of CPU sockets.

-startdate (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (default=now)
Set the initial date of the real time clock. Valid format for date are: *now* or *2006-06-17T16:01:21* or *2006-06-17*.

-startup `[[order=]\d+] [,up=\d+] [,down=\d+] `
Startup and shutdown behavior. Order is a non-negative number defining the general startup order. Shutdown in done with reverse ordering. Additionally you can set the *up* or *down* delay in seconds, which specifies a delay to wait before the next VM is started or stopped.

-tablet boolean (default=1)
Enable/disable the USB tablet device.

-tdf boolean (default=0)
Enable/disable time drift fix.

-template boolean (default=0)
Enable/disable Template.

-unused[n] string
Reference to unused volumes. This is used internally, and should not be modified manually.

-usb[n] [host=]<HOSTUSBDEVICE|spice> [,usb3=<1|0>]
Configure an USB device (n is 0 to 4).

-vcpus integer (1 -N) (default=0)
Number of hotplugged vcpus.

-vga (cirrus | qxl | qxl2 | qxl3 | qxl4 | serial0 | serial1 | serial2 | ser

Select the VGA type.

-virtio[n] [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>] [,bps=<b

Use volume as VIRTIO hard disk (n is 0 to 15).

-watchdog [[model=]<i6300esb|ib700>] [,action=<reset|shutdown|poweroff|paus

Create a virtual hardware watchdog device.

qm showcmd <vmid>

Show command line which is used to start the VM (debug info).

<vmid> integer (1 -N)
The (unique) ID of the VM.

qm shutdown <vmid> [OPTIONS]

Shutdown virtual machine. This is similar to pressing the power button on a physical machine.This will send an ACPI event for the guest OS, which should then proceed to a clean shutdown.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-forceStop boolean (default=0)
Make sure the VM stops.

-keepActive boolean (default=0)
Do not decativate storage volumes.

-skiplock boolean
Ignore locks - only root is allowed to use this option.

-timeout integer (0 -N)
Wait maximal timeout seconds.

qm snapshot <vmid> <snapname> [OPTIONS]

Snapshot a VM.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<snapname> string
The name of the snapshot.

-description string
A textual description or comment.

-vmstate boolean
Save the vmstate

qm start <vmid> [OPTIONS]

Start virtual machine.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-machine (pc|pc(-i440fx)?-\d+\.\d+(\.pxe)?|q35|pc-q35-\d+\.\d+(\.pxe)?)

Specific the Qemu machine type.

-migratedfrom string
The cluster node name.

-skiplock boolean
Ignore locks - only root is allowed to use this option.

-stateuri string
Some command save/restore state from this location.

qm status <vmid> [OPTIONS]

Show VM status.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-verbose boolean
Verbose output format

qm stop <vmid> [OPTIONS]

Stop virtual machine. The qemu process will exit immediately. Thisis akin to pulling the power plug of a running computer and may damage the VM data

<vmid> integer (1 -N)
The (unique) ID of the VM.

-keepActive boolean (default=0)
Do not decativate storage volumes.

-migratedfrom string
The cluster node name.

-skiplock boolean
Ignore locks - only root is allowed to use this option.

-timeout integer (0 -N)
Wait maximal timeout seconds.

qm suspend <vmid> [OPTIONS]

Suspend virtual machine.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-skiplock boolean
Ignore locks - only root is allowed to use this option.

qm template <vmid> [OPTIONS]

Create a Template.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-disk (ide0 | ide1 | ide2 | ide3 | sata0 | sata1 | sata2 | sata3 | sata4 |

If you want to convert only 1 disk to base image.

qm terminal <vmid> [OPTIONS]

Open a terminal using a serial device (The VM need to have a serial device configured, for example *serial0: socket*)

<vmid> integer (1 -N)
The (unique) ID of the VM.

-iface (serial0 | serial1 | serial2 | serial3)
Select the serial device. By default we simply use the first suitable device.

qm unlink <vmid> -idlist <string> [OPTIONS]

Unlink/delete disk images.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-force boolean
Force physical removal. Without this, we simple remove the disk from the config file and create an additional configuration entry called *unused[n]*, which contains the volume ID. Unlink of unused[n] always cause physical removal.

-idlist string
A list of disk IDs you want to delete.

qm unlock <vmid>

Unlock the VM.

<vmid> integer (1 -N)
The (unique) ID of the VM.

qm vncproxy <vmid>

Proxy VM VNC traffic to stdin/stdout

<vmid> integer (1 -N)
The (unique) ID of the VM.

qm wait <vmid> [OPTIONS]

Wait until the VM is stopped.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-timeout integer (1 -N)
Timeout in seconds. Default is to wait forever.

A.5 qmrestore - Restore QemuServer *vzdump* Backups

qmrestore help

qmrestore <archive> <vmid> [OPTIONS]

Restore QemuServer vzdump backups.

<archive> string
The backup file. You can pass - to read from standard input.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-force boolean
Allow to overwrite existing VM.

-pool string
Add the VM to the specified pool.

-storage string
Default storage.

-unique boolean
Assign a unique random ethernet address.

A.6 pct - Proxmox Container Toolkit

pct <COMMAND> [ARGS] [OPTIONS]

pct clone <vmid> <newid> -experimental <boolean> [OPTIONS]

Create a container clone/copy

<vmid> integer (1 -N)
The (unique) ID of the VM.

<newid> integer (1 -N)
VMID for the clone.

-description string
Description for the new CT.

-experimental boolean (default=0)
The clone feature is experimental, set this flag if you know what you are doing.

-full boolean (default=0)
Create a full copy of all disk. This is always done when you clone a normal CT. For CT templates, we try to create a linked clone by default.

-hostname string
Set a hostname for the new CT.

-pool string
Add the new CT to the specified pool.

-snapname string
The name of the snapshot.

-storage string
Target storage for full clone.

> **Note**
> Requires option(s): full

pct config <vmid>

Get container configuration.

<vmid> integer (1 -N)
The (unique) ID of the VM.

pct console <vmid>

Launch a console for the specified container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

pct create <vmid> <ostemplate> [OPTIONS]

Create or restore a container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<ostemplate> string
The OS template or backup file.

-arch (amd64 | i386) (default=amd64)
OS architecture type.

-cmode (console | shell | tty) (default=tty)
Console mode. By default, the console command tries to open a connection to one of the available tty devices. By setting cmode to *console* it tries to attach to /dev/console instead. If you set cmode to *shell*, it simply invokes a shell inside the container (no login).

-console boolean (default=1)
Attach a console device (/dev/console) to the container.

-cpulimit number (0 -128) (default=0)
Limit of CPU usage.

Note
If the computer has 2 CPUs, it has a total of *2* CPU time. Value *0* indicates no CPU limit.

-cpuunits integer (0 -500000) (default=1024)
CPU weight for a VM. Argument is used in the kernel fair scheduler. The larger the number is, the more CPU time this VM gets. Number is relative to the weights of all the other running VMs.

Note
You can disable fair-scheduler configuration by setting this to 0.

-description string
Container description. Only used on the configuration web interface.

-force boolean
Allow to overwrite existing container.

-hostname string
Set a host name for the container.

-ignore-unpack-errors boolean
Ignore errors when extracting the template.

-lock (backup | migrate | rollback | snapshot)
Lock/unlock the VM.

-memory integer (16 -N) (default=512)
Amount of RAM for the VM in MB.

-mp[n] [volume=]<volume> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>] [,quota=<1

Use volume as container mount point.

-nameserver string
Sets DNS server IP address for a container. Create will automatically use the setting from the host if you neither set searchdomain nor nameserver.

-net[n] name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv

Specifies network interfaces for the container.

-onboot boolean (default=0)
Specifies whether a VM will be started during system bootup.

-ostype (alpine | archlinux | centos | debian | fedora | gentoo | opensuse

OS type. This is used to setup configuration inside the container, and corresponds to lxc setup scripts in /usr/share/lxc/config/<ostype>.common.conf. Value *unmanaged* can be used to skip and OS specific setup.

-password
Sets root password inside container.

-pool string
Add the VM to the specified pool.

-protection boolean (default=0)
Sets the protection flag of the container. This will prevent the CT or CT's disk remove/update operation.

-restore boolean
Mark this as restore task.

-rootfs [volume=]<volume> [,acl=<1|0>] [,quota=<1|0>] [,ro=<1|0>] [,size=<D

Use volume as container root.

-searchdomain string
Sets DNS search domains for a container. Create will automatically use the setting from the host if you neither set searchdomain nor nameserver.

-ssh-public-keys filepath
Setup public SSH keys (one key per line, OpenSSH format).

-startup `[[order=]\d+] [,up=\d+] [,down=\d+] `
Startup and shutdown behavior. Order is a non-negative number defining the general startup order. Shutdown in done with reverse ordering. Additionally you can set the *up* or *down* delay in seconds, which specifies a delay to wait before the next VM is started or stopped.

-storage string (default=local)
Default Storage.

-swap integer (0 -N) (default=512)
Amount of SWAP for the VM in MB.

-template boolean (default=0)
Enable/disable Template.

-tty integer (0 -6) (default=2)
Specify the number of tty available to the container

-unprivileged boolean (default=0)
Makes the container run as unprivileged user. (Should not be modified manually.)

-unused[n] string
Reference to unused volumes. This is used internally, and should not be modified manually.

pct delsnapshot `<vmid> <snapname> [OPTIONS]`

Delete a LXC snapshot.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<snapname> string
The name of the snapshot.

-force boolean
For removal from config file, even if removing disk snapshots fails.

pct destroy `<vmid>`

Destroy the container (also delete all uses files).

<vmid> integer (1 -N)
The (unique) ID of the VM.

pct enter `<vmid>`

Launch a shell for the specified container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

pct exec `<vmid> [<extra-args>]`

Launch a command inside the specified container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<extra-args> array
Extra arguments as array

pct fsck <vmid> [OPTIONS]

Run a filesystem check (fsck) on a container volume.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-device (mp0 | mp1 | mp2 | mp3 | mp4 | mp5 | mp6 | mp7 | mp8 | mp9 | rootfs)

A volume on which to run the filesystem check

-force boolean (default=0)
Force checking, even if the filesystem seems clean

pct help [<cmd>] [OPTIONS]

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

pct list

LXC container index (per node).

pct listsnapshot <vmid>

List all snapshots.

<vmid> integer (1 -N)
The (unique) ID of the VM.

pct migrate <vmid> <target> [OPTIONS]

Migrate the container to another node. Creates a new migration task.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<target> string
Target node.

-force boolean
Force migration despite local bind / device mounts. WARNING: identical bind / device mounts need to be available on the target node.

-online boolean
Use online/live migration.

pct mount <vmid>

Mount the container's filesystem on the host. This will hold a lock on the container and is meant for emergency maintenance only as it will prevent further operations on the container other than start and stop.

<vmid> integer (1 -N)
The (unique) ID of the VM.

pct pull <vmid> <path> <destination> [OPTIONS]

Copy a file from the container to the local system.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<path> string
Path to a file inside the container to pull.

<destination> string
Destination

-group string
Owner group name or id.

-perms string
File permissions to use (octal by default, prefix with *0x* for hexadecimal).

-user string
Owner user name or id.

pct push <vmid> <file> <destination> [OPTIONS]

Copy a local file to the container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<file> string
Path to a local file.

<destination> string
Destination inside the container to write to.

-group string
Owner group name or id. When using a name it must exist inside the container.

-perms string
File permissions to use (octal by default, prefix with *0x* for hexadecimal).

-user string
Owner user name or id. When using a name it must exist inside the container.

pct resize <vmid> <disk> <size> [OPTIONS]

Resize a container mountpoint.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<disk> (mp0 | mp1 | mp2 | mp3 | mp4 | mp5 | mp6 | mp7 | mp8 | mp9 | rootfs)

The disk you want to resize.

<size> \+?\d+(\.\d+)?[KMGT]?
The new size. With the + sign the value is added to the actual size of the volume and without it, the value is taken as an absolute one. Shrinking disk size is not supported.

-digest string
Prevent changes if current configuration file has different SHA1 digest. This can be used to prevent concurrent modifications.

pct restore <vmid> <ostemplate> [OPTIONS]

Create or restore a container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<ostemplate> string
The OS template or backup file.

-arch (amd64 | i386) (default=amd64)
OS architecture type.

-cmode (console | shell | tty) (default=tty)
Console mode. By default, the console command tries to open a connection to one of the available tty devices. By setting cmode to *console* it tries to attach to /dev/console instead. If you set cmode to *shell*, it simply invokes a shell inside the container (no login).

-console boolean (default=1)
Attach a console device (/dev/console) to the container.

-cpulimit number (0 -128) (default=0)
Limit of CPU usage.

> **Note**
> If the computer has 2 CPUs, it has a total of *2* CPU time. Value *0* indicates no CPU limit.

-cpuunits integer (0 -500000) (default=1024)
CPU weight for a VM. Argument is used in the kernel fair scheduler. The larger the number is, the more CPU time this VM gets. Number is relative to the weights of all the other running VMs.

> **Note**
> You can disable fair-scheduler configuration by setting this to 0.

-description string
Container description. Only used on the configuration web interface.

-force boolean
Allow to overwrite existing container.

-hostname string
Set a host name for the container.

-ignore-unpack-errors boolean
Ignore errors when extracting the template.

-lock (backup | migrate | rollback | snapshot)
Lock/unlock the VM.

-memory integer (16 -N) (default=512)
Amount of RAM for the VM in MB.

-mp[n] [volume=]<volume> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>] [,quota=<1

Use volume as container mount point.

-nameserver string
Sets DNS server IP address for a container. Create will automatically use the setting from the host if you neither set searchdomain nor nameserver.

-net[n] name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv

Specifies network interfaces for the container.

-onboot boolean (default=0)
Specifies whether a VM will be started during system bootup.

-ostype (alpine | archlinux | centos | debian | fedora | gentoo | opensuse

OS type. This is used to setup configuration inside the container, and corresponds to lxc setup scripts in /usr/share/lxc/config/<ostype>.common.conf. Value *unmanaged* can be used to skip and OS specific setup.

-password
Sets root password inside container.

-pool string
Add the VM to the specified pool.

-protection boolean (default=0)
Sets the protection flag of the container. This will prevent the CT or CT's disk remove/update operation.

-rootfs [volume=]<volume> [,acl=<1|0>] [,quota=<1|0>] [,ro=<1|0>] [,size=<D

Use volume as container root.

-searchdomain string
Sets DNS search domains for a container. Create will automatically use the setting from the host if you neither set searchdomain nor nameserver.

-ssh-public-keys filepath
Setup public SSH keys (one key per line, OpenSSH format).

-startup `[[order=]\d+] [,up=\d+] [,down=\d+] `
Startup and shutdown behavior. Order is a non-negative number defining the general startup order. Shutdown in done with reverse ordering. Additionally you can set the *up* or *down* delay in seconds, which specifies a delay to wait before the next VM is started or stopped.

-storage string (default=local)
Default Storage.

-swap integer (0 -N) (default=512)
Amount of SWAP for the VM in MB.

-template boolean (default=0)
Enable/disable Template.

-tty integer (0 -6) (default=2)
Specify the number of tty available to the container

-unprivileged boolean (default=0)
Makes the container run as unprivileged user. (Should not be modified manually.)

-unused[n] string
Reference to unused volumes. This is used internally, and should not be modified manually.

pct resume <vmid>

Resume the container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

pct rollback <vmid> <snapname>

Rollback LXC state to specified snapshot.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<snapname> string
The name of the snapshot.

pct set <vmid> [OPTIONS]

Set container options.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-arch (amd64 | i386) (default=amd64)
OS architecture type.

-cmode (console | shell | tty) (default=tty)
Console mode. By default, the console command tries to open a connection to one of the available tty devices. By setting cmode to *console* it tries to attach to /dev/console instead. If you set cmode to *shell*, it simply invokes a shell inside the container (no login).

-console boolean (default=1)
Attach a console device (/dev/console) to the container.

-cpulimit number (0 -128) (default=0)
Limit of CPU usage.

> **Note**
> If the computer has 2 CPUs, it has a total of *2* CPU time. Value *0* indicates no CPU limit.

-cpuunits integer (0 -500000) (default=1024)
CPU weight for a VM. Argument is used in the kernel fair scheduler. The larger the number is, the more CPU time this VM gets. Number is relative to the weights of all the other running VMs.

> **Note**
> You can disable fair-scheduler configuration by setting this to 0.

-delete string
A list of settings you want to delete.

-description string
Container description. Only used on the configuration web interface.

-digest string
Prevent changes if current configuration file has different SHA1 digest. This can be used to prevent concurrent modifications.

-hostname string
Set a host name for the container.

-lock (backup | migrate | rollback | snapshot)
Lock/unlock the VM.

-memory integer (16 -N) (default=512)
Amount of RAM for the VM in MB.

-mp[n] [volume=]<volume> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>] [,quota=<1

Use volume as container mount point.

-nameserver string
Sets DNS server IP address for a container. Create will automatically use the setting from the host if you neither set searchdomain nor nameserver.

-net[n] name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv

Specifies network interfaces for the container.

-onboot boolean (default=0)
Specifies whether a VM will be started during system bootup.

-ostype (alpine | archlinux | centos | debian | fedora | gentoo | opensuse

OS type. This is used to setup configuration inside the container, and corresponds to lxc setup scripts in /usr/share/lxc/config/<ostype>.common.conf. Value *unmanaged* can be used to skip and OS specific setup.

-protection boolean (default=0)
Sets the protection flag of the container. This will prevent the CT or CT's disk remove/update operation.

-rootfs [volume=]<volume> [,acl=<1|0>] [,quota=<1|0>] [,ro=<1|0>] [,size=<D

Use volume as container root.

-searchdomain string
Sets DNS search domains for a container. Create will automatically use the setting from the host if you neither set searchdomain nor nameserver.

-startup `[[order=]\d+] [,up=\d+] [,down=\d+] `
Startup and shutdown behavior. Order is a non-negative number defining the general startup order. Shutdown in done with reverse ordering. Additionally you can set the *up* or *down* delay in seconds, which specifies a delay to wait before the next VM is started or stopped.

-swap integer (0 -N) (default=512)
Amount of SWAP for the VM in MB.

-template boolean (default=0)
Enable/disable Template.

-tty integer (0 -6) (default=2)
Specify the number of tty available to the container

-unprivileged boolean (default=0)
Makes the container run as unprivileged user. (Should not be modified manually.)

-unused[n] string
Reference to unused volumes. This is used internally, and should not be modified manually.

pct shutdown <vmid> [OPTIONS]

Shutdown the container. This will trigger a clean shutdown of the container, see lxc-stop(1) for details.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-forceStop boolean (default=0)
Make sure the Container stops.

-timeout integer (0 -N) (default=60)
Wait maximal timeout seconds.

pct snapshot <vmid> <snapname> [OPTIONS]

Snapshot a container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

<snapname> string
The name of the snapshot.

-description string
A textual description or comment.

pct start <vmid> [OPTIONS]

Start the container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-skiplock boolean
Ignore locks - only root is allowed to use this option.

pct stop <vmid> [OPTIONS]

Stop the container. This will abruptly stop all processes running in the container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-skiplock boolean
Ignore locks - only root is allowed to use this option.

pct suspend <vmid>

Suspend the container.

<vmid> integer (1 -N)
The (unique) ID of the VM.

pct template <vmid> -experimental <boolean> [OPTIONS]

Create a Template.

<vmid> integer (1 -N)
The (unique) ID of the VM.

-experimental boolean (default=0)
The template feature is experimental, set this flag if you know what you are doing.

pct unlock <vmid>

Unlock the VM.

<vmid> integer (1 -N)
The (unique) ID of the VM.

pct unmount <vmid>

Unmount the container's filesystem.

<vmid> integer (1 -N)
The (unique) ID of the VM.

A.7 pveam - Proxmox VE Appliance Manager

pveam <COMMAND> [ARGS] [OPTIONS]

pveam available [OPTIONS]

List available templates.

-section (system | turnkeylinux)
Restrict list to specified section.

pveam download <storage> <template>

Download appliance templates.

<storage> string
Only list status for specified storage

<template> string
The template wich will downloaded

pveam help [<cmd>] [OPTIONS]

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

pveam list <storage>

Get list of all templates on storage

<storage> string
Only list status for specified storage

pveam remove <template_path>

Remove a template.

<template_path> string
The template to remove.

pveam update

Update Container Template Database.

A.8 pvecm - Proxmox VE Cluster Manager

pvecm <COMMAND> [ARGS] [OPTIONS]

pvecm add <hostname> [OPTIONS]

Adds the current node to an existing cluster.

<hostname> string
Hostname (or IP) of an existing cluster member.

-force boolean
Do not throw error if node already exists.

-nodeid integer (1 -N)
Node id for this node.

-ring0_addr string
Hostname (or IP) of the corosync ring0 address of this node. Defaults to nodes hostname.

-ring1_addr string
Hostname (or IP) of the corosync ring1 address, this needs an valid configured ring 1 interface in the cluster.

-votes integer (0 -N)
Number of votes for this node

pvecm addnode `<node> [OPTIONS]`

Adds a node to the cluster configuration.

<node> string
The cluster node name.

-force boolean
Do not throw error if node already exists.

-nodeid integer (1 -N)
Node id for this node.

-ring0_addr string
Hostname (or IP) of the corosync ring0 address of this node. Defaults to nodes hostname.

-ring1_addr string
Hostname (or IP) of the corosync ring1 address, this needs an valid bindnet1_addr.

-votes integer (0 -N)
Number of votes for this node

pvecm create `<clustername> [OPTIONS]`

Generate new cluster configuration.

<clustername> string
The name of the cluster.

-bindnet0_addr string
This specifies the network address the corosync ring 0 executive should bind to and defaults to the local IP address of the node.

-bindnet1_addr string
This specifies the network address the corosync ring 1 executive should bind to and is optional.

-nodeid integer (1 -N)
Node id for this node.

-ring0_addr string
Hostname (or IP) of the corosync ring0 address of this node. Defaults to the hostname of the node.

-ring1_addr string
Hostname (or IP) of the corosync ring1 address, this needs an valid bindnet1_addr.

-rrp_mode (active | none | passive) (default=none)
This specifies the mode of redundant ring, which may be none, active or passive. Using multiple interfaces only allows *active* or *passive*.

-votes integer (1 -N)
Number of votes for this node.

pvecm delnode `<node>`

Removes a node to the cluster configuration.

<node> string
The cluster node name.

pvecm expected `<expected>`

Tells corosync a new value of expected votes.

<expected> integer (1 -N)
Expected votes

pvecm help `[<cmd>] [OPTIONS]`

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

pvecm keygen `<filename>`

Generate new cryptographic key for corosync.

<filename> string
Output file name

pvecm nodes

Displays the local view of the cluster nodes.

pvecm status

Displays the local view of the cluster status.

pvecm updatecerts `[OPTIONS]`

Update node certificates (and generate all needed files/directories).

-force boolean
Force generation of new SSL certifate.

-silent boolean
Ignore errors (i.e. when cluster has no quorum).

A.9 pveum - Proxmox VE User Manager

pveum <COMMAND> [ARGS] [OPTIONS]

pveum acldel <path> -roles <string> [OPTIONS]

Update Access Control List (add or remove permissions).

<path> string
Access control path

-groups string
List of groups.

-propagate boolean (default=1)
Allow to propagate (inherit) permissions.

-roles string
List of roles.

-users string
List of users.

pveum aclmod <path> -roles <string> [OPTIONS]

Update Access Control List (add or remove permissions).

<path> string
Access control path

-groups string
List of groups.

-propagate boolean (default=1)
Allow to propagate (inherit) permissions.

-roles string
List of roles.

-users string
List of users.

pveum groupadd <groupid> [OPTIONS]

Create new group.

<groupid> string
no description available

-comment string
no description available

pveum groupdel `<groupid>`

Delete group.

`<groupid>` `string`
no description available

pveum groupmod `<groupid> [OPTIONS]`

Update group data.

`<groupid>` `string`
no description available

`-comment` `string`
no description available

pveum help `[<cmd>] [OPTIONS]`

Get help about specified command.

`<cmd>` `string`
Command name

`-verbose` `boolean`
Verbose output format.

pveum passwd `<userid>`

Change user password.

`<userid>` `string`
User ID

pveum roleadd `<roleid> [OPTIONS]`

Create new role.

`<roleid>` `string`
no description available

`-privs` `string`
no description available

pveum roledel `<roleid>`

Delete role.

`<roleid>` `string`
no description available

pveum rolemod `<roleid> -privs <string> [OPTIONS]`

Create new role.

`<roleid>` `string`
no description available

`-append` `boolean`
no description available

> **Note**
> Requires option(s): `privs`

`-privs` `string`
no description available

pveum ticket `<username> [OPTIONS]`

Create or verify authentication ticket.

`<username>` `string`
User name

`-otp` `string`
One-time password for Two-factor authentication.

`-path` `string`
Verify ticket, and check if user have access *privs* on *path*

> **Note**
> Requires option(s): `privs`

`-privs` `string`
Verify ticket, and check if user have access *privs* on *path*

> **Note**
> Requires option(s): `path`

`-realm` `string`
You can optionally pass the realm using this parameter. Normally the realm is simply added to the username <username>@<relam>.

pveum useradd `<userid> [OPTIONS]`

Create new user.

<userid> string
User ID

-comment string
no description available

-email string
no description available

-enable boolean (default=1)
Enable the account (default). You can set this to *0* to disable the accout

-expire integer (0 -N)
Account expiration date (seconds since epoch). *0* means no expiration date.

-firstname string
no description available

-groups string
no description available

-keys string
Keys for two factor auth (yubico).

-lastname string
no description available

-password
Initial password.

pveum userdel `<userid>`

Delete user.

<userid> string
User ID

pveum usermod `<userid> [OPTIONS]`

Update user configuration.

<userid> string
User ID

-append boolean
no description available

> **Note**
> Requires option(s): `groups`

-comment string
no description available

-email string
no description available

-enable boolean
Enable/disable the account.

-expire integer (0 -N)
Account expiration date (seconds since epoch). *0* means no expiration date.

-firstname string
no description available

-groups string
no description available

-keys string
Keys for two factor auth (yubico).

-lastname string
no description available

A.10 vzdump - Backup Utility for VMs and Containers

vzdump `help`

vzdump `{<vmid>} [OPTIONS]`

Create backup.

<vmid> string
The ID of the guest system you want to backup.

-all boolean (default=0)
Backup all known guest systems on this host.

-bwlimit integer (0 -N) (default=0)
Limit I/O bandwidth (KBytes per second).

-compress (0 | 1 | gzip | lzo) (default=0)
Compress dump file.

-dumpdir string
Store resulting files to specified directory.

-exclude string
Exclude specified guest systems (assumes --all)

-exclude-path string
Exclude certain files/directories (shell globs).

-ionice integer (0 -8) (default=7)
Set CFQ ionice priority.

-lockwait integer (0 -N) (default=180)
Maximal time to wait for the global lock (minutes).

-mailnotification (always | failure) (default=always)
Specify when to send an email

-mailto string
Comma-separated list of email addresses that should receive email notifications.

-maxfiles integer (1 -N) (default=1)
Maximal number of backup files per guest system.

-mode (snapshot | stop | suspend) (default=snapshot)
Backup mode.

-node string
Only run if executed on this node.

-pigz integer (default=0)
Use pigz instead of gzip when N>0. N=1 uses half of cores, N>1 uses N as thread count.

-quiet boolean (default=0)
Be quiet.

-remove boolean (default=1)
Remove old backup files if there are more than *maxfiles* backup files.

-script string
Use specified hook script.

-size integer (500 -N) (default=1024)
Unused, will be removed in a future release.

-stdexcludes boolean (default=1)
Exclude temporary files and logs.

-stdout boolean
Write tar to stdout, not to a file.

-stop boolean (default=0)
Stop runnig backup jobs on this host.

-stopwait integer (0 -N) (default=10)
Maximal time to wait until a guest system is stopped (minutes).

-storage string
Store resulting file to this storage.

-tmpdir string
Store temporary files to specified directory.

A.11 ha-manager - Proxmox VE HA Manager

ha-manager <COMMAND> [ARGS] [OPTIONS]

ha-manager add <sid> [OPTIONS]

Create a new HA resource.

<sid> <type>:<name>
HA resource ID. This consists of a resource type followed by a resource specific name, separated with colon (example: vm:100 / ct:100). For virtual machines and containers, you can simply use the VM or CT id as a shortcut (example: 100).

-comment string
Description.

-group string
The HA group identifier.

-max_relocate integer (0 -N) (default=1)
Maximal number of service relocate tries when a service failes to start.

-max_restart integer (0 -N) (default=1)
Maximal number of tries to restart the service on a node after its start failed.

-state (disabled | enabled) (default=enabled)
Resource state.

-type (ct | vm)
Resource type.

ha-manager config [OPTIONS]

List HA resources.

-type (ct | vm)
Only list resources of specific type

ha-manager disable <sid>

Disable a HA resource.

<sid> <type>:<name>
HA resource ID. This consists of a resource type followed by a resource specific name, separated with colon (example: vm:100 / ct:100). For virtual machines and containers, you can simply use the VM or CT id as a shortcut (example: 100).

ha-manager enable <sid>

Enable a HA resource.

<sid> <type>:<name>
HA resource ID. This consists of a resource type followed by a resource specific name, separated with colon (example: vm:100 / ct:100). For virtual machines and containers, you can simply use the VM or CT id as a shortcut (example: 100).

ha-manager groupadd <group> –nodes <string> [OPTIONS]

Create a new HA group.

<group> string
The HA group identifier.

–comment string
Description.

–nodes <node>[:<pri>]{,<node>[:<pri>]}*
List of cluster node names with optional priority. We use priority *0* as default. The CRM tries to run services on the node with highest priority (also see option *nofailback*).

–nofailback boolean (default=0)
The CRM tries to run services on the node with the highest priority. If a node with higher priority comes online, the CRM migrates the service to that node. Enabling nofailback prevents that behavior.

–restricted boolean (default=0)
Services on unrestricted groups may run on any cluster members if all group members are offline. But they will migrate back as soon as a group member comes online. One can implement a *preferred node* behavior using an unrestricted group with one member.

–type (group)
Group type.

ha-manager groupconfig

Get HA groups.

ha-manager groupremove <group>

Delete ha group configuration.

<group> string
The HA group identifier.

ha-manager groupset <group> [OPTIONS]

Update ha group configuration.

<group> string
The HA group identifier.

-comment string
Description.

-delete string
A list of settings you want to delete.

-digest string
Prevent changes if current configuration file has different SHA1 digest. This can be used to prevent concurrent modifications.

-nodes <node>[:<pri>]{,<node>[:<pri>]}*
List of cluster node names with optional priority. We use priority *0* as default. The CRM tries to run services on the node with highest priority (also see option *nofailback*).

-nofailback boolean (default=0)
The CRM tries to run services on the node with the highest priority. If a node with higher priority comes online, the CRM migrates the service to that node. Enabling nofailback prevents that behavior.

-restricted boolean (default=0)
Services on unrestricted groups may run on any cluster members if all group members are offline. But they will migrate back as soon as a group member comes online. One can implement a *preferred node* behavior using an unrestricted group with one member.

ha-manager help `[<cmd>] [OPTIONS]`

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

ha-manager migrate `<sid> <node>`

Request resource migration (online) to another node.

<sid> <type>:<name>
HA resource ID. This consists of a resource type followed by a resource specific name, separated with colon (example: vm:100 / ct:100). For virtual machines and containers, you can simply use the VM or CT id as a shortcut (example: 100).

<node> string
The cluster node name.

ha-manager relocate `<sid> <node>`

Request resource relocatzion to another node. This stops the service on the old node, and restarts it on the target node.

<sid> <type>:<name>
HA resource ID. This consists of a resource type followed by a resource specific name, separated with colon (example: vm:100 / ct:100). For virtual machines and containers, you can simply use the VM or CT id as a shortcut (example: 100).

<node> string
The cluster node name.

ha-manager remove <sid>

Delete resource configuration.

<sid> <type>:<name>
HA resource ID. This consists of a resource type followed by a resource specific name, separated with colon (example: vm:100 / ct:100). For virtual machines and containers, you can simply use the VM or CT id as a shortcut (example: 100).

ha-manager set <sid> [OPTIONS]

Update resource configuration.

<sid> <type>:<name>
HA resource ID. This consists of a resource type followed by a resource specific name, separated with colon (example: vm:100 / ct:100). For virtual machines and containers, you can simply use the VM or CT id as a shortcut (example: 100).

-comment string
Description.

-delete string
A list of settings you want to delete.

-digest string
Prevent changes if current configuration file has different SHA1 digest. This can be used to prevent concurrent modifications.

-group string
The HA group identifier.

-max_relocate integer (0 -N) (default=1)
Maximal number of service relocate tries when a service failes to start.

-max_restart integer (0 -N) (default=1)
Maximal number of tries to restart the service on a node after its start failed.

-state (disabled | enabled) (default=enabled)
Resource state.

ha-manager status [OPTIONS]

Display HA manger status.

-verbose boolean (default=0)
Verbose output. Include complete CRM and LRM status (JSON).

Appendix B

Service Daemons

B.1 pve-firewall - Proxmox VE Firewall Daemon

pve-firewall `<COMMAND> [ARGS] [OPTIONS]`

pve-firewall compile

Compile and print firewall rules. This is useful for testing.

pve-firewall help `[<cmd>] [OPTIONS]`

Get help about specified command.

`<cmd>` `string`
: Command name

`-verbose` `boolean`
: Verbose output format.

pve-firewall localnet

Print information about local network.

pve-firewall restart

Restart the Proxmox VE firewall service.

pve-firewall simulate `[OPTIONS]`

Simulate firewall rules. This does not simulate kernel *routing* table. Instead, this simply assumes that routing from source zone to destination zone is possible.

`-dest` `string`
: Destination IP address.

`-dport` `integer`
: Destination port.

`-from (host|outside|vm\d+|ct\d+|vmbr\d+/\S+)` (default=`outside`)
: Source zone.

-protocol (tcp|udp) (default=tcp)
Protocol.

-source string
Source IP address.

-sport integer
Source port.

-to (host|outside|vm\d+|ct\d+|vmbr\d+/\S+) (default=host)
Destination zone.

-verbose boolean (default=0)
Verbose output.

pve-firewall start [OPTIONS]

Start the Proxmox VE firewall service.

-debug boolean (default=0)
Debug mode - stay in foreground

pve-firewall status

Get firewall status.

pve-firewall stop

Stop firewall. This removes all Proxmox VE related iptable rules. The host is unprotected afterwards.

B.2 pvedaemon - Proxmox VE API Daemon

pvedaemon <COMMAND> [ARGS] [OPTIONS]

pvedaemon help [<cmd>] [OPTIONS]

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

pvedaemon restart

Restart the daemon (or start if not running).

pvedaemon start [OPTIONS]

Start the daemon.

-debug boolean (default=0)
Debug mode - stay in foreground

pvedaemon status

Get daemon status.

pvedaemon stop

Stop the daemon.

B.3 pveproxy - Proxmox VE API Proxy Daemon

pveproxy <COMMAND> [ARGS] [OPTIONS]

pveproxy help [<cmd>] [OPTIONS]

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

pveproxy restart

Restart the daemon (or start if not running).

pveproxy start [OPTIONS]

Start the daemon.

-debug boolean (default=0)
Debug mode - stay in foreground

pveproxy status

Get daemon status.

pveproxy stop

Stop the daemon.

B.4 pvestatd - Proxmox VE Status Daemon

pvestatd <COMMAND> [ARGS] [OPTIONS]

pvestatd help [<cmd>] [OPTIONS]

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

pvestatd restart

Restart the daemon (or start if not running).

pvestatd start [OPTIONS]

Start the daemon.

-debug boolean (default=0)
Debug mode - stay in foreground

pvestatd status

Get daemon status.

pvestatd stop

Stop the daemon.

B.5 spiceproxy - SPICE Proxy Service

spiceproxy <COMMAND> [ARGS] [OPTIONS]

spiceproxy help [<cmd>] [OPTIONS]

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

spiceproxy restart

Restart the daemon (or start if not running).

spiceproxy start [OPTIONS]

Start the daemon.

-debug boolean (default=0)
Debug mode - stay in foreground

spiceproxy status

Get daemon status.

spiceproxy stop

Stop the daemon.

B.6 pmxcfs - Proxmox Cluster File System

pmxcfs [OPTIONS]

Help Options:

-h, --help
Show help options

Application Options:

-d, --debug
Turn on debug messages

-f, --foreground
Do not daemonize server

-l, --local
Force local mode (ignore corosync.conf, force quorum)

This service is usually started and managed using systemd toolset. The service is called *pve-cluster*.

```
systemctl start pve-cluster

systemctl stop pve-cluster

systemctl status pve-cluster
```

B.7 pve-ha-crm - Cluster Ressource Manager Daemon

pve-ha-crm <COMMAND> [ARGS] [OPTIONS]

pve-ha-crm help [<cmd>] [OPTIONS]

Get help about specified command.

<cmd> string
Command name

-verbose boolean
Verbose output format.

pve-ha-crm start [OPTIONS]

Start the daemon.

-debug boolean (default=0)
Debug mode - stay in foreground

pve-ha-crm status

Get daemon status.

pve-ha-crm stop

Stop the daemon.

B.8 pve-ha-lrm - Local Ressource Manager Daemon

pve-ha-lrm `<COMMAND> [ARGS] [OPTIONS]`

pve-ha-lrm help `[<cmd>] [OPTIONS]`

Get help about specified command.

`<cmd>` `string`
: Command name

`-verbose` `boolean`
: Verbose output format.

pve-ha-lrm start `[OPTIONS]`

Start the daemon.

`-debug` `boolean` (default=0)
: Debug mode - stay in foreground

pve-ha-lrm status

Get daemon status.

pve-ha-lrm stop

Stop the daemon.

Appendix C

Configuration Files

C.1 Datacenter Configuration

The file */etc/pve/datacenter.cfg* is a configuration file for Proxmox VE. It contains cluster wide default values used by all nodes.

C.1.1 File Format

The file uses a simple colon separated key/value format. Each line has the following format:

```
OPTION: value
```

Blank lines in the file are ignored, and lines starting with a # character are treated as comments and are also ignored.

C.1.2 Options

console: (applet | html5 | vv)
Select the default Console viewer. You can either use the builtin java applet (VNC), an external virt-viewer comtatible application (SPICE), or an HTML5 based viewer (noVNC).

email_from: string
Specify email address to send notification from (default is root@$hostname)

fencing: (both | hardware | watchdog) (default=watchdog)
Set the fencing mode of the HA cluster. Hardware mode needs a valid configuration of fence devices in /etc/pve/ha/fence.cfg. With both all two modes are used.

Warning
hardware and *both* are EXPERIMENTAL & WIP

http_proxy: http://.*
Specify external http proxy which is used for downloads (example: *http://username:password@host:port/*)

keyboard: (da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | fr-ca |

Default keybord layout for vnc server.

language: (de | en)
Default GUI language.

max_workers: integer (1 -N)
Defines how many workers (per node) are maximal started on actions like *stopall VMs* or task from the ha-manager.

migration_unsecure: boolean
Migration is secure using SSH tunnel by default. For secure private networks you can disable it to speed up migration.

Appendix D

Firewall Macro Definitions

Amanda Amanda Backup

Action	proto	dport	sport
PARAM	udp	10080	
PARAM	tcp	10080	

Auth Auth (identd) traffic

Action	proto	dport	sport
PARAM	tcp	113	

BGP Border Gateway Protocol traffic

Action	proto	dport	sport
PARAM	tcp	179	

BitTorrent BitTorrent traffic for BitTorrent 3.1 and earlier

Action	proto	dport	sport
PARAM	tcp	6881:6889	
PARAM	udp	6881	

BitTorrent32 BitTorrent traffic for BitTorrent 3.2 and later

Action	proto	dport	sport
PARAM	tcp	6881:6999	
PARAM	udp	6881	

CVS Concurrent Versions System pserver traffic

Action	proto	dport	sport
PARAM	tcp	2401	

Ceph Ceph Storage Cluster traffic (Ceph Monitors, OSD & MDS Deamons)

Action	proto	dport	sport
PARAM	tcp	6789	
PARAM	tcp	6800:7300	

Citrix Citrix/ICA traffic (ICA, ICA Browser, CGP)

Action	proto	dport	sport
PARAM	tcp	1494	
PARAM	udp	1604	
PARAM	tcp	2598	

DAAP Digital Audio Access Protocol traffic (iTunes, Rythmbox daemons)

Action	proto	dport	sport
PARAM	tcp	3689	
PARAM	udp	3689	

DCC Distributed Checksum Clearinghouse spam filtering mechanism

Action	proto	dport	sport
PARAM	tcp	6277	

DHCPfwd Forwarded DHCP traffic

Action	proto	dport	sport
PARAM	udp	67:68	67:68

DHCPv6 DHCPv6 traffic

Action	proto	dport	sport
PARAM	udp	546:547	546:547

DNS Domain Name System traffic (upd and tcp)

Action	proto	dport	sport
PARAM	udp	53	
PARAM	tcp	53	

Distcc Distributed Compiler service

Action	proto	dport	sport
PARAM	tcp	3632	

FTP File Transfer Protocol

Action	proto	dport	sport
PARAM	tcp	21	

Finger Finger protocol (RFC 742)

Action	proto	dport	sport
PARAM	tcp	79	

GNUnet GNUnet secure peer-to-peer networking traffic

Action	proto	dport	sport
PARAM	tcp	2086	
PARAM	udp	2086	
PARAM	tcp	1080	
PARAM	udp	1080	

GRE Generic Routing Encapsulation tunneling protocol

Action	proto	dport	sport
PARAM	47		

Git Git distributed revision control traffic

Action	proto	dport	sport
PARAM	tcp	9418	

HKP OpenPGP HTTP keyserver protocol traffic

Action	proto	dport	sport
PARAM	tcp	11371	

HTTP Hypertext Transfer Protocol (WWW)

Action	proto	dport	sport
PARAM	tcp	80	

HTTPS Hypertext Transfer Protocol (WWW) over SSL

Action	proto	dport	sport
PARAM	tcp	443	

ICPV2 Internet Cache Protocol V2 (Squid) traffic

Action	proto	dport	sport
PARAM	udp	3130	

ICQ AOL Instant Messenger traffic

Action	proto	dport	sport
PARAM	tcp	5190	

IMAP Internet Message Access Protocol

Action	proto	dport	sport
PARAM	tcp	143	

IMAPS Internet Message Access Protocol over SSL

Action	proto	dport	sport
PARAM	tcp	993	

IPIP IPIP capsulation traffic

Action	proto	dport	sport
PARAM	94		

IPsec IPsec traffic

Action	proto	dport	sport
PARAM	udp	500	500
PARAM	50		

IPsecah IPsec authentication (AH) traffic

Action	proto	dport	sport
PARAM	udp	500	500
PARAM	51		

IPsecnat IPsec traffic and Nat-Traversal

Action	proto	dport	sport
PARAM	udp	500	
PARAM	udp	4500	
PARAM	50		

IRC Internet Relay Chat traffic

Action	proto	dport	sport
PARAM	tcp	6667	

Jetdirect HP Jetdirect printing

Action	proto	dport	sport
PARAM	tcp	9100	

L2TP Layer 2 Tunneling Protocol traffic

Action	proto	dport	sport
PARAM	udp	1701	

LDAP Lightweight Directory Access Protocol traffic

Action	proto	dport	sport
PARAM	tcp	389	

LDAPS Secure Lightweight Directory Access Protocol traffic

Action	proto	dport	sport
PARAM	tcp	636	

MSNP Microsoft Notification Protocol

Action	proto	dport	sport
PARAM	tcp	1863	

MSSQL Microsoft SQL Server

Action	proto	dport	sport
PARAM	tcp	1433	

Mail Mail traffic (SMTP, SMTPS, Submission)

Action	proto	dport	sport
PARAM	tcp	25	
PARAM	tcp	465	
PARAM	tcp	587	

Munin Munin networked resource monitoring traffic

Action	proto	dport	sport
PARAM	tcp	4949	

MySQL MySQL server

Action	proto	dport	sport
PARAM	tcp	3306	

NNTP NNTP traffic (Usenet).

Action	proto	dport	sport
PARAM	tcp	119	

NNTPS Encrypted NNTP traffic (Usenet)

Action	proto	dport	sport
PARAM	tcp	563	

NTP Network Time Protocol (ntpd)

Action	proto	dport	sport
PARAM	udp	123	

NeighborDiscovery IPv6 neighbor solicitation, neighbor and router advertisement

Action	proto	dport	sport
PARAM	icmpv6	router-solicitation	
PARAM	icmpv6	router-advertisement	
PARAM	icmpv6	neighbor-solicitation	
PARAM	icmpv6	neighbor-advertisement	

OSPF OSPF multicast traffic

Action	proto	dport	sport
PARAM	89		

OpenVPN OpenVPN traffic

Action	proto	dport	sport
PARAM	udp	1194	

PCA Symantec PCAnywere (tm)

Action	proto	dport	sport
PARAM	udp	5632	
PARAM	tcp	5631	

POP3 POP3 traffic

Action	proto	dport	sport
PARAM	tcp	110	

POP3S Encrypted POP3 traffic

Action	proto	dport	sport
PARAM	tcp	995	

PPtP Point-to-Point Tunneling Protocol

Action	proto	dport	sport
PARAM	47		
PARAM	tcp	1723	

Ping ICMP echo request

Action	proto	dport	sport
PARAM	icmp	echo-request	

PostgreSQL PostgreSQL server

Action	proto	dport	sport
PARAM	tcp	5432	

Printer Line Printer protocol printing

Action	proto	dport	sport
PARAM	tcp	515	

RDP Microsoft Remote Desktop Protocol traffic

Action	proto	dport	sport
PARAM	tcp	3389	

RIP Routing Information Protocol (bidirectional)

Action	proto	dport	sport
PARAM	udp	520	

RNDC BIND remote management protocol

Action	proto	dport	sport
PARAM	tcp	953	

Razor Razor Antispam System

Action	proto	dport	sport
ACCEPT	tcp	2703	

Rdate Remote time retrieval (rdate)

Action	proto	dport	sport
PARAM	tcp	37	

Rsync Rsync server

Action	proto	dport	sport
PARAM	tcp	873	

SANE SANE network scanning

Action	proto	dport	sport
PARAM	tcp	6566	

SMB Microsoft SMB traffic

Action	proto	dport	sport
PARAM	udp	135,445	
PARAM	udp	137:139	
PARAM	udp	1024:65535	137
PARAM	tcp	135,139,445	

SMBswat Samba Web Administration Tool

Action	proto	dport	sport
PARAM	tcp	901	

SMTP Simple Mail Transfer Protocol

Action	proto	dport	sport
PARAM	tcp	25	

SMTPS Encrypted Simple Mail Transfer Protocol

Action	proto	dport	sport
PARAM	tcp	465	

SNMP Simple Network Management Protocol

Action	proto	dport	sport
PARAM	udp	161:162	
PARAM	tcp	161	

SPAMD Spam Assassin SPAMD traffic

Action	proto	dport	sport
PARAM	tcp	783	

SSH Secure shell traffic

Action	proto	dport	sport
PARAM	tcp	22	

SVN Subversion server (svnserve)

Action	proto	dport	sport
PARAM	tcp	3690	

SixXS SixXS IPv6 Deployment and Tunnel Broker

Action	proto	dport	sport
PARAM	tcp	3874	
PARAM	udp	3740	
PARAM	41		
PARAM	udp	5072,8374	

Squid Squid web proxy traffic

Action	proto	dport	sport
PARAM	tcp	3128	

Submission Mail message submission traffic

Action	proto	dport	sport
PARAM	tcp	587	

Syslog Syslog protocol (RFC 5424) traffic

Action	proto	dport	sport
PARAM	udp	514	
PARAM	tcp	514	

TFTP Trivial File Transfer Protocol traffic

Action	proto	dport	sport
PARAM	udp	69	

Telnet Telnet traffic

Action	proto	dport	sport
PARAM	tcp	23	

Telnets Telnet over SSL

Action	proto	dport	sport
PARAM	tcp	992	

Time RFC 868 Time protocol

Action	proto	dport	sport
PARAM	tcp	37	

Trcrt Traceroute (for up to 30 hops) traffic

Action	proto	dport	sport
PARAM	udp	33434:33524	
PARAM	icmp	echo-request	

VNC VNC traffic for VNC display's 0 - 99

Action	proto	dport	sport
PARAM	tcp	5900:5999	

VNCL VNC traffic from Vncservers to Vncviewers in listen mode

Action	proto	dport	sport
PARAM	tcp	5500	

Web WWW traffic (HTTP and HTTPS)

Action	proto	dport	sport
PARAM	tcp	80	
PARAM	tcp	443	

Webcache Web Cache/Proxy traffic (port 8080)

Action	proto	dport	sport
PARAM	tcp	8080	

Webmin Webmin traffic

Action	proto	dport	sport
PARAM	tcp	10000	

Whois Whois (nicname, RFC 3912) traffic

Action	proto	dport	sport
PARAM	tcp	43	

Appendix E

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.)

The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading

or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION You may not copy, modify, sublicense, or distribute the Document except as expressly

provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.