PROXMOX VE

用户手册

Ver:6.0.0

编辑记录

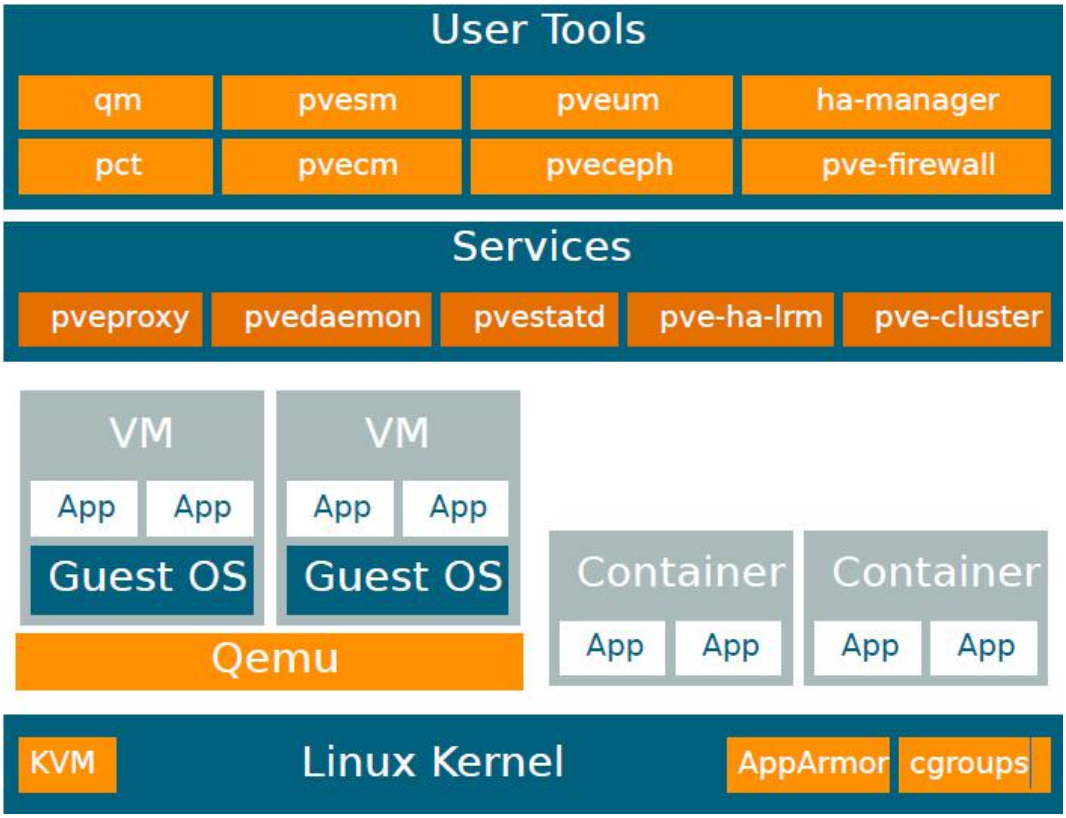
翻译版本	贡献人	完成时间	对应英文版本
V1.0	漠眼镜	2017-02-16	V4.4 December 9, 2016
V1.01	Tinkering	2017-06-13	V4.4 December 9, 2016
V5.2.0	漠眼镜	2019-02-20	V5.2 May 16, 2018
V5.3.0	漠眼镜	2019-06-03	V5.3 November 29, 2018
V5.4.0	漠眼镜	2019-06-18	V5.4 April 8, 2019
V6.0.0	漠眼镜	2019-12-10	V6.0 July 15, 2019

目录

第 1 章 简介

Proxmox VE 是一个既可以运行虚拟机也可以运行容器的虚拟化平台。Proxmox VE 基于 Debian Linux 开发，并且完全开源。出于灵活性的考虑，Proxmox VE 同时支持两种虚拟化技术：KVM 虚拟机和 LXC 容器。

Proxmox VE 的一个重要设计目标就是尽可能简化管理员的工作。你既可以用单机模式使用 Proxmox VE，也可以组建多节点 Proxmox VE 集群。所有的管理工作都可以通过基于 web 页面的管理界面完成，即使是一个小白用户也可以在几分钟内上手安装使用 Proxmox VE。



1.1 集中管理

尽管很多人一开始都使用单机方式运行 Proxmox VE，但实际上 Proxmox VE 可以横向扩展为一个拥有大量节点的集群。Proxmox VE 的默认安装方式中就已经包含了全套的集群套件。

独特的多主集群架构

内嵌的 WebGUI 管理控制台可以让你纵览所有的 KVM 虚拟机、LXC 容器和整个集群。你也

可以通过 WebGUI 轻松管理你的虚拟机、容器、存储和集群。完全没有必要另外安装单独的管理服务器。

Proxmox 集群文件系统（pmxcfs）

Proxmox VE 使用专门设计的基于数据库的 Proxmox 文件系统（pmxcfs）保存配置文件。这个文件系统足以让你保存几千台虚拟机的配置信息，并且能够通过 corosync 将配置文件实时复制到 Proxmox VE 集群的所有节点。Proxmox 文件系统一方面将所有数据都保存在服务器磁盘的一个数据库文件上，以避免数据丢失，另一方面在内存里也复制了一个副本，以提高性能。其中内存副本的最大容量为 30M，虽然不是很大，但足以保存几千台虚拟机配置信息。
截至目前，Proxmox 是唯一利用这种集群文件系统管理配置信息的虚拟化平台。

基于 Web 的管理界面

Proxmox VE 的使用很简单。管理操作可以通过内嵌的 WebGUI 完成—不需要专门安装管理工具或基于大型数据库的管理服务器节点。多主集群架构能够让你通过任意节点管理整个集群。基于 JavaScript 框架（ExtJS）开发的集中 Web 管理界面不仅能够让你通过 GUI 界面控制一切功能，而且可以浏览每个节点的历史活动和 syslog 日志，例如虚拟机备份恢复日志、虚拟机在线迁移日志、HA 活动日志等。

命令行工具

对于那些用惯了 Unix Shell 或 Windows Powershell 的高级用户，Proxmox VE 提供了一个命令行界面，可以管理虚拟化环境里的全部组件。这个命令行工具不仅有 Tab 键补全功能，而且提供了完善的 Unix man 形式的技术文档。

REST API

Proxmox VE 使用了 RESTful 形式的 API。开发人员选用 JSON 作为主要数据格式，所有的 API 定义均采用 JSON 语法。第三方管理工具很容易就可以将 Proxmox VE 的 API 集成进去。

基于角色的权限管理

在 Proxmox VE 中你可以用基于角色的方法对所有对象（包括虚拟机、存储、节点等等）设置用户管理权限。你可以定义权限，并控制对每个对象的访问。Proxmox VE 的权限管理方式实际上类似于访问控制列表：每个权限都针对特定主体（用户或用户组），每个角色（一组权限）都被限制在特定目录。

多种身份认证

Proxmox VE 支持多种用户身份认证方法，具体包括 Microsoft 活动目录，LDAP，Linux 系统用户认证，Proxmox VE 内嵌身份认证。

1.2 支持多种存储类型

Proxmox VE 支持多种存储技术。虚拟机镜像既可以保存在服务器本地存储，也可以保存在基于 NFS 或 SAN 的共享存储设备上。你可以根据需要自由地为 Proxmox VE 配置多种存储。实际上，Debian Linux 支持的所有类型的存储技术都可以用于 Proxmox VE。
用共享存储来保存虚拟机镜像有一个很大的好处，那就是 Proxmox VE 集群中的所有节点都可以直接访问到该虚拟机镜像，虚拟机就可以从一个 Proxmox VE 节点在线迁移到其他节点运行，并且虚拟机在迁移过程中可以保持连续运行，无需关机。
Proxmox VE 目前支持的网络共享存储类型如下：

- LVM 卷组（基于 iSCSI 网络存储）
- iSCSI 网络存储设备
- NFS 共享存储
- CIFS 共享存储
- Ceph RBD
- iSCSI 卷
- GlusterFS

支持的本地存储类型如下：

- LVM 卷组（基于本地磁盘、FC 磁盘、DRBD 等）
- 目录（本地文件系统）
- ZFS

1.3 虚拟机备份与恢复

Proxmox VE 内嵌了虚拟机备份工具（vzdump），可以在线创建 KVM 虚拟机和 LXC 容器的快照备份。创建的备份不仅包括虚拟机和容器的完整镜像数据，同时包含了相应的配置文件信息。
KVM 虚拟机在线备份功能兼容所有的存储类型，对于保存在 NFS、CIFS、iSCSI LUN、Ceph RBD 上的虚拟机镜像，均可以进行备份。目前新的备份文件格式进行过特别优化，确保备份过程的高效和快速（优化内容包括稀疏磁盘镜像文件、非连续镜像文件数据、最小化 I/O 等）。

1.4 高可用集群

多节点 Proxmox VE HA 集群支持用户自定义配置高可用的虚拟机。Proxmox VE HA 集群基于久经考验的 Linux HA 技术，能够提供稳定可靠的 HA 服务。

1.5 支持多种虚拟网络技术

Proxmox VE 支持基于桥接模式的虚拟网络。在该模式下，所有的虚拟机共享一个虚拟交换机，效果就相当于所有的虚拟机同时接入了同一个交换机一样。虚拟交换机还可以和 Proxmox VE 的物理网卡桥接，以便相关虚拟机和外部网络进行 TCP/IP 通讯。
此外，Proxmox VE 还支持 VLANs（802.1q）和网卡绑定/链路聚合技术。用户可以充分利用 Linux 网络组件的强大功能，在 Proxmox VE 服务器上构建非常复杂和多样的虚拟网络环境。

1.6 内嵌防火墙

你可以利用 Proxmox VE 的内嵌防火墙对任意虚拟机和容器的网络通信流量进行过滤。还可以利用“Security groups”把常用防火墙策略和集合分组管理。

1.7 开源的原因

Proxmox VE 使用 Linux 作为内核，并且基于 Debian GNU/Linux 构建用户空间组件。Proxmox VE 的源代码基于 GNU Affero General Public License, version 3 发布。这确保你可以在任何时候都可以自由查看 Proxmox VE 源代码，并向该项目共享代码。
在 Proxmox 上我们坚持使用开源软件。使用开源软件不仅能确保能使用所有功能，还保证了软件的安全和可靠。我们认为每个人都有权访问软件的源代码，以便于更好的运行软件、打包软件、为软件提交新的代码。我们鼓励每个人向 Proxmox VE 贡献代码，同时我们将确保 Proxmox VE 始终保持专业软件的品质。
开源软件还能帮助你节约成本，避免你的基础设施产生单一厂商依赖问题。

1.8 Proxmox VE 的优势

- 开源软件
- 没有单一厂商依赖
- Linux 内核

- 快速安装，易于使用
- 基于 Web 的管理界面
- REST API
- 庞大而活跃的社区
- 很低的管理和部署成本

1.9 获取支持

1.9.1 Proxmox VE Wiki

Proxmox VE 技术资料的一个主要来源就是 Proxmox VE Wiki。其中既有官方参考文档，也有用户贡献的内容。

1.9.2 社区支持论坛

Proxmox VE 本身是完全开源的。我们鼓励用户在 Proxmox VE Community Forum 讨论和分享关于 Proxmox VE 的知识。这个论坛完全由 Proxmox 支持团队主持，并且拥有来自全世界的众多用户群体。毫无疑问，这个论坛是一个非常棒的获取 Proxmox VE 相关信息的途径。

1.9.3 电子邮件列表

通过电子邮件快速访问 Proxmox VE 社区的方式如下。

- 用户电子邮件列表：PVE User List

Proxmox VE 开发者的主要通讯频道如下。

- 开发者电子邮件列表：PVE development discussion

1.9.4 商业支持

Proxmox Server Solutions Gmbh 提供了 Proxmox VE 商业支持服务 Proxmox VE Subscription Service Plan。在发生问题时，订阅了 Proxmox VE 服务的系统管理员可以通过专门的支持渠道寻求支持，并可以在规定的响应时间内获得 Proxmox VE 开发人员的帮助。要获取更多具体信息以及咨询折扣，请直接联系 Proxmox 销售团队 Proxmox sales team。

1.9.5 Bug 提交及跟踪

我们有一个公开的 Bug 跟踪系统 http://bugzilla.proxmox.com。如果你遇到了 bug，可以在这里创建相关 bug 记录。你可以通过这个系统跟踪 bug 状态，也可以在第一时间获得 bug 修复的消息。

1.10 项目历程

Proxmox VE 项目始于 2007 年，并在 2008 年发布了第一个 stable 版本。当时我们采用了 OpenVZ 容器技术和 KVM 虚拟机技术。但集群功能十分有限，用户界面也很简陋（页面由服务器生成）。
但我们很快用 Corosync 集群组件开发了新的集群功能，并通过引入新的 Proxmox 集群文件系统（pmxcfs）很好地对用户屏蔽了集群的复杂性。这是一个很大的进步。用户管理一个有 16 个节点的集群就和管理一个单机系统一样简单。
我们还引入了新的 REST API，并用 JSON 定义了所有的 API。借助 REST API，第三方不仅可以将 Proxmox VE 集成到他们现有的 IT 基础设施中，并且可以很容易地开发新的服务。
此外，利用新的 REST API，我们用基于 Javascript 的 HTML5 应用替换了原有的用户界面。我们还用 noVNC 替换了原来基于 Java 的 VNC 控制台组件。现在你只需要通过浏览器就可以直接访问虚拟机桌面。
支持不同类型的存储技术是 Proxmox VE 另一个重要特性。其中值得一提的是，在 2014 年 Proxmox VE 就默认支持 ZFS，这在 Linux 发行版中是第一个。另一个重要的里程碑是在 Proxmox VE 服务器上运行 Ceph 存储服务，从而提供了一种性价比极高的部署方式。
我们是最早提供 KVM 虚拟机商业软件技术服务的公司之一。KVM 技术本身在不断演进，目前已经是被广泛使用的虚拟机技术，而且随着每个新版本发布都会有新功能推出。我们开发了 KVM 虚拟机在线备份功能，从而可以对保存在任何存储设备上的 KVM 虚拟机进行快照式备份。
Proxmox VE 4.0 的一个重大变化就是舍弃了 OpenVZ 容器并转向了 LXC 容器技术。目前容器技术已经深度整合到了 Proxmox VE 当中，并且可以和虚拟机在同一个存储和网络环境中同时使用。

1.11 参与完善 Proxmox VE 文档

根据你想要改进的内容主题，可以用不同的方式提交给 Proxmox VE 开发人员。
如果你想修正当前文档中的错误，可以使用 Proxmox bug tracker 提交更正后的文档。
如果你想增加新的内容主题，则取决于你希望增加的文档内容类型：

- 如果想增加的内容仅限于你自己的部署环境，那么添加到 wiki 上是最合适的。例如，你想增加的内容是关于特定虚拟机的配置信息，比如是针对一个冷门操作系统的最佳 Qemu 驱动组合，那么就非常适合用 wiki 文章的形式来记录。

- 如果想增加的内容是关于一般性问题，并且对所有用户都会有帮助，你可以尝试添加到参考文档。参考文档使用 asciidoc 文档格式编写。你可以先克隆参考文档代码

仓库 git://git.proxmox.com/git/pve-docs.git，然后按照 README.adoc 中的指示来编写新的内容。

参与完善 Proxmox VE 文档就和在 Wikipedia 上写文章一样简单，并且也是参与大型开源软件项目的一种有趣的尝试。

➢ 注意

如果你有兴趣了解 Proxmox VE 代码仓库的使用，那么可以先从 wiki 文章 Development Documentation 开始。

1.12 翻译 Proxmox VE

Proxmox VE 有很多非英语国家的用户，为便于全世界用户使用，我们需要志愿者提供翻译服务。我们很高兴有本地用户将 Proxmox VE 翻译成自己的母语，并极力邀请这样的用户加入改善 Proxmox VE。
Proxmox VE 的语言文件位于 git 仓库。对于熟悉 git 的人，就可以按照开发文档来参加翻译了。
事实上，翻译本身并不需要特别的专业技能。无需配置开发环境，直接下载语言文件也可以。在语言文件对应的“raw”链接上点击鼠标右键，选择“另存为”即可。可以直接将翻译结果连同贡献者协议书一并发送给 office@proxmox.com。
我们使用 gettext 来翻译 Proxmox VE。因此，实际翻译工作就是翻译 msgid，并写到下面对应的 msgstr 处。类似 Poedit 的工具会更便于使用，特别对于没有编程经验的人更是如此。

第 2 章 Proxmox VE 安装

Proxmox VE 基于 Debian Linux 操作系统，官方提供有 ISO 光盘镜像，其中包含了一个完整的 Debian Linux 操作系统（Proxmox VE 5.x 使用的是"stretch"版本的 Debian）和 Proxmox VE 的所有基本软件包。
使用安装向导可以帮助完成整个安装过程，包括本地磁盘分区，基本系统设置（例如，时区，语言，网络），软件包安装等。使用官方 ISO 可以在几分钟内完成安装，这也是首推使用官方 ISO 安装的原因。
当然，也可以先安装 Debian 操作系统，然后再安装 Proxmox VE 软件包。但这种安装方法需要对 Proxmox VE 有很深入的了解，仅推荐高级用户使用。

2.1 系统安装需求

对于生产用 Proxmox VE 服务器，建议为服务器配置较好的硬件。时刻牢记，如果你在一台服务器上运行了 10 台虚拟机，那么一旦服务器硬件故障，那么这 10 台虚拟机就会全部当机。Proxmox VE 支持集群式部署，而利用内嵌的集群功能，可以实现对多台服务器的集中管理。
Proxmox VE 支持服务器本地磁盘（DAS），SAN，NAS 和分布式存储（Ceph DRBD）等多种虚拟机镜像存储技术。具体可详见第 8 章"Proxmox VE 存储"。

2.1.1 最小硬件配置，适用于测试评估场景

- CPU 要求为 Intel EMT64 或 AMD64，需要支持 Intel VT/AMD-V 虚拟化。
- 内存不低于 2GB，以确保操作系统和 Proxmox VE 服务正常运行。如需运行虚拟机，需相应增加更多内存。如需运行 Ceph 或 ZFS，还需要增配内存，大概 1TB 存储空间增加 1GB 内存。
- 高性能高冗余存储资源，最好使用 SSD 盘。
- 操作系统盘 :带有电池保护缓存的硬 RAID 卡 ,没有硬 RAID 卡时可以使用带有 SSD 缓存的 ZFS。
- 虚拟机存储：本地磁盘可以采用带有电池保护缓存的硬 RAID 卡，或不带硬 RAID 卡的 ZFS。ZFS 和 Ceph 都不能和硬 RAID 控制器同时使用。也可以共享分布式存储。

- 多块千兆网卡。根据所用存储技术和集群配置，可以选配更多网卡。也可使用 10Gbit 或更高速网卡。
- 如需使用 PCI 直通，必须采用支持 VT-d/AMD-d 的 CPU。

2.1.2 推荐系统硬件配置

- CPU：64 位（Intel EMT64 或 AMD64），推荐使用多核 CPU
- CPU 和主板需要支持 Intel VT/AMD-V 技术，以便支持 KVM 全虚拟化功能
- 内存：8GB，如果要运行虚拟机则应配置更多
- 硬 RAID 卡，带有电池保护（BBU）或闪存保护的写缓存
- 高性能硬盘，最好是 15k 转速的 SAS 盘，配置成 Raid10
- 最少 2 块以太网卡，也根据采用的共享存储技术配置更多网卡

2.1.3 性能概览

Proxmox VE 安装完成后，你可以运行 pveperf 命令查看 CPU 和硬盘性能概要。

➢ 注意

这仅仅是一个非常便捷且粗略的性能指标。建议你进行更多的性能测试，特别是在需要了解系统 I/O 性能指标时。

2.1.4 Web 管理界面支持的浏览器

- Firefox，当年发行的版本，或最新的扩展支持版本
- Chrome，当年发行的版本
- 微软目前支持的 Internet Explorer 版本（在 2019 年，指 IE11 或 IE Edge）
- Safari，当前发布版本

如果 Proxmox VE 检测到你在使用移动终端访问，则会跳转到轻量级的专为触摸屏设计的管理界面。

2.2 使用官方 ISO 光盘镜像安装 Proxmox VE

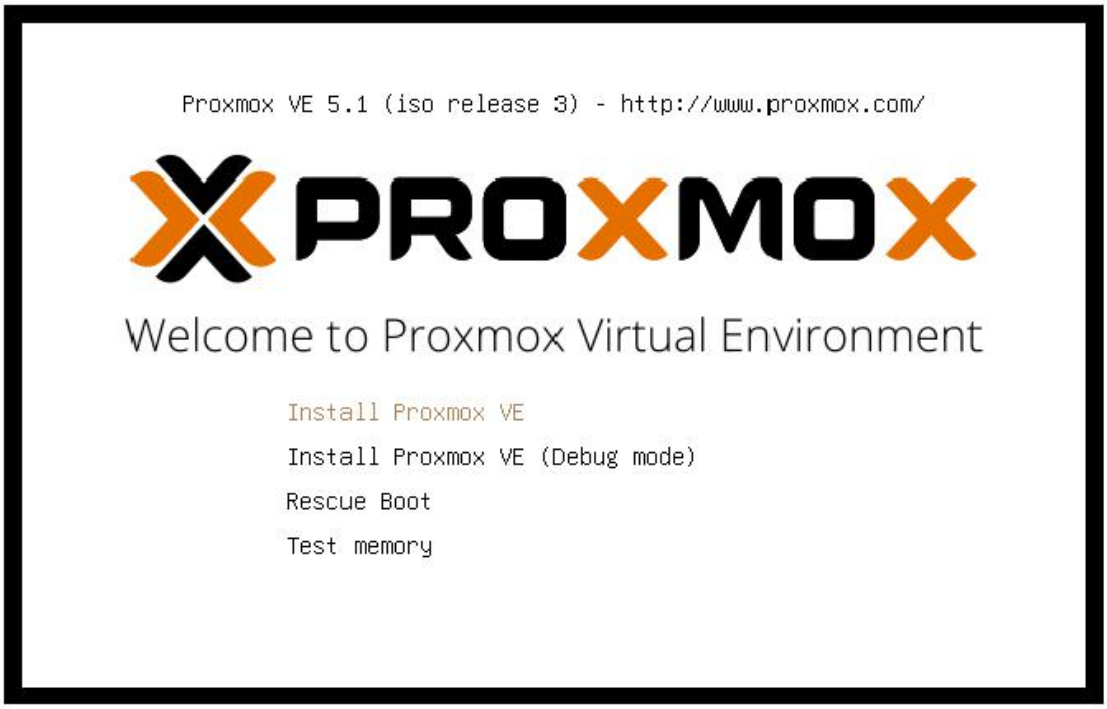
可以从 http://www.proxmox.com/en/downloads 下载 ISO 镜像。ISO 镜像中包含以下内容：

- 完整的操作系统（64 位 Debian Linux）
- Proxmox VE 安装程序，用于硬盘分区及操作系统安装。目前支持的文件系统有 ext4、ext3、xfs 和 ZFS
- 集成有 LXC 和 KVM 的 Proxmox VE 内核
- 用于管理虚拟机、容器及其他 IT 资源的全套管理工具
- Web 管理界面，以便于用户使用以上管理工具。

➢ 注意

默认情况下，Proxmox VE 会格式化服务器的所有硬盘，原有硬盘数据都会被格式化掉。

请将安装介质（例如，U 盘或 CD-ROM 光盘）插入计算机，并从安装介质引导启动。



选择 Proxmox VE 安装介质对应的启动项（例如，从 USB 启动）后，就可以看到 Proxmox VE 菜单，接下来可以选择的有：

- Install Proxmox VE

正常启动安装程序。

➢ 提示

可以通过键盘来操作安装向导。只要组合按下 ALT 键和按钮上带下划线的字母键即可进行操作。例如，按下组合键 ALT+N 相当于点击 Next 按钮。

- Install Proxmox VE（Debug mode）

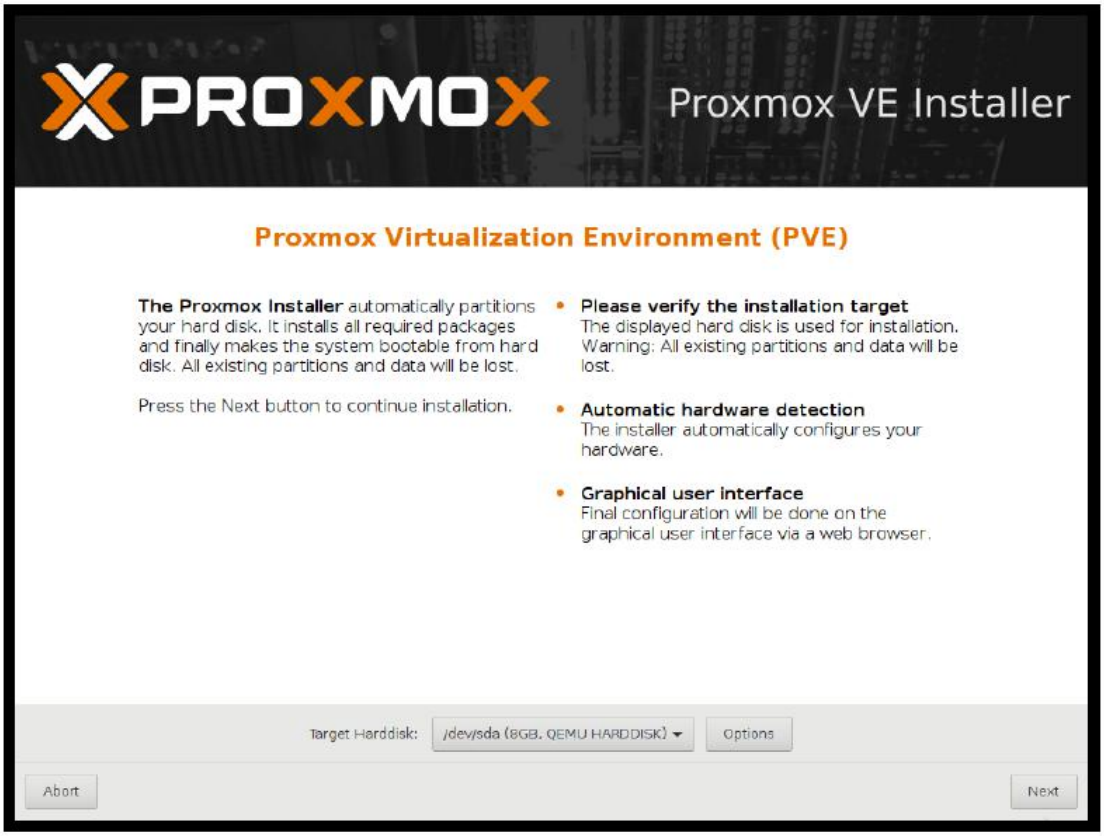
以调试模式启动安装程序。安装程序会在安装过程中数次启动命令行 shell 控制台，以便你对安装过程中的错误进行调试和检测。当调试完毕后，可以按组合键 Ctrl+D 退出 shell 调试控制台，系统将继续进行下一个安装步骤。这个选项主要供开发人员使用，不建议普通用户选择该选项安装 Proxmox VE。

- Resuse Boot

该选项允许你启动服务器上之前安装的 Proxmox VE 系统。引导程序将搜索服务器硬盘，如果发现之前安装的 Proxmox VE，将直接启动该硬盘上安装的 Proxmox VE。这个选项对于修复引导扇区（grub）错误，或是在 BIOS 无法读取引导扇区时非常有用。

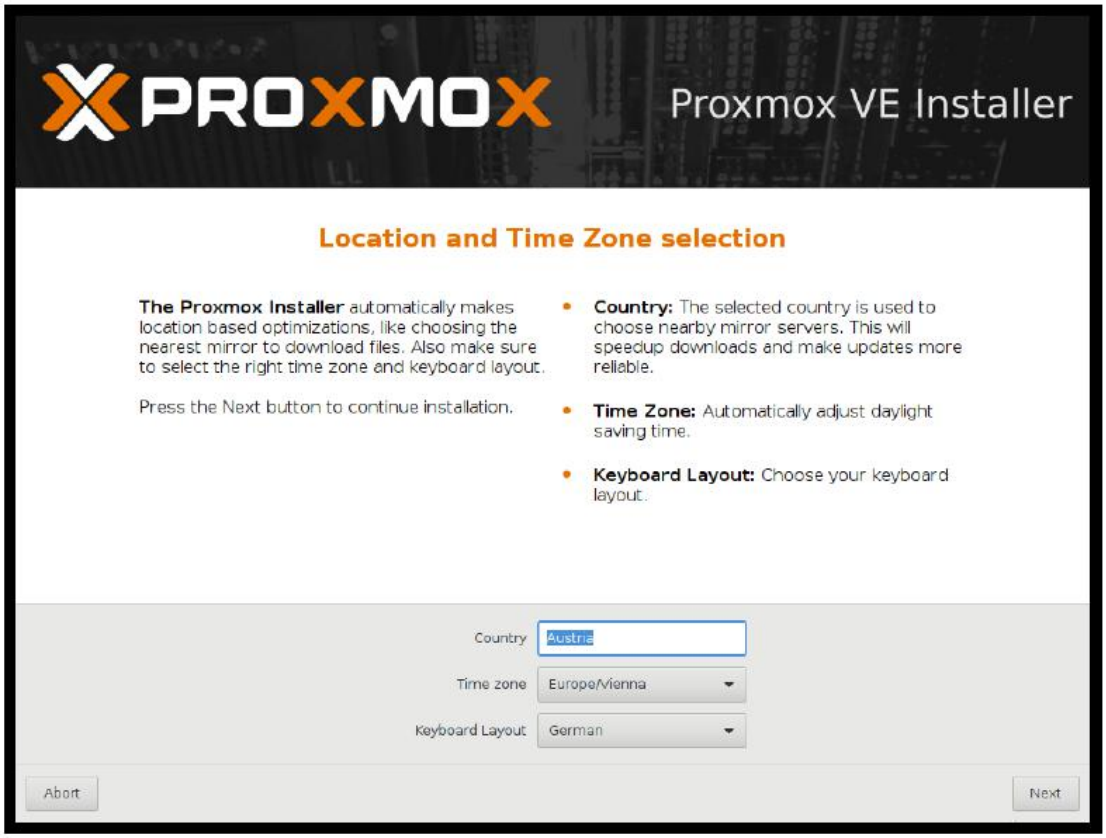
- Test Memory

该选项将启动 memtest86+程序，以检测服务器内存是否正常。

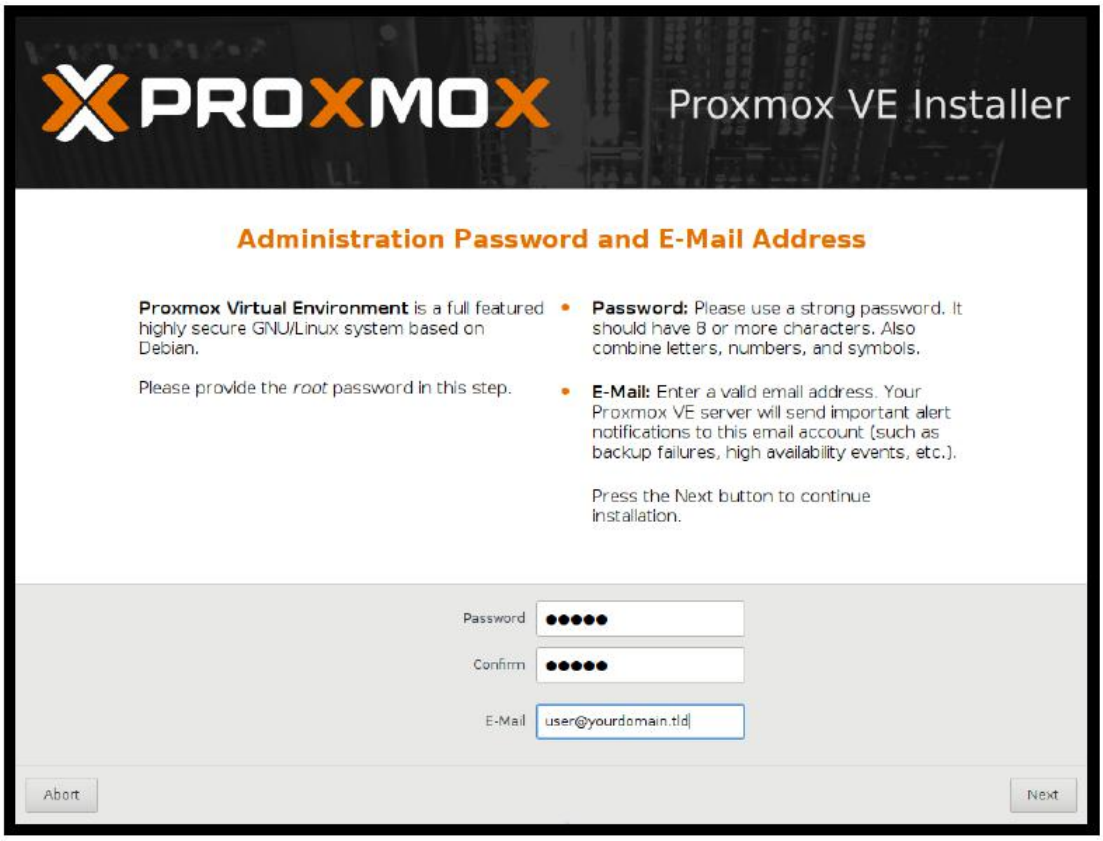


通常情况下都是选择 Install Proxmox VE 菜单并开始安装过程。安装程序随后会提示你选择

安装的目标硬盘。Options 按钮用于设置目标硬盘所用的文件系统类型，默认采用的是 ext4 文件系统。如果选择使用 ext3、ext4 或 xfs 文件系统，安装目标硬盘将被格式化为 LVM 卷组，并可进一步设置 LVM 的空间大小（见 2.2.1 节）。
推荐使用 ZFS 文件系统。ZFS 提供多种级别软 RAID，如果服务器没有 RAID 卡硬件，那么 ZFS 将会特别方便。通过 Options 按钮可以设置 ZFS 的 RAID 级别，并在磁盘列表里根据需要选择硬盘设备组成 ZFS 文件系统。此外，ZFS 还提供其他丰富的参数项（见 2.2.2 节）。



安装程序接下来的配置页面主要用于设置一些基本配置，例如你的位置、时区和键盘布局。位置信息用于选择离你最近的源服务器，以便加快升级更新速度。通常安装程序能够自动检测有关设置，除非自动检测失败，或者你想要使用某种特殊的键盘布局设置，否则你几乎不需要手工调整这些设置。



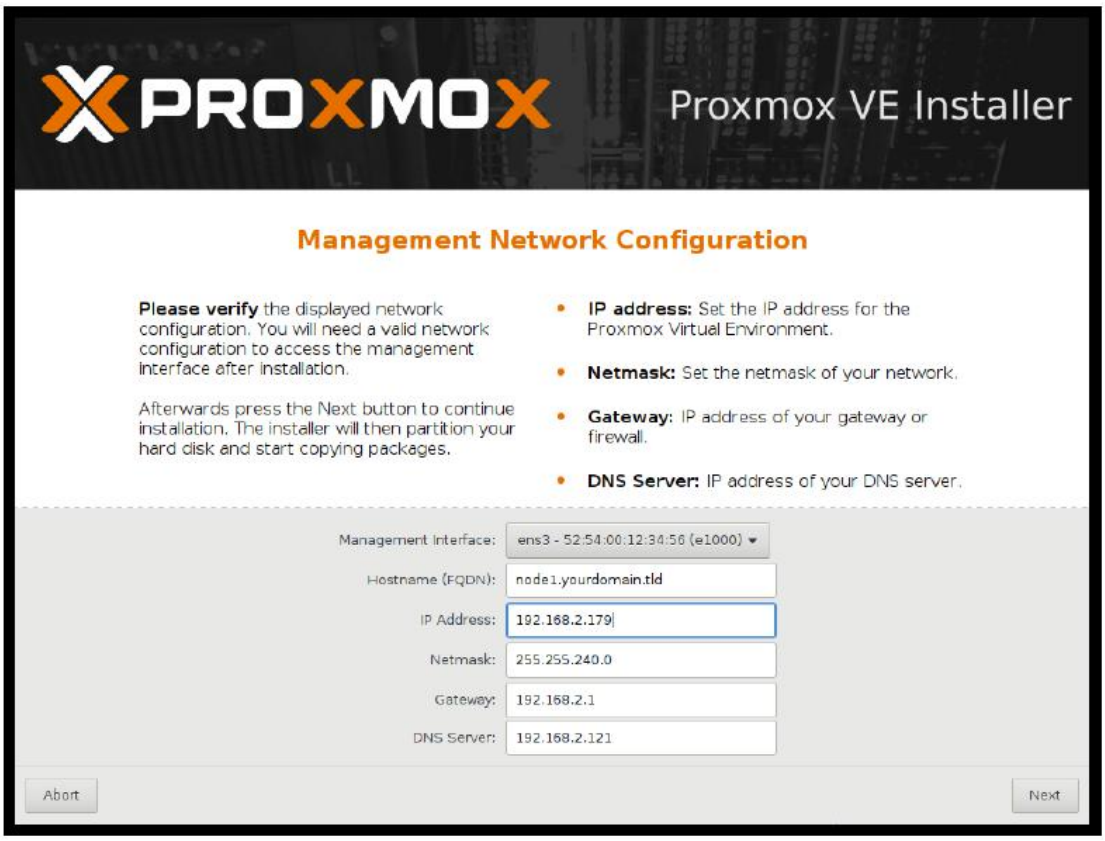
接下来你需要设置电子邮件地址和超级用户（root）口令。口令长度不得少于 5 个字符，我们强烈推荐设置复杂的高强度口令，以下是一些关于设置强口令的指引：

- 口令长度至少要 12-14 个字符
- 口令由小写字母、大写字母、数字和特殊字符混合构成
- 避免使用重复字符，键盘位置连续的字符，单词，连续的字母或数字序列，用户名，亲戚或宠物名字，恋人（不管是现任还是前任）或其他的一些身份信息（例如身份证号码，父母姓名或生日）。

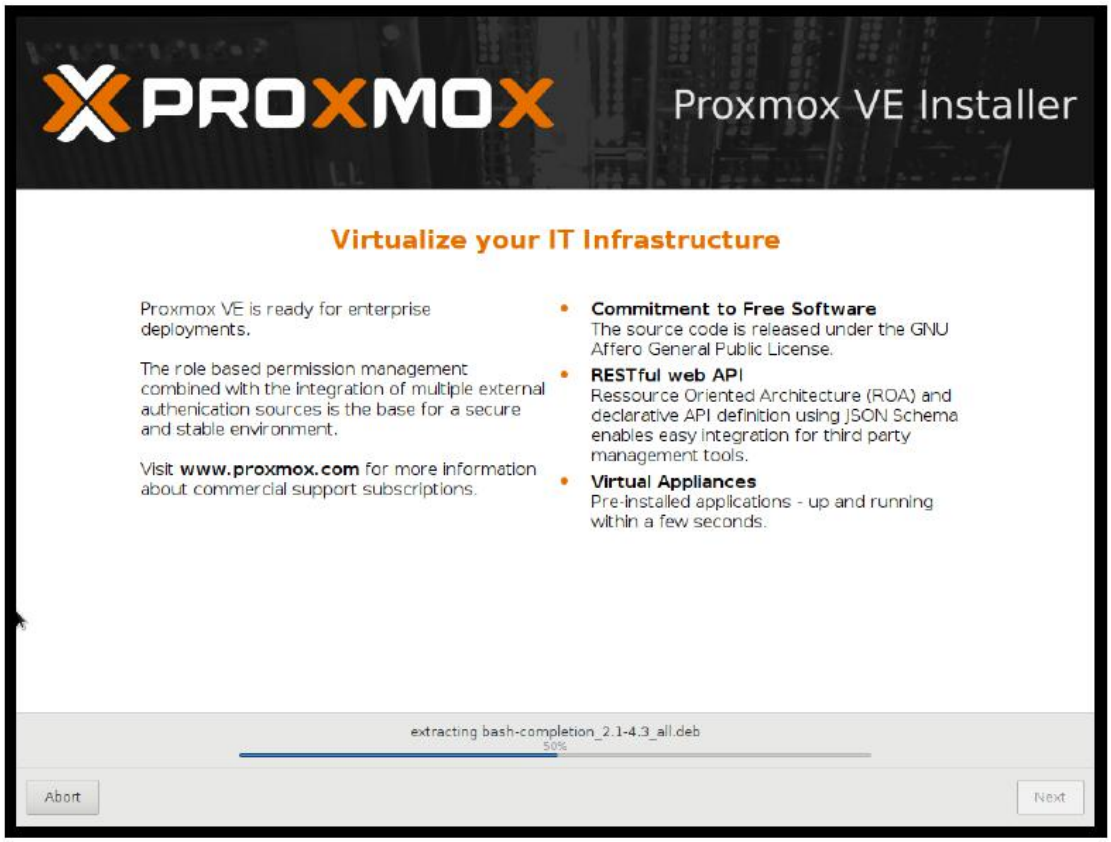
系统会自动给系统管理员发送一些提示信息，例如：

- 可用的升级软件包信息
- 定时调度任务错误告警信息

所有这些提示信息都会自动发送到设置的电子邮件地址。



安装程序的最后一个步骤是进行网络设置。你可以使用 IPv4 或 IPv6 其中一个，但不能选择同时使用两种协议。如果你确实需要同时使用两种 IP 网络协议，可以在安装完成后再进行配置。



现在，如果你点击 Next 按钮，安装程序将自动开始硬盘格式化和软件包复制安装过程。待安装程序完成，取出安装盘，重启服务器即可。
进一步的配置可以在 Proxmox 的 Web 管理界面中进行。只需要在你的浏览器中输入并打开安装时设置的服务器 IP 地址即可（https://你的服务器 IP 地址:8006）。

➢ 注意

首次登录默认用户是 root（认证域为 realm PAM），口令为安装过程中设置的口令。

2.2.1 LVM 高级配置

Proxmox VE 会创建一个名为 pve 的 LVM 卷组（VG），以及三个逻辑卷（LVs），分别为 root，data，swap。这些逻辑卷的容量大小可以通过如下参数设置：

- hdsize

用于设置 VG 使用的硬盘空间大小。通过设置这个参数，你可以配置硬盘留出部分空间用作其他目的（例如在同一块硬盘上配置其他物理卷 PV 和卷组 VG，以创建 LVM 类型存储服务）

- swapsize

用于设置 swap 逻辑卷的容量大小。默认和服务器物理内存容量大小一致，但最小不低于 4GB，最大不超过 hdsize/8。

➢ 注意

如设置为 0，将不会创建 swap 卷。

- maxroot

用于设置 root 逻辑卷的容量大小。root 卷用于保存 Proxmox VE 操作系统镜像。

- maxvz

用于设置 data 逻辑卷的容量大小。

➢ 注意

使用 LVM thin 模式时，只有 datasize 大于 4GB 时才会创建 data 卷。

➢ 注意

如设置为 0，将不会创建 data 卷，并将相应自动调整存储配置。

- minfree

用于设置 pve 卷组的预留空间大小。如果 pve 卷组总容量大小超过 128GB，则使用默认值 16GB，否则设置为 hdsize/8。

➢ 注意

LVM 预留空间用于创建快照（如使用 lvmthin，则无须设置该参数）。

2.2.2 ZFS 高级配置参数

安装程序会自动创建名为 rpool 的 ZFS 存储池。使用 ZFS 时，默认不会创建 swap 空间。可以预留部分未分区空间以创建 swap，也可以在安装完成后手工创建一个 swap zvol 卷，但是要注意手工创建 swap 可能会导致问题。

- ashift

用于设置存储池的 ashift 值。ashift 值不能小于存储池中的磁盘扇区大小（2 的 ashift 次幂就是扇区大小），或任何新加入存储池的磁盘扇区大小（例如，用新磁盘替换故障磁盘时）。

- compress

用于设置是否启用 rpool 的压缩功能。

- checksum

用于设置 rpool 的校验算法类型。

- copies

用于设置 rpool 的 copies 参数值。该参数不能代替磁盘冗余功能，具体原因以及配置语法参见 man 手册的 zfs（8）页面。

- hdsize

用于设置总的 HD 大小。通过设置该参数，可以在 HD 上预留空间用于其他用途（例如，创建 swap 分区）。hdsize 仅对启动盘有效，例如，RAID0，RAID1 或 RAID10 中的第一个磁盘镜像，或 RAID-Z[123]中的所有磁盘。

2.2.3 ZFS 性能提示

ZFS 非常消耗内存资源，所以如果要使用 ZFS 作为存储，服务器需要尽量多配置内存。最佳实践是为以 4GB 为基础，每 TB 裸磁盘容量增加 1GB 容量内存。

ZFS 支持使用高性能 SSD 盘作为写缓存。该写缓存称为 ZFS Intent Log（ZIL）。你可以在

Proxmox VE 安装完成后使用如下命令添加 ZIL
zpool add <pool-name> log </dev/path_to_fast_ssd>

2.3 用 U 盘安装 Proxmox VE

目前官方提供的 Proxmox VE 安装介质是一种混合型的 ISO 镜像。有两种使用方法：

- 将 ISO 镜像烧录到 CD 上使用
- 将裸区块镜像（IMG）文件直接复制到闪存介质上（USB 盘）

用 U 盘安装 Proxmox VE 不仅速度快而且更加方便，也是推荐使用的安装方式。

2.3.1 制作 U 盘安装介质

需要将 ISO 镜像复制到 U 盘上，以便从 U 盘启动安装。
首先下载 ISO 镜像，下载地址为
https://www.proxmox.com/en/downloads/category/iso-images-pve
U 盘容量至少为 1GB。

➢ 注意

不要用 UNetbootin 或 Rufus。

☒ 重要

请确保 U 盘没有被挂载，并且没有任何重要数据。

2.3.2 GNU/Linux 下的制作过程

你可以直接用 dd 命令制作 U 盘镜像。首先下载 ISO 镜像，然后将 U 盘插入计算机。找出 U 盘的设备名，然后运行如下命令：
dd if=proxmox-ve_* .iso of=/dev/XYZ bs=1M

➢ 注意

请用正确的设备名替换上面命令中的/dev/XZY。

☒ 警告

请务必小心，不要把硬盘数据覆盖掉！

如何找到 U 盘的设备名

你可以比较 U 盘插入计算机前后 dmesg 命令输出的最后一行内容，也可以用 lsblk。
打开命令行终端，运行命令
lsblk
然后将 U 盘插入计算机，再次运行命令
lsblk
你会发现有新的设备，这个新设备就是你所要操作的 U 盘。

2.3.3 OSX 下的制作过程

打开命令行终端（在 Spotlight 中 query Terminal）。
用 hdiutil 的 convert 选项将.iso 文件转换为.img 格式，示例如下。
hdiutil convert -format UDRW -o proxmox-ve_*.dmg proxmox-ve_*.iso

---

➢ 提示

OS X 倾向于自动为输出文件增加.dmg 后缀名。

运行命令获取当前设备列表：
diskutil list
然后将 U 盘插入计算机，再次运行命令，获取分配给 U 盘的设备节点名称（例如 /dev/diskX）。
diskutil list
diskutil unmountDisk /dev/diskX

➢ 注意

用前面命令中返回的设备序号替换 X。

---

sudo dd if=proxmox-ve_*.dmg of=/dev/rdiskX bs=1m

➢ 注意

前面命令使用了 rdiskX 而不是 diskX，这可以提高写入速度。

2.3.4 Windows 下的制作过程

从 https://etcher.io 下载 Etcher，选择 ISO 和你的 U 盘。
如不成功，可改用 OSForenics USB installer，下载地址为 http://www.osforensics.org/portability.html

2.4.5 用 U 盘引导启动服务器

将制作好的 U 盘介质插入服务器，确保服务器能从 USB 启动（查看服务器 BIOS 设置）。然后根据安装向导指示安装 Proxmox VE。

2.3 在 Debian 系统上安装 Proxmox VE

Proxmox VE 以 Debian 软件包形式打包，所以你可以先安装 Debian 操作系统，然后再安装 Proxmox VE。首先按 3.1 节设置软件源，然后运行安装命令：

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

在 Debian 操作系统上安装 Proxmox VE 看起来很简单，但实际上你需要先按要求安装 Debian 系统，并且自行配置服务器本地硬盘的用途。此外，网络配置也完全由你自行手工完成。总之，这种安装方式根本不简单，如果你计划使用 LVM 或 ZFS 会更加复杂。

你可以在 wiki 上找到一个详细的安装步骤。

第 3 章 Proxmox VE 服务器管理

Proxmox VE 基于著名的 Debian Linux 发行版。也就是说，你可以充分利用 Debian 操作系统的所有软件包资源，以及 Debian 完善的技术文档。可以在线阅读 Debian Administrator's Handbook，深入了解 Debian 操作系统的有关内容（见[Hertzorg13]）。
Proxmox VE 安装后默认配置使用 Debian 的默认软件源，所以你可以通过该软件源直接获取补丁修复和安全升级。此外，我们还提供了 Proxmox VE 自己的软件源，以便升级 Proxmox VE 相关的软件包。这其中还包括了部分必要的 Debian 软件包升级补丁。
我们还为 Proxmox VE 提供了一个专门优化过的 Linux 内核，其中开启了所有必需的虚拟化和容器功能。该内核还提供了 ZFS 相关驱动程序，以及多个硬件驱动程序。例如我们在内核中包含了 Intel 网卡驱动以支持最新的 Intel 硬件设备。
后续章节将集中讨论虚拟化相关内容。有些内容是关于 Debian 和 Proxmox VE 的不同之处，有些是关于 Proxmox VE 的日常管理任务。更多内容请参考 Debian 相关文档。

3.1 软件源

所有基于 Debian 的操作系统都使用 APT 命令作为软件包管理工具。软件源列表定义在/etc/apt/sources.list 文件中，以及/etc/apt/sources.d 目录下后缀名为.list 的文件中。既可以直接使用 apt-get 命令升级软件，也可以使用 GUI 界面升级。
软件源文件 sources.list 的每一行都定义了一个软件源，最常用的软件源一般放在前面。在 sources.list 中，空行会被忽略，字符#及以后的内容会被解析为注释。可以用 apt-get update 命令获取软件源中的软件包信息。

/etc/apt/souces.list 文件

```
deb http://ftp.debian.org/debian buster main contrib
deb http://ftp.debian.org/debian buster-updates main contrib
# security updates
deb http://security.debian.org buster/updates main contrib.
```

此外，Proxmox VE 还提供了另外三个软件源。

3.1.1 Proxmox VE 企业版软件源

Proxmox VE 企业版软件源是默认的、稳定的、推荐使用的软件源，供订阅了 Proxmox VE 企业版的用户使用。该软件源包含了最稳定的软件包，适用于生产环境使用。软件源 pve-enterprise 默认是启用的。

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

deb https://enterprise.proxmox.com/debian buster pve-enterprise

一旦有软件包有新的升级，root@pam 用户就会收到有关新软件包的电子邮件通知。在 GUI 界面，可以查看每个软件包的变更历史（如果有的话），其中有升级的每个细节。所以你永远不会错过重要的安全补丁。

请注意，你必须提供订阅密钥才可以访问企业版软件源。我们提供有不同级别的订阅服务，具体信息可以查看网址 https://www.proxmox.com/en/proxmox-ve/pricing。

➢ 注意

如果你没有订阅 Proxmox VE 企业版，可以将企业版软件源配置信息在软件源配置文件中注释掉（在该行开头插入一个#字符），以避免系统发出错误提示信息。这种情况下可以配置使用 pve-no-subscriptin 软件源。

3.1.2 Proxmox VE 免费版软件源

顾名思义，你无需付费订阅即可访问 Proxmox VE 免费版软件源。该版本适用于测试或非生产环境。我们不推荐在生产环境中使用 Proxmox VE 免费版，因为该版本的软件包往往没有经过足够充分的测试和验证。

我们推荐将免费版软件源配置在/etc/apt/sources.list 文件中。

/etc/apt/sources.list 文件

```
deb http://ftp.debian.org/debian buster main contrib
deb http://ftp.debian.org/debian buster-updates main contrib
# PVE pve-no-subscription repository provided by proxmox.com,
# NOT recommended for production use
deb http://download.proxmox.com/debian/pve buster pve-no-subscription
# security updates
deb http://security.debian.org buster/updates main contrib
```

3.1.3 Proxmox VE 测试版软件源

最后，我们还提供了一个名为 pvetest 的测试版软件源。这个版本的软件源包含了最新版本的软件包，主要供开发人员测试新功能和新特性使用。同样，你可以在配置文件/etc/apt/sources.list 中设置该软件源，如下：

在 sources.list 中配置 pvetest 的示例

```
deb http://download.proxmox.com/debian/pve buster pvetest
```

☒ 警告

pvetest 软件源仅供新功能测试和 bug 修复补丁测试使用（顾名思义）。

3.1.4 Proxmox VE Ceph 软件源

Proxmox VE 的 Ceph 主软件源包含有 Ceph 相关软件包，可用于生产环境。也可以用该软件源更新你的 Ceph client。

/etc/apt/sources.list.d/ceph.list 文件

deb http://download.proxmox.com/debian/ceph-nautilus buster main

3.1.5 Proxmox VE Ceph 测试软件源

Proxmox VE 的 Ceph 测试软件源包含有等待进入主软件源的 Ceph 软件包，主要用于测试新版 Ceph。

/etc/apt/sources.list.d/ceph.list 文件

deb http://download.proxmox.com/debian/ceph-nautilus buster main

3.1.6 Proxmox VE Ceph Luminous 升级软件源

该软件源针对 Proxmox VE 6.0 提供 Ceph Luminous 版本软件包。可用于将 Proxmox VE 集群升级到 6.0 版，操作系统升级至 Debian Buster，同时保持 Ceph Luminous 不变，并可在后续根据需要再行升级。

/etc/apt/sources.list.d/ceph.list 文件

deb http://download.proxmox.com/debian/ceph-luminous buster main

3.1.7 SecureApt

我们使用 GnuPG 为以上软件源中的 Release 文件做了电子签名，APT 通过该电子签名来验证并确保相关软件包来自可信任的软件源。
如果你是用官方 ISO 镜像安装的 Proxmox VE，那么电子签名验证密钥随会一并安装到服务器。如果你使用其他方式安装 Proxmox VE，则可以手动下载验证密钥。下载命令如下：

```
# wget http://download.proxmox.com/debian/proxmox-ve-release-6.x.gpg  -
-O /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
```

并可以用如下命令检查密钥
sha512sum /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
acca6f416917e8e11490a08a1e2842d500b3a5d9f322c6319db0927b2901c3eae23cfb5cd5df6facf2b57399d3cfa52ad7769ebdd75d9b204549ca147da5262
/etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
或如下命令
md5sum /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg
f3f6c5a3a67baf38ad178e5ff1ee270c /etc/apt/trusted.gpg.d/proxmox-ve-release-6.x.gpg

3.2 系统软件升级

我们会定期更新 Proxmox VE 三个软件源的软件包。你可以通过 GUI 管理界面安装升级软件包，也可以直接以命令行方式运行 apt-get 升级：
apt-get update
apt-get dist-upgrade

➢ 注意

apt 软件包管理系统非常灵活，有无数的选项和特性。可以运行 man apt-get 或查看[Hertzog13]获取相关信息。

你应该定期执行以上升级操作，也可以在我们发布安全更新时执行升级。重大系统升级通知会通过 Proxmox VE Community Forum 发布。随升级通知发布的还会有具体的升级细节。

➢ 提示

我们建议定期升级 Proxmox VE，因为及时获取最新的安全补丁是非常重要的。

3.3 网络配置

网络配置可以通过 GUI 进行设置，也可以手工编辑配置文件/etc/network/interfaces 来设置，不管哪种方式，所有的网络配置参数都保存在该文件里。Man 手册文件 interfaces (5)中有完整的设置说明。Proxmox VE 尽量为用户保留编辑配置文件的设置方式，但最好还是使用 GUI 来修改设置，这样可以有效避免配置错误。
一旦网络配置完成，可以通过 Debian 命令 ifup 和 ifdown 来重启端口使配置生效。

➢ 注意

Proxmox VE 并不直接编辑配置文件/etc/network/interfaces 。实际上，网络配置变更会首先保存在临时文件/etc/network/interfaces.new 里，然后在重启服务器时再用该配置文件覆盖/etc/network/interfaces。

3.3.1 命名规范

目前我们采用的网络设备命名规范如下：

- 网卡：en*，即 systemd 类的网络接口命名。新安装的 Proxmox VE 5.0 将采用该命名规范。
- 网卡：eth[N]，其中 0 ≤ N（eth0，eth1， ...）。Proxmox VE 5.0 之前的版本采用该命名规范。从旧版 Proxmox VE 升级至 5.0 以上版本时，网卡命名将保持不变，继续沿用该规范。
- 网桥：vmbr[N]，其中 0 ≤ N ≤ 4094（vmbr0 - vmbr4094）
- 网口组合：bond[N]，其中 0 ≤ N（bond0，bond1， ...）
- VLANs：只需要将 VLAN 编号附加到网络设备名称后面，并用 "." 分隔（eht0.50，bond1.30）

采用命名规范将网络设备名称和网络设备类型关联起来，能够大大降低网络故障排查难度。

Systemd 网卡命名规范

Systemd 采用 en 作为网卡设备名称前缀。名称后续字符由网卡驱动和命名规范匹配先后顺序决定。

- o<index>[n<phys_port_name>|d<dev_port>]—板载设备命名
- s<slot>[f<function>][n<phys_port_name>|d<dev_port>]—按设备热插拔 ID 命名
- [P<domain>]p<bus>s<slot>[f<function>][n<phys_port_name>|d<dev_port>]—按设备总线 ID 命名
- x<MAC>—按设备 MAC 地址命名

最常见的命名模式如下

- eno1—第一个板载 NIC 网卡
- enp3s0f1—位于 3 号 PCI 总线，0 号插槽，NIC 功能号为 1 的 NIC 网卡。

更多信息参见 Predictable Network Interface Names。

3.3.2 网络配置规划

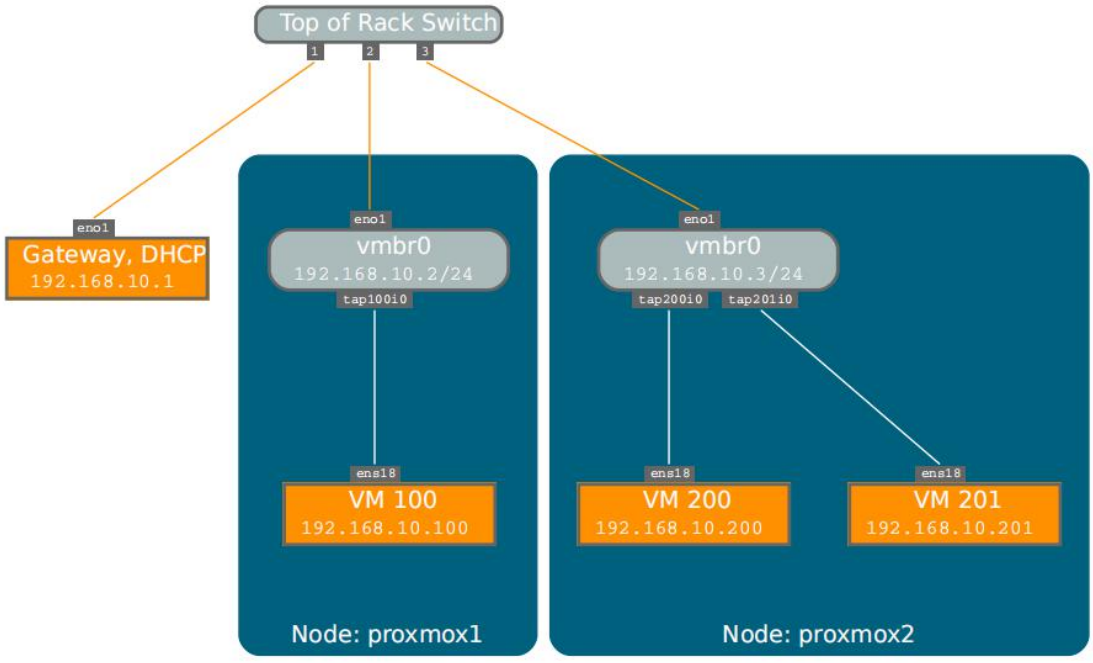
你需要根据当前的网络规划以及可用资源，决定具体采用的网络配置模式。可选模式包括网桥、路由以及地址转换三种类型。

Proxmox VE 服务器位于内部局域网，通过外部网关与互联网连接
这种情况下最适宜采用网桥模式。这也是新安装 Proxmox VE 服务器默认采用的模式。该模式下，所有虚拟机通过虚拟网卡与 Proxmox VE 虚拟网桥连接。其效果类似于虚拟机网卡直接连接在局域网交换机上，而 Proxmox VE 服务器就扮演了这个交换机的角色。

Proxmox VE 托管于主机供应商，并分配有一段互联网公共 IP 地址
这种情况下，可根据主机供应商分配的资源和权限，选择网桥模式或路由模式。

Proxmox VE 托管于主机供应商，但只有一个互联网公共 IP 地址
这种情况下，虚拟机访问外部互联网的唯一办法就是通过地址转换。如果外部网络需要访问虚拟机，还需要配置端口转发。
为日后维护使用方便，可以配置 VLANs（IEEE 802.1q）和网卡绑定，也就是“链路聚合”。这样就可以灵活地建立复杂虚拟机网络结构。

3.3.3 基于网桥的默认配置



网桥相当于一个软件实现的物理交换机。所有虚拟机共享一个网桥，在多个域的网络环境中，也可以创建多个网桥以分别对应不同网络域。理论上，每个 Proxmox VE 最多可以支持 4094 个网桥。
Proxmox VE 安装程序会创建一个名为 vmbr0 的网桥，并和检测到的服务器第一块网卡桥接。配置文件/etc/network/interfaces 中的对应配置信息如下：

```
auto lo
iface lo inet loopback

iface eno1 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 eno1
        bridge_stp off
        bridge_fd 0
```

在基于网桥的默认配置下，虚拟机看起来就和直接接入物理网络一样。尽管所有虚拟机共享一根网线接入网络，但每台虚拟机都使用自己独立的 MAC 地址访问网络。

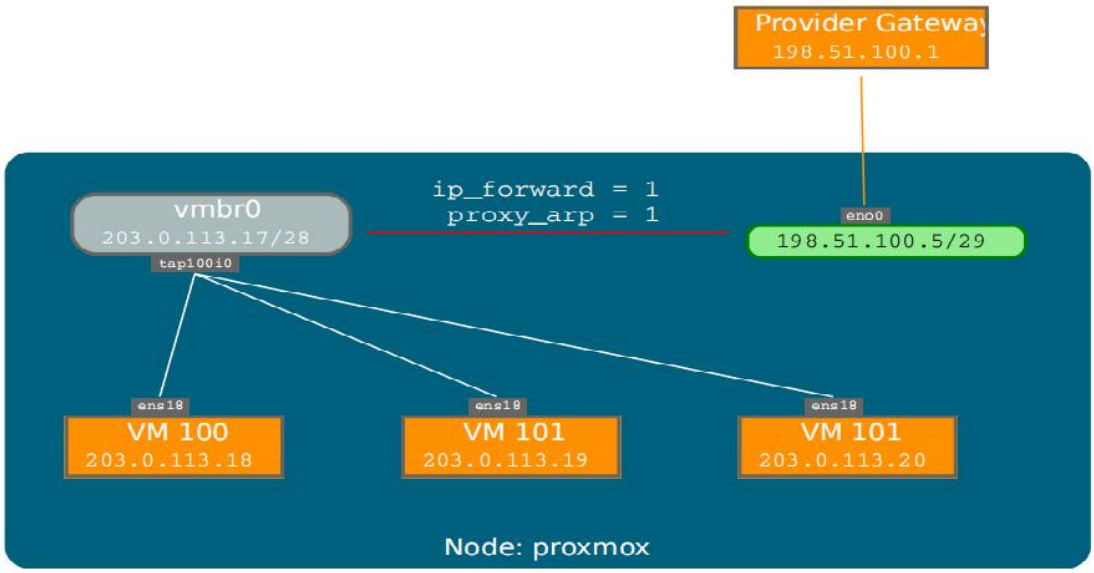
3.3.4 路由配置

但大部分 IPC 服务器供应商并不支持基于网桥的默认配置方式，出于网络安全的考虑，一旦发现网络接口上有多个 MAC 地址出现，他们会立刻禁用相关网络端口。

➢ 提示

也有一些 IPC 服务器供应商允许你注册多个 MAC 地址。这就可以避免上面提到的问题，但这样你就需要注册每一个虚拟机 MAC 地址，实际操作会非常麻烦。

你可以用配置“路由”的方式让多个虚拟机共享一个网络端口，这样就可以避免上面提到的问题。这种方式可以确保所有的对外网络通信都使用同一个 MAC 地址。



常见的应用场景是，你有一个可以和外部网络通信的 IP 地址（假定为 192.51.100.5），还有一个供虚拟机使用的 IP 地址段（203.0.113.16/29）。针对该场景，我们推荐使用如下配置：

```
auto lo
iface lo inet loopback

auto eno1
iface eno1 inet static
        address 192.51.100.5
        netmask 255.255.255.0
        gateway 192.51.100.1
        post-up echo 1 > /proc/sys/net/ipv4/ip_forward
        post-up echo 1 > /proc/sys/net/ipv4/conf/eno1/proxy_arp

auto vmbr0
iface vmbr0 inet static
        address 203.0.113.17
        netmask 255.255.255.248
        bridge_ports none
        bridge_stp off
        bridge_fd 0
```

3.3.5 基于 iptables 的网络地址转换配置（NAT）

利用地址转换技术，所有虚拟机可以使用内部私有 IP 地址，并通过 Proxmox VE 服务器的 IP 来访问外部网络。Iptables 将改写虚拟机和外部网络通信数据包，对于虚拟机向外部网络发出的数据包，将源 IP 地址替换成服务器 IP 地址，对于外部网络返回数据包，将目的地址替换为对应虚拟机 IP 地址。

```
auto lo
iface lo inet loopback

auto eno1
#real IP address
iface eno1 inet static
        address 192.51.100.5
        netmask 255.255.255.0
        gateway 192.51.100.1

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 eno1
        -j MASQUERADE
post-down iptables -t nat -D POSTROUTING -s '10.10.10.0/24' -o eno1
        -j MASQUERADE
```

3.3.6 Linux 多网口绑定

多网口绑定（也称为网卡组或链路聚合）是一种将多个网卡绑定成单个网络设备的技术。利用该技术可以实现某个或多个目标，例如提高网络链路容错能力，增加网络通信性能等。类似光纤通道和光纤交换机这样的高速网络硬件的价格一般都非常昂贵。利用链路聚合技术，将两个物理网卡组成一个逻辑网卡，能够将网络传输速度加倍。大部分交换机设备都已经支持 Linux 内核的这个特性。如果你的服务器有多个以太网口，你可以将这些网口连接到不同的交换机，以便将故障点分散到不同的网络设备，一旦有物理线路故障或网络设备故障发生，多网卡绑定会自动将通信流量从故障线路切换到正常线路。
链路聚合技术可以有效减少虚拟机在线迁移的时延，并提高 Proxmox VE 集群服务器节点之间的数据复制速度。
目前一共有 7 种网口绑定模式：

- 轮询模式（blance-rr）：网络数据包将按顺序从绑定的第一个网卡到最后一个网卡轮流发送。这种模式可以同时实现负载均衡和链路容错效果。
- 主备模式（active-backup）：该模式下网卡组中只有一个网卡活动。只有当活动的网卡故障时，其他网卡才会启动并接替该网卡的工作。整个网卡组使用其中一块网卡的 MAC 地址作为对外通信的 MAC 地址，以避免网络交换机产生混乱。这种模式仅能实现链路容错效果。
- 异或模式（balance-xor）：网络数据包按照异或策略在网卡组中选择一个网卡发送（[源 MAC 地址 XOR 目标 MAC 地址] MOD 网卡组中网卡数量）。对于同一个目标 MAC 地址，该模式每次都选择使用相同网卡通信。该模式能同时实现负载均衡和链路容错效果。
- 广播模式（broadcast）：网络数据包会同时通过网卡组中所有网卡发送。该模式能实现链路容错效果。
- IEEE 802.3ad 动态链路聚合模式（802.3ad）（LACP）：该模式会创建多个速度和双工配置一致的聚合组。并根据 802.3ad 标准在活动聚合组中使用所有网卡进行通信。

- 自适应传输负载均衡模式（balance-tlb）：该 Linux 网卡绑定模式无须交换机支持即可配置使用。根据当前每块网卡的负载情况（根据链路速度计算的相对值），流出的网络数据包流量会自动进行均衡。流入的网络流量将由当前指定的一块网卡接收。如果接收流入流量的网卡故障，会自动重新指定一块网卡接收网络数据包，但该网卡仍沿用之前故障网卡的 MAC 地址。

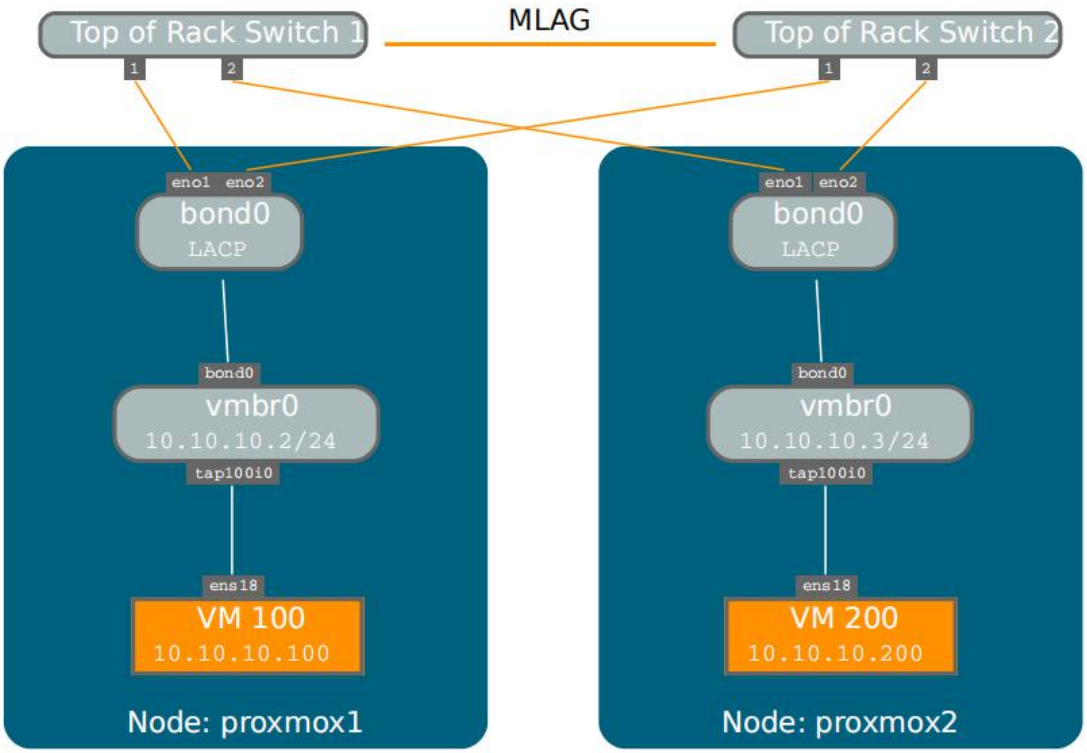
- 自适应负载均衡模式（均衡的 IEEE 802.3ad 动态链路聚合模式（802.3ad）（LACP）:-alb）：该模式是在 blance-tlb 模式的基础上结合了 IPV4 网络流量接收负载均衡（rlb）特性，并且无须网络交换机的专门支持即可配置使用。网络流量接收负载均衡基于 ARP 协商实现。网卡组驱动将自动截获本机的 ARP 应答报文，并使用网卡组中其中一块网卡的 MAC 地址覆盖 ARP 报文中应答的源 MAC 地址，从而达到不同的网络通信对端和本机不同 MAC 地址通信的效果。

在网络交换机支持 LACP (IEEE 802.3ad) 协议的情况下，推荐使用 LACP 绑定模式 (802.3ad)，其他情况建议使用 active-backup 模式。
对于 Proxmox 集群网络的网卡绑定，目前仅支持 active-backup 模式，其他模式均不支持。下面所列的网卡绑定配置示例可用于分布式/共享存储网络配置。其主要优势是能达到更高的传输速度，同时实现网络链路容错的效果。

示例：基于固定 IP 地址的多网卡绑定

```
auto lo
iface lo inet loopback

iface eno1 inet manual

iface eno2 inet manual

auto bond0
iface bond0 inet static
        slaves eno1 eno2
        address 192.168.1.2
        netmask 255.255.255.0
        bond_miimon 100
        bond_mode 802.3ad
        bond_xmit_hash_policy layer2+3

auto vmbr0
iface vmbr0 inet static
        address 10.10.10.2
        netmask 255.255.255.0
        gateway 10.10.10.1
        bridge_ports eno1
```

bridge_stp off
bridge_fd 0



另一种配置方法是直接使用网卡组作为虚拟交换机桥接端口，从而实现虚拟机网络的容错效果。

示例：利用多网卡绑定配置网桥端口

```
auto lo
iface lo inet loopback

iface eno1 inet manual

iface eno2 inet manual

auto bond0
iface bond0 inet maunal
        slaves eno1 eno2
        bond_miimon 100
        bond_mode 802.3ad
        bond_xmit_hash_policy layer2+3

auto vmbr0
iface vmbr0 inet static
        address 10.10.10.2
        netmask 255.255.255.0
        gateway 10.10.10.1
```

bridge_ports bond0
bridge_stp off
bridge_fd 0

3.3.7 VLAN 802.1Q

虚网（VLAN）是基于 2 层网络的广播域划分和隔离技术。利用 VLAN，可以在一个物理网络中创建多个（最多 4096）相互隔离的子网。
每个 VLAN 都有一个独立的，称为 tag 的编号。相应的，每个网络数据包都被标上 tag 以标明其所属的 VLAN。

基于 VLAN 的虚拟机网络

Proxmox VE 直接支持 VLAN 模式。创建虚拟机时，可以指定 tag，使该虚拟机接入对应 VLAN。VLAN tag 是虚拟机网络配置的一部分。根据虚拟网桥配置的不同，网络层支持多种不同的 VLAN 实现模式：

- Linux 网桥感知 VLAN 配置模式：这种方式下，每个虚拟机的虚拟网卡都分配了一个 VLAN tag，Linux 网桥将自动支持这些虚拟网卡的 VLAN tag。也可以将虚拟网卡配置为 trunk 模式，但 VLAN tag 的配置就需要在虚拟机内部另行配置完成。
- Linux 网桥传统 VLAN 配置模式：不同于 VLAN 感知模式，传统模式下的 Linux 网桥不能直接支持配有 VLAN tag 的虚拟网卡，而需要为每个 VLAN 另行创建一个虚拟网桥，以便连接该属于 VLAN 的虚拟网卡设备。例如要在默认网桥上配置一个 VLAN 5 供虚拟机使用，就需要创建一个虚拟网卡 eno1.5 和虚拟网桥 vmbr0v5，然后重启服务器以使其生效。
- Open vSwitch VLAN 配置模式：基于 OVS VLAN 特性实现。
- 虚拟机配置 VLAN 模式：这种配置模式下，VLAN 是在虚拟机内部配置实现的。这时，有关配置完全在虚拟机内部实现，不受外部配置干扰。这种配置模式最大好处是可以在一个虚拟网卡上同时运行多个 VLAN。

Proxmox VE 主机上的 VLAN

Proxmox VE 主机上配置 VLAN，可以将主机网络通信与其他网络的逻辑隔离。实际上，你可以在任意网络设备上配置使用 VLAN tag（如网卡、多网卡绑定，网桥）。一般情况下，你应该在与物理网卡最近，中间虚拟设备数最少的接口设备上配置 VLAN tag。
思考一下，在默认网络配置下，你会在的哪个设备上配置 Proxmxo VE 的管理地址 VLAN？

示例：在传统 Linux Bridge 上利用 VLAN 5 配置 Proxmox VE 管理 IP

auto lo

```
iface lo inet loopback

iface eno1 inet manual

iface eno1.5 inet manual

auto vmbr0v5
iface vmbr0v5 inet static
        address 10.10.10.2
        netmask 255.255.255.0
        gateway 10.10.10.1
        bridge_ports eno1.5
        bridge_stp off
        bridge_fd 0

auto vmbr0
iface vmbr0 inet manual
        bridge_ports eno1
        bridge_stp off
        bridge_fd 0
```

示例：在启用 VLAN 感知的 Linux Bridge 上利用 VLAN 5 配置 Proxmox VE 管理 IP

```
auto lo
iface lo inet loopback

iface eno1 inet manual

auto vmbr0.5
iface vmbr0.5 inet static
        address 10.10.10.2
        netmask 255.255.255.0
        gateway 10.10.10.1

auto vmbr0
iface vmbr0 inet manual
        bridge_ports eno1
        bridge_stp off
        bridge_fd 0
        bridge_vlan_aware yes
```

下一个示例配置实现的功能完全一样，但是增加了多网卡绑定，以避免单链路故障。

示例：在传统 Linux Bridge 上利用 bond0 和 VLAN 5 配置 Proxmox VE 管理 IP

```
auto lo
```

```
iface lo inet loopback

iface eno1 inet manual

iface eno2 inet manual

auto bond0
iface bond0 inet manual
        slaves eno1 eno2
        bond_miimon 100
        bond_mode 802.3ad
        bond_xmit_hash_policy layer2+3

iface bond0.5 inet manual

auto vmbr0v5
iface vmbr0v5 inet static
        address 10.10.10.2
        netmask 255.255.255.0
        gateway 10.10.10.1
        bridge_ports bond0.5
        bridge_stp off
        bridge_fd 0

auto vmbr0
iface vmbr0 inet manual
        bridge_ports bond0
        bridge_stp off
        bridge_fd 0
```

3.4 时钟同步

Proxmox VE 集群的正常运行基于一个重要条件，那就是所有节点的服务器时钟需要精确地同步。其他一些组件，例如 Ceph，如果各节点的时钟不同步，也不能正常工作。
各节点之间的时钟同步可利用“网络时间协议”（NTP）实现。Proxmox VE 默认使用 systemd-timesyncd 作为 NTP 客户端，并默认配置使用一组互联网服务器作为时间源。大部分情况下，系统安装后即可自动实现时钟同步。

3.4.1 使用自定义 NPT 服务器

大多数情况下，默认的 NTP 服务器可能并不适用。例如你的 Proxmox VE 服务器根本连不上 internet 互联网（比如，防火墙策略禁止了互联网连接），这时你需要自行搭建本地 NTP 服务器，并配置 systemd-timesyncd 和本地 NTP 服务器进行时钟同步。

/etc/systemd/timesyncd.conf 默认配置

[Time]
Servers=ntp1.example.com ntp2.example.com ntp3.example.com ntp4.example.com

重启时钟同步服务后（systemctl restart systemd-timesyncd），你可以查看日志以验证新配置的 NTP 服务器是否已被启用（journalctl –since -1h -u systemd-timesyncd）。
日志输出示例如下：
...
Oct 07 14:58:36 node1 systemd[1]: Stopping Network Time Synchronization...
Oct 07 14:58:36 node1 systemd[1]: Starting Network Time Synchronization...
Oct 07 14:58:36 node1 systemd[1]: Started Network Time Synchronization.
Oct 07 14:58:36 node1 systemd-timesyncd[13514]: Using NTP server 10.0.0.1:123 (ntp1.example.com).
Oct 07 14:58:36 nora systemd-timesyncd[13514]: interval/delta/delay/jitter/drift 64s/-0.002s/0.020s/0.000s/-31ppm
...

3.5 外部监控服务器

从 Proxmox VE 4.0 开始，你可以搭建外部监控服务器，集中收集 Proxmox VE 服务器主机、虚拟机和存储的监控数据。
目前支持的外部监控服务器有：

- Graphite（参见 http://graphiteapp.org ）
- InfluxDB（参见 https://www.influxdata.com/time-series-platform/influxdb ）

外部监控服务器配置信息保存在/etc/pve/status.cfg 文件中。

3.5.1 配置 Graphite 服务器

Graphite 服务器可按如下格式配置
graphite: your-id
        server your-server
        port your-port
        path your-path

其中 your-port 默认设置为 2003，your-path 默认设置为 proxmox。
Proxmox VE 服务器使用 udp 协议发送监控数据，所以 graphite 服务器需要进行相应配置。

3.5.2 配置 Influxdb 插件

可按如下格式配置

```
influxdb: your-id
        server your-server
        port your-port
```

Proxmox VE 服务器使用 udp 协议发送监控数据，所以 influxdb 服务器需要进行相应配置。
下面是一个 influxdb 配置示例（配置在 influxdb 服务器上）：

```
[[udp]]
        enabled = true
        bind-address = "0.0.0.0:8089"
        database = "proxmox"
        batch-size = 1000
        batch-timeout = "1s"
```

按如上配置，influxdb 服务器将监听 8089 端口，并将接收到的数据写入 proxmox 数据库。

3.5.3 多实例配置示例

其中 id 为可选参数。但在配置多个同类型实例时，就必须为每个实例设置不同的 id。
以下是一个完整的 status.cfg 配置示例

```
graphite:
    server 10.0.0.5
influxdb: influx1
    server 10.0.0.6
    port 8089
influxdb: influx2
    server 10.0.0.7
    port 8090
```

3.6 硬盘健康状态监控

为 Proxmox VE 配置存储设备时，不仅要考虑健壮性和冗余性，还应该考虑监控硬盘的健康状态。
从 4.3 版开始，Proxmox VE 集成了 smartmontools 软件包。这是一套可用来监控管理服务器本地硬盘 S.M.A.R.T 系统的工具。
你可以运行如下命令获取磁盘的状态信息：

smartctl -a /dev/sdX

其中/dev/sdX 是某个本地磁盘的设备文件名称。
如果命令输出如下结果：
SMART support is: Disabled
你就需要运行以下命令启用 SMART 监控：
smartctl -s on /dev/sdX

你可以运行 man smartctl 命令查看技术手册，以获取关于 smartctl 使用方法的更多信息。
默认状态下，smartctltools 守护进程 smartd 将会被启动并自动运行，每 30 分钟对/dev/sdX 和/dev/hdX 进行一次扫描，如果检测到的错误和警告信息，就向服务器的 root 用户发送 e-mail。
可以运行 man smartd 和 man smartd.conf 命令，以获取关于 smartd 配置方法的更多信息。
如果你的服务器使用了硬件 raid 卡管理硬盘，一般都会有专门工具来监控 raid 阵列中的硬盘以及 raid 阵列本身。具体可以咨询 raid 卡控制器的厂商。

3.7 逻辑卷管理器（LVM）

大部分用户都直接将 Proxmox VE 安装到服务器本地硬盘。Proxmox VE 安装光盘为服务器本地硬盘配置提供了多种选项，目前默认的选项为 LVM。默认情况下，安装程序会引导你选择一块硬盘用于安装 Proxmox VE 系统，并将该硬盘格式化为物理卷（PV）的形式以创建卷组（VG）pve。下面是在一个 8GB 容量的小硬盘上测试安装 Proxmox VE 后的 LVM 配置输出：

```
# 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
```

安装程序会在 VG 上创建三个逻辑卷（LV）

```
# 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

格式化为 ext4 文件系统，用于安装 Proxmox VE 操作系统

- swap

用于 Swap 分区

- data

使用 LVM-thin 格式化，用于保存虚拟机镜像。LVM-thin 在对于虚拟机快照和克隆的效率较高，所以非常适合用于保存虚拟机镜像。
到 Proxmox VE 4.1 为止，安装程序会创建一个标准的名为“data”的逻辑卷，并挂载在/var/lib/vz 路径下。
从 Proxmox VE 4.2 开始，安装程序会以 LVM-thin 形式创建“data”逻辑卷，并用于存储基于块存储的虚拟机镜像，而/var/lib/vz 是根文件系统下的一个路径。

3.7.1 硬件

我们强烈推荐使用基于硬件 RAID 控制器（带有电池保护写缓存，BBU）的服务器安装 Proxmox VE。这样不仅能提高性能，提供冗余性，还能简化故障硬盘的更换操作（，可热插拔）。
LVM 本身无须任何特殊的硬件支持，并且对于内存的需求也很低。

3.7.2 启动引导程序

默认情况下，安装程序会安装两个启动引导程序。第一个分区会安装标准的 GRUB 启动引导程序。第二个分区会格式化为 EFI 系统分区（EFI System Partition，ESP），这样就可以启动基于 EFI 的系统。

3.7.3 创建卷组

假定我们现在有一块空白磁盘设备，其设备文件为/dev/sdb，可以用如下方法利用该磁盘创建卷组“vmdata”。

☒ 警告

执行以下这些命令将会彻底销毁/dev/sdb 上的原有数据。

首先创建一个分区。
sgdisk -N 1 /dev/sdb
然后使用静默方式创建一个物理卷（PV），并设置元数据大小为 250K。
pvcreate --metadatasize 250k -y -ff /dev/sdb1
然后在/dev/sdb1 上创建卷组“vmdata”。
vgcreate vmdata /dev/sdb1

3.7.4 为/var/lib/vz 添加 LV

可用如下命令添加“薄模式”的逻辑卷（thin LV）。
lvcreate -n <Name> -V <Size[M,G,T]> <VG>/<LVThin_pool>

示例如下：
lvcreate -n vz -V 10G pve/data

然后在该逻辑卷上创建文件系统，如下。
mkfs.ext4 /dev/data/vz
最后，将该逻辑卷挂载到文件系统。

☒ 警告

挂载前需确保/var/lib/vz 下为空。默认/var/lib/vz 不为空。

为确保系统重启后自动挂载，可运行如下命令，修改/etc/fstab 文件的配置内容。
echo '/dev/pve/vz /var/lib/vz ext4 defaults 0 2' >> /etc/fstab

3.7.5 调整 thin pool 的容量

可以用如下命令调整 LV 容量大小和元数据池大小。
lvresize --size +<size[\M,G,T]> --poolmetadatasize +<size[\M,G]> <VG>/<LVThin_pool>

➢ 注意

扩展数据池容量的同时，必须相应扩展元数据池的容量。

3.7.6 创建 LVM-thin 存储池

首先要创建一个卷组，然后才能创建“薄模式”存储池。可以参考 LVM 一节查看创建卷组的步骤。创建“薄模式”存储池命令如下。
lvcreate -L 80G -T -n vmstore vmdata

3.8 Linux 上的 ZFS

ZFS 是由太阳微系统公司（Sun Microsystems）设计的同时集成有文件管理功能和逻辑卷管

理功能的文件系统。从 Proxmox VE 3.4 开始，Proxmox VE 引入了 ZFS 文件系统作为可选的文件系统和根文件系统。Proxmox VE 官方 ISO 光盘镜像已经集成了 ZFS 所需的软件包，用户无须手工编译即可直接使用 ZFS。
利用 ZFS，用户可以在配置有限的硬件设备上尽可能多的享受企业级的文件系统服务，进一步还可以利用 SSD 缓存或者全 SSD 技术获得极高的系统性能。利用常见的 CPU 和内存硬件，只需简单的配置，ZFS 即可实现昂贵的基于硬件 RAID 卡的管理功能。
ZFS 的技术优势如下：

- 可通过 Proxmox VE 的图形界面或命令行方式进行配置管理，容易使用。
- 高可靠性
- 有效防止数据损坏丢失
- 文件系统级的数据压缩
- 支持快照功能
- 支持 Copy-on-write
- 多种软 RAID 模式：RAID0，RAID1，RAID10，RAIDZ-1，RAIDZ-2，RAIDZ-3
- 支持 SSD 缓存
- 自我数据修复
- 连续数据完整性检查和验证
- 支持大容量数据存储
- 有效防止数据损坏丢失
- 基于网络的数据异步复制
- 开源软件
- 支持数据加密

3.8.1 硬件

ZFS 对于内存配置的依赖较高，一般最少需要为 ZFS 配置 8GB 内存。实际生产中，最好基于你的预算配置尽可能多的内存。为防止数据损坏，我们建议你使用高端的 ECC 内存条。
如果你要为 ZFS 配置独立的缓存盘或文件系统日志盘，最好使用企业级的 SSD 盘（例如 Intel SSD DC S3700 系列）。这能够极大的提升整体性能表现。

☒ 重要

不要使用有独立缓存管理的硬件控制器。ZFS 需要能够直接访问磁盘。ZFS 可以和 HBA 卡配合使用，也可以和“IT”模式的 LSI 控制器及类似硬件配合使用。

如果你是在 Proxmox VE 虚拟机环境里测试安装 Proxmox VE（嵌套虚拟化），不要使用 virtio 类型的虚拟磁盘，目前 ZFS 不支持 virtio 类型磁盘。建议为虚拟机配置 IDE 或 SCSI 类型的虚拟磁盘（virtio SCSI 控制器）。

3.8.2 用于根文件系统

当你使用 Proxmox VE 安装程序安装时，可以选择使用 ZFS 作为根文件系统。同时你需要在安装过程中选择配置 RAID 级别。

- RAID0

也称为“条带模式”。该模式下 ZFS 卷的容量为所有硬盘容量之和。但 RAID0 不提供任何冗余性，卷中任何一块硬盘故障都会导致整个卷不可用。

- RAID1

也称为“镜像模式”。该模式下，数据会以复制方式同时写入所有硬盘。该模式至少需要 2 块容量一样的硬盘，而整个卷的容量就等于单块盘的容量。

- RAID10

该模式组合了 RAID0 和 RAID1 模式。配置使用该模式至少需要 4 块硬盘。

- RAIDZ-1

类似于 RAID5 模式，提供 1 块硬盘故障冗余。配置使用该模式至少需要 3 块硬盘。

- RAIDZ-2

类似于 RAID5 模式，提供 2 块硬盘故障冗余。配置使用该模式至少需要 4 块硬盘。

- RAIDZ-3

类似于 RAID5 模式，提供 3 块硬盘故障冗余。配置使用该模式至少需要 5 块硬盘。

安装程序会自动完成硬盘分区，构建名为“rpool”的 ZFS 存储池，并在 rpool/ROOT/pve-1 上安装根文件系统。
安装程序还会创建一个名为 rpool/data 的子卷，用于保存虚拟机镜像。为便于使用 Proxmox VE 工具管理该 ZFS 卷，安装程序会在/etc/pve/storage.cfg 文件中创建以下配置信息：

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

安装完成后，你可以运行 zpool 命令查看 ZFS 存储池的状态：

```
# 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
```

可以用 ZFS 命令配置管理 ZFS 文件系统。下面的命令用于列出安装 Proxmox VE 后的 ZFS 文件系统。

```
# 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          -
```

3.8.3 系统引导程序

取决于服务器使用 EFI 还是传统 BIOS，Proxmox VE 安装程序将设置使用 grub 或 systemd-boot 作为主引导程序。详见 3.10 节 Proxmox VE 主机引导程序。

3.8.4 ZFS 管理

本节将给出 ZFS 日常管理操作的范例。ZFS 本身非常强大，具有很多命令选项。管理 ZFS 最主要的两个命令是 zfs 和 zpool。这两个命令都有非常完善的技术手册，可以使用如下命令查看：

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

- 创建新的存储池

至少需要有 1 块硬盘才可以创建一个新的存储池。可以用参数 ashift 指定区块大小（2 的 asshift 次幂），且不能小于所用磁盘区块的大小。
zpool create -f -o ashift=12 <pool> <device>
还可以执行以下命令激活数据压缩功能。
zfs set compression=lz4 <pool>

- 创建新的 RAID-0 存储池

至少需要 1 块硬盘。
zpool create -f -o ashift=12 <pool> <device1> <device2>

- 创建新的 RAID-1 存储池

至少需要 2 块硬盘。
zpool create -f -o ashift=12 <pool> mirror <device1> <device2>

- 创建新的 RAID-10 存储池

至少需要 4 块硬盘。
zpool create -f -o ashift=12 <pool> mirror <device1> <device2> mirror <device3> <device4>

- 创建新的 RAIDZ-1 存储池

至少需要 3 块硬盘。
zpool create -f -o ashift=12 <pool> raidz1 <device1> <device2> <device3>

- 创建新的 RAIDZ-2 存储池

至少需要 4 块硬盘。
zpool create -f -o ashift=12 <pool> raidz2 <device1> <device2> <device3> <device4>

- 创建新的带有缓存（L2ARC）的存储池

可以使用独立的硬盘分区(建议使用 SSD)作为缓存, 以提高 ZFS 性能。如下命令中, <device> 处可以列出多个硬盘设备，命令格式就像“创建带 RAID 的存储池”一样。
zpool create -f -o ashift=12 <pool> <device> cache <cache_device>

- 创建新的带日志（ZIL）的存储池

可以使用独立的硬盘分区（建议使用 SSD）记录文件系统日志，以提高 ZFS 性能。如下命令中，<device>处可以列出多个硬盘设备，命令格式就像“创建带 RAID 的存储池”一样。
zpool create -f -o ashift=12 <pool> <device> log <log_device>

- 为已有的存储池添加缓存和日志盘

如果你要为一个未配置缓存和日志盘的 ZFS 存储池添加缓存和日志盘, 首先需要使用 parted 或者 gdisk 将 SSD 盘划分为两个分区。

☒ 重要

确保使用 GPT 分区表。

日志盘的大小最大为物理内存容量的一半，通常都不大。SSD 盘剩余空间可用作缓存。
zpool add -f <pool> log <device-part1> cache <device-part2>

- 更换故障磁盘

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

- 在使用 systemd-boot 时更换故障的系统磁盘

sgdisk <healthy bootable device> -R <new device>
sgdisk -G <new device>
zpool replace -f <pool> <old zfs partition> <new zfs partition>
pve-efiboot-tool format <new disk's ESP>
pve-efiboot-tool init <new disk's ESP>

➢ 注意

ESP 是 EFI 系统分区，Proxmo VE 5.4 以后安装程序会将其设置为启动盘的#2 分区。详见将一个新分区设置为同步 ESP。

3.8.5 使用邮件通知

ZFS 有一个事件守护进程，专门监控 ZFS 内核模块产生的各类事件。当发生严重错误，例如存储池错误时，该进程还可以发送邮件通知。
可以编辑配置文件/etc/zfs/zed.d/zed.rc 以激活邮件通知功能。只需将配置参数 ZED_EMAIL_ADDR 前的注释符号去除即可，如下：
ZED_EMAIL_ADDR="root"
请注意，Proxmox VE 会将邮件发送给为 root 用户配置的电子邮件地址。

☒ 重要

只需激活 ZED_MAIL_ADDR 参数即可。其他配置参数均为可选项目，非必须项。

3.8.6 配置 ZFS 内存使用上限

最好将 ZFS 的缓存容量 ZFS ARC 上限设置为物理内存容量的一半，以避免内存短缺导致系统性能变坏。具体做法是编辑配置文件/etc/modprobe.d/zfs.conf，插入如下内容：
options zfs zfs_arc_max=8589934592

上例中设置 ZFS 缓存容量上限为 8GB。

☒ 重要

如果根文件系统也使用了 ZFS，你必须在每次修改该参数后更新 initramfs，如下：
update-initramfs –u

3.8.7ZFS 上的 SWAP

使用 zvol 创建 SWAP 分区可能会导致一些问题，例如系统卡死或者很高的 IO 负载。特别是在向外部存储备份文件时会容易触发此类问题。
我们强烈建议为 ZFS 配置足够的物理内存，避免系统出现可用内存不足的情形。如果实在想要创建一个 SWAP 分区，最好是直接在物理磁盘上创建。可以在安装 Proxmox VE 时通过高级选项设置预留磁盘空间，以便创建 SWAP。此外，你可以调低“swappiness”参数值。通常，设置为 10 比较好。
sysctl -w vm.swappiness=10

如果需要将 swappiness 参数设置持久化，可以编辑文件/etc/sysctl.conf，插入下内容：
vm.swappiness=10

表 3.1 Linux 内核 swappiness 参数设置表

值	对应策略
vm.swappiness=0	内核仅在内存耗尽时进行 swap。
vm.swappiness=1	内核仅执行最低限度的 swap。
vm.swappiness=10	当系统有足够多内存时，可考虑使用该值，以提高系统性能。
vm.swappiness=60	默认设置值。
vm.swappiness=100	内核将尽可能使用 swap。

3.8.8 加密 ZFS 数据集

ZFS on Linux 在 0.8.0 版之后引入了本地数据集加密功能。将 ZFS on Linux 升级后，就可以对指定存储池启用加密功能：

```
# zpool get feature@encryption tank
NAME    PROPERTY              VALUE        SOURCE
tank    feature@encryption    disabled     local
```

```
# zpool set feature@encryption=enabled
# zpool get feature@encryption tank
NAME    PROPERTY              VALUE         SOURCE
tank    feature@encryption    enabled       local
```

☒ 警告

目前还不支持通过 Grub 从加密数据集启动系统，并且在启动过程中对自动解锁加密数据集的支持也很弱。不支持加密功能的旧版 ZFS 也不能解密相关数据。

➢ 注意

建议在启动后手工解锁存储数据集，或使用 zfs load-key 命令将启动中解锁数据集所需 key 信息写到定制参数中。

☒ 警告

在对生产数据正式启用加密功能前，建议建立并测试备份程序有效性。注意，一旦 key 信息丢失，将永远不可再访问加密数据。

加密功能需要在创建数据集/zvol 时启用，并将被子数据集自动继承启用。例如，可用以下命令创建加密数据集 tank/encrypted_data，并提供给 Proxmox VE 使用。

```
# zfs create -o encryption=on -o keyformat=passphrase tank/encrypted_data
Enter passphrase:
Re-enter passphrase:
# pvesm add zfspool encrypted_zfs -pool tank/encrypted_data
```

在该存储上创建的所有客户机卷/磁盘都将被自动加密，加密秘钥将共享使用父存储的秘钥。如需使用存储，需要使用 zfs load-key 命令加载秘钥信息：

```
# zfs load-key tank/encrypted_data
Enter passphrase for 'tank/encrypted_data':
```

也可以使用随机秘钥文件代替手工输入的秘钥。只需在创建数据集时设置 keylocation 和 keyformat 参数即可，或者使用 zfs change-key 命令改变已有数据集的设置。

```
# dd if=/dev/urandom of=/path/to/keyfile bs=32 count=1
# zfs change-key -o keyformat=raw -o keylocation=file:///path/to/keyfile ←-
tank/encrypted_data
```

☒ 警告

使用加密文件时，需要特别注意保护好加密文件，确保不被越权访问，或者意外丢失。一旦丢失了秘钥文件，就不可能再访问相关数据。

如果客户机磁盘创建在加密数据集上，虚拟磁盘将自动继承启用 encryptionroot 参数。对每个数据集的 encryptionroot，都需要加载秘钥信息。

具体信息可以查看 man zfs 手册的 Encryption 小节，包括 encryptionroot，encryption，keylocation，keyformat 和 keystatus 属性，zfs load-key，zfs unload-key 和 zfs change-key 命令。

3.9 证书管理

3.9.1 集群通信认证

每个 Proxmox VE 集群都会创建内部（自签名）认证证书（CA），并以此为每个节点生成一个签名证书。这些证书用于集群内 pveproxy 服务通信加密，以及 SPICE 控制台的 Shell/Console 通信加密。
CA 证书和密钥保存在 pmxcfs（详情见第 7 章）。

3.9.2 认证 API 和 Web GUI 界面

Proxmox 每个服务器的 pveproxy 服务都提供了 REST API 接口和 Web GUI 界面。
在 pveproxy 中可用的 CA 认证方式如下：

1.默认情况下，使用指定节点认证文件/etc/pve/nodes/NODENAME/pve-ssl.pem。该证书由集群 CA 签发，因此浏览器和操作系统默认不信任该证书。
2.使用外部证书（例如，由商业 CA 签发的证书）。
3.使用 ACME（例如，Let's Encrypt）获得可信证书。

第 2、3 种认证方式使用/etc/pve/local/pveproxy-ssl.pem（和/etc/pve/local/pveproxy-ssl.key，默认没有保护口令）。
可以通过 Proxmox VE 节点管理命令管理证书（详情见 man 手册 pvenode（1））。

☒ 警告

不要替换或手工修改自动生成的节点证书文件/etc/pve/local/pve-ssl.pem 和/etc/pve/local/pve-ssl.key，以及集群 CA 文件/etc/pve/pve-root-ca.pem 和/etc/pve/priv/pve-root-ca.key。

通过 ACME 获取可信证书

Proxmox VE 内置了自动化证书管理 **A**utomatic **C**ertificate **M**anagement **E**nviroment，即 **ACME** 协议，可以通过 Let's Encrypt 轻松获取可信 TLS 证书，从而被大多数操作系统和浏览器直接接受。
目前实现的 ACME 端点是 Let's Encrypt（LE）及其 staging 环境（详见 https://letsencrypt.org），均使用标准的 HTTP 挑战问答模式。
限于 rate-limit，LE staging 可用于试验。
使用 Let's Encrypt 前需准备好以下条件：

1.Proxmox 服务器对互联网开放 80 端口访问权限。
2.Proxmox 服务器没有其他服务使用 80 端口。
3.请求的（子）域名能够被解析为节点的互联网公共 IP 地址。
4.需要接受 Let's Encrypt 的 ToS。

当前 GUI 仅使用默认 ACME 账号。

示例：使用 pvenode 命令获取 Let's Encrypt 证书

```
root@proxmox:~# pvenode acme account register default mail@example.invalid
Directory endpoints:
0) Let's Encrypt V2 (https://acme-v02.api.letsencrypt.org/directory)
1) Let's Encrypt V2 Staging (https://acme-staging-v02.api.letsencrypt.org/ ↵
directory)
2) Custom
Enter selection:
1

Attempting to fetch Terms of Service from 'https://acme-staging-v02.api. ↵
    letsencrypt.org/directory'..
Terms of Service: https://letsencrypt.org/documents/LE-SA-v1.2-November ↵
    -15-2017.pdf
Do you agree to the above terms? [y|N]y

Attempting to register account with 'https://acme-staging-v02.api. ↵
    letsencrypt.org/directory'..
Generating ACME account key..
Registering ACME account..
Registration successful, account URL: 'https://acme-staging-v02.api. ↵
    letsencrypt.org/acme/acct/xxxxxxx'
Task OK
root@proxmox:~# pvenode acme account list
default
root@proxmox:~# pvenode config set --acme domains=example.invalid
root@proxmox:~# pvenode acme cert order
Loading ACME account details
Placing ACME order
Order URL: https://acme-staging-v02.api.letsencrypt.org/acme/order/ ↵
    xxxxxxxxxxxxxx

Getting authorization details from
'https://acme-staging-v02.api.letsencrypt.org/acme/authz/ ↵
    xxxxxxxxxxxxxxxxxxxxx-xxxxxxxxxxxxx-xxxxxxx'
... pending!
```

```
Setting up webserver
Triggering validation
Sleeping for 5 seconds
Status is 'valid'!

All domains validated!

Creating CSR
Finalizing order

Checking order status
valid!

Downloading certificate
Setting pveproxy certificate and key
Restarting pveproxy
Task OK
```

从 staging 切换至常规 ACME 目录

并不支持直接修改账户的 ACME 目录。如果想将一个账户的 ACME 目录从 staging 修改为常规方式，需要先删停用账户，然后新建一个账户。
该步骤也适用于修改 GUI 中的默认 ACME 账户。

示例：将默认的 ACME 账户从 staging 切换至常规目录

```
root@proxmox:~# pvenode acme account info default
Directory URL: https://acme-staging-v02.api.letsencrypt.org/directory
Account URL: https://acme-staging-v02.api.letsencrypt.org/acme/acct/6332194
Terms Of Service: https://letsencrypt.org/documents/LE-SA-v1.2-November ←-
    -15-2017.pdf

Account information:
ID: xxxxxxx
Contact:
    - mailto:example@proxmox.com
Creation date: 2018-07-31T08:41:44.54196435Z
Initial IP: 192.0.2.1
Status: valid

root@proxmox:~# pvenode acme account deactivate default
Renaming account file from '/etc/pve/priv/acme/default' to '/etc/pve/priv/ ←-
    acme/_deactivated_default_4'
Task OK

root@proxmox:~# pvenode acme account register default example@proxmox.com
```

```
Directory endpoints:
0) Let's Encrypt V2 (https://acme-v02.api.letsencrypt.org/directory)
1) Let's Encrypt V2 Staging (https://acme-staging-v02.api.letsencrypt.org/ ←-
    directory)
2) Custom
Enter selection:
0

Attempting to fetch Terms of Service from 'https://acme-v02.api.letsencrypt ←-
    .org/directory'..
Terms of Service: https://letsencrypt.org/documents/LE-SA-v1.2-November ←-
    -15-2017.pdf
Do you agree to the above terms? [y|N]y

Attempting to register account with 'https://acme-v02.api.letsencrypt.org/ ←-
    directory'..
Generating ACME account key..
Registering ACME account..
Registration successful, account URL: 'https://acme-v02.api.letsencrypt.org ←-
    /acme/acct/39335247'
Task OK
```

ACME 证书自动更新

Proxmox VE 服务器节点成功配置 ACME 证书后（使用 pvenode 命令或 GUI 均可），证书将通过 pve-daily-update.service 自动更新。目前，证书将在过期后或到期前 30 天内尝试自动更新。

3.10 主机引导程序

Proxmox VE 目前有两种引导程序可以选用，具体根据安装时采用的磁盘配置决定。
对于使用 ZFS 为根文件系统的 EFI 系统，将使用 systemd-root。其他部署方式将使用标准 grub 引导程序（在 Debian 上安装 Proxmox VE 时将使用这种方式）

3.10.1 安装程序分区方案

Proxmox VE 安装程序会在系统引导盘创建 3 个分区。引导磁盘类型如下：

- 可使用 ext4 或 xfs 文件系统磁盘。
- 可使用 ZFS 的第一个 vdev：

  -第一个磁盘，配置为 RAID0

-所有磁盘，配置为 RAID1，RAIDZ1，RAIDZ2，RAIDZ3

-前两个磁盘，配置为 RAID10

创建的分区有：

- BIOS Boot 分区，容量 1MB（gdisk 类型 EF02）
- EFI 系统分区，容量 512MB（ESP，gdisk 类型 EF00）
- 第三个分区，容量按照 hdsize 参数决定，或使用所选磁盘的剩余空间

在 BIOS 模式（--target i386-pc）下，grub 将安装在所有可启动磁盘的 BIOS Boot 分区，以兼容旧的系统。

3.10.2 Grub

多年来，grub 是 Linux 系统事实上的标准引导程序，文档十分完善。
Grub 将从/boot 目录获取 Linux 内核和 initrd 镜像，配置文件/boot/grub/grub.cfg 包含了内核安装时的相关信息。

配置信息

修改 grub 配置信息，可以通过/etc/default/grub，或配置/etc/default/grub.d 下的相关文件。并可以运行如下命令生成新的/boot/grub/grub.cfg：
‘update-grub‘

3.10.3 Systemd-boot

Systemd-boot 是一个轻量级的 EFI 引导程序。它从 EFI 服务分区（ESP）读取内核和 initrd 镜像。直接从 ESP 加载内核的好处是，不需要为访问存储设备而反复实现驱动。在使用 ZFS 作为主文件系统的场景中，这就意味着可以直接使用主存储池的所有功能特性，不再局限于 grub 实现的功能子集，或专门创建独立的 boot 存储池。
在使用磁盘阵列时（RAID1，RAID10，RAIDZ），所有可引导启动的磁盘（属于第一个 vdev 的磁盘）都会分配一个 ESP，以确保第一个启动设备故障时系统仍然能够启动。各个 ESP 的数据由内核钩子脚本/etc/kernel/postinst.d/zz-pve-efiboot 自动同步。该脚本将指定版本的内核和 initrd 镜像复制到每一个 ESP 根目录的 EFI/proxmox/目录，并创建相应的配置文件，路径是 loader/entries/proxmox-*.conf。辅助脚本 pve-efiboot-tool 用于管理同步的 ESP 和相关内容。
默认配置的内核版本如下：

- 当前运行的内核
- 软件包升级新安装的内核
- 已安装的版本最新的两个内核

- 最近两个内核系列（如 4.15、5.0）的最新版本内核
- 手工指定的内核（详见后续内容）

日常运行中，ESP 将不会被挂载到文件系统，而 grub 会将 ESP 挂载到/boot/efi 下。这能够有效避免 ESP 的 vfat 文件系统损坏，如系统崩溃可能导致文件系统故障，并避免了主引导设备故障时手工调整/etc/fstab 配置的麻烦。

配置信息

sysemd-boot 的配置信息保存在 loader/loader.conf 中。详细配置信息参见 man 手册的 loader.conf(5)页面中。
每条启动引导记录都保存在 loader/entries/的独立目录下
引导记录配置文件示例 entry.conf 如下（/指 ESP 的根目录）：

```
title        Proxmox
version    5.0.15-1-pve
options    root=ZFS=rpool/ROOT/pve-1 boot=zfs
linux      /EFI/proxmox/5.0.15-1-pve/vmlinuz-5.0.15-1-pve
initrd     /EFI/proxmox/5.0.15-1-pve/initrd.img-5.0.15-1-pve
```

手工指定可启动内核

可以使用 pve-efiboot-tool kernel add 命令将指定内核和 initrd 镜像添加到可启动内核列表中。
以下命令将 5.0.15-1-pve 版本的内核添加到内核列表，并同步更新所有 ESP：
pve-efiboot-tool kernel add 5.0.15-1-pve
pve-efiboot-tool kernel list 将列出当前可用的所有版本内核：

```
# pve-efiboot-tool kernel list
Manually selected kernels:
5.0.15-1-pve
Automatically selected kernels:
5.0.12-1-pve
4.15.18-18-pve
```

pve-efiboot-tool kernel remove 命令能够从内核列表删除指定版本内核，示例如下：

```
pve-efiboot-tool kernel remove 5.0.15-1-pve
```

➢ 注意

手工添加或删除内核后，需要执行 pve-efiboot-tool refresh 更新所有的 ESP。

创建新分区并用于同步 ESP

如果需要将一个分区格式化并初始化为同步 ESP，比如在替换 rpool 中的故障 vdev 后，或将一个不支持同步机制的旧文件系统转换为 ESP，可以使用 pve-efiboot-tool 和 pve-kernel-help 命令。

☒ 警告

命令 format 将格式化指定分区，请务必确保使用正确的设备分区参数。

以下示例将一个空分区/dev/sda2 格式化为 ESP：
pve-efiboot-tool format /dev/sda2
以下命令将一个已存在的但未挂载的 ESP 分区加入 Proxmox VE 的内核升级同步机制：
pve-efiboot-tool init /dev/sda2
命令执行后，/etc/kernel/pve-efiboot-uuids 将新增一行对应于新增分区的 UUID 信息，并自动更新所有 ESP。

更新所有 ESP 配置

如需复制并配置所有可用内核，同时确保/etc/kernel/pve-efiboot-uuids 所列出的所有 ESP 分区保持同步，可执行以下命令：
pve-efiboot-tool refresh
（效果等同于基于 grub 引导的系统上执行 update-grub）
该命令一般在修改过内核命令行，或需要同步所有内核和 initrd 镜像时执行。

3.10.4 编辑内核命令行

根据使用的引导程序不同，可以按以下方式修改内核命令行：

Grub

使用 Grub 引导时，内核命令行参数保存在/etc/default/grub 的 GRUB_CMDLINE_LINUX_DEFAULT 参数中。运行 update-grub 命令，可以将其内容附加在/boot/grub/grub.cfg 中所有的 linux 项目后。

Systemd-boot

使用 systemd-boot 引导时，内核命令行参数保存在/etc/kernel/cmdline 中。运行/etc/kernel/postinst.d/zz-pve-efiboot 会将其添加到配置文件 loader/entries/proxmox-*.conf 中的 option 行中。

第 4 章 超融合基础设施

Proxmox VE 虚拟化平台内部集成了计算、存储和网络资源，具有高可用集群管理、备份/恢复以及灾难恢复等特性。所有组件均通过软件定义并互相兼容。
因此可以通过 web 界面实现所有资源和功能的统一管理。这使 Proxmox VE 成为部署管理开源超融合基础设施的理想选择。

4.1Proxmox VE 超融合基础设施的优势

超融合基础设施（HCI）特别适用于预算有限但对基础设施要求较高的场景，如远程分支分支机构部署，私有云或公有云部署等。
HCI 的优势如下：

- 可扩展性：可实现计算、网络和存储的无缝扩展（例如实现服务器、存储的快速独立升级扩容）。
- 低成本：作为开源软件，Proxmox VE 集成了计算、存储、网络、备份和管理等所有功能组件。能够轻松替代昂贵的计算/存储基础设施。
- 数据安全高效：内部集成了备份和灾难恢复功能。
- 简单易用：统一管理界面，配置使用简单。
- 开源软件：有效避免单一厂商依赖。

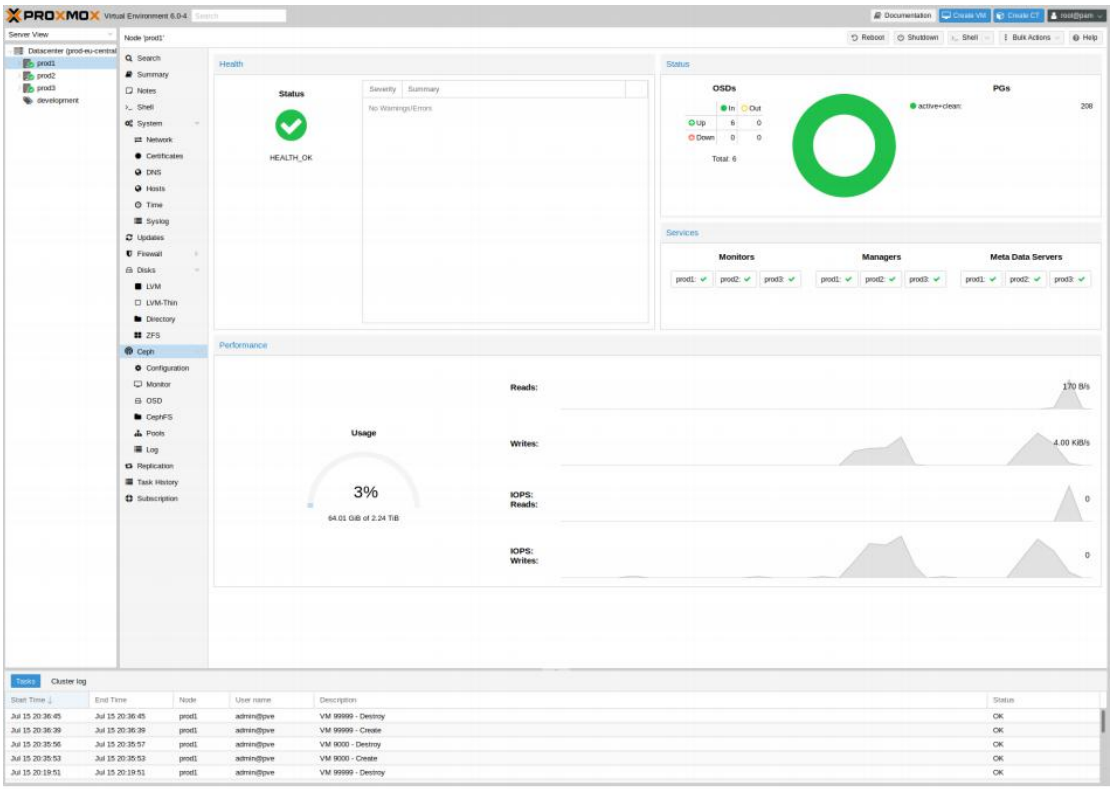
4.2 基于 Proxmox VE 的 Ceph 服务

Proxmox VE 统一了计算和存储功能。集群的物理节点既可以同时用于计算（运行虚拟机和容器）和多副本存储。传统的计算资源和存储资源管理功能可以由统一的超融合应用实现，无需再部署专用存储网络设备（SANs）和网络存储设备（NAS）。通过集成开源软件定义存储平台 Ceph，Proxmox VE 能够直接在虚拟机服务器节点上运行和管理 Ceph 存储。
Ceph 是一个高性能、高可靠、高可扩展的分布式对象存储和文件系统。
Proxmox VE 集成的 Ceph 优势有：

- 可以通过 CLI 和 GUI 轻松安装管理
- 支持薄模式存储
- 支持快照
- 自动修复

- 容量最大可扩充至 exabyte 级别
- 支持多种性能和冗余级别的存储池
- 多副本，高容错
- 可在低成本硬件运行
- 无需硬件 raid 控制器
- 开源软件

对于中小规模部署场景，可以直接将 Ceph 安装到 Proxmox VE 服务器，以实现 RADOS 块设备（RBD）功能。详见 8.14 节“基于 Ceph RADOS 块设备的后端存储”。当前主流硬件拥有足够强大的 CPU 和内存资源，能够满足在同一节点同时运行虚拟机和存储服务的需要。为简化管理，Proxmox VE 提供了 pveceph 工具来安装管理 Ceph 服务。
Ceph 包含一组后台服务，以实现 RBD 存储功能[a]：



- Ceph Monitor（ceph-mon）
- Ceph Manager（ceph-mgr）
- Ceph OSD（ceph-osd；对象存储服务）

➢ 提示

[a] Ceph 简介参见 http://docs.ceph.com/docs/master/start/intro/

强烈建议了解熟悉一下 Ceph 相关架构[b]和概念。[c]

4.2.1 前置条件

创建超融合的 Proxmox+Ceph 集群至少需要 3 台独立服务器。
更多建议参考 Ceph 网站。

CPU

应优先选用高主频 CPU 以降低延时。一个简单的经验是，为每个 Ceph 服务分配一个 CPU 核心（或 CPU 线程），以确保 Ceph 拥有足够资源，能够以良好性能稳定运行。

内存

采用超融合部署方式时，需要密切关注内存使用情况。除了虚拟机和容器必须使用的内存外，Ceph 也需要有足够内存，才能提供稳定且良好的性能。一个经验是，每 1TiB 数据，OSD 需要占用 1GiB 内存。OSD 缓存还需要额外配置内存。

网络

建议为 Ceph 准备专用的 10Gb 或者更高性能的网络。如果没有 10Gb 交换机设备，也可以使用网状网络[d]。
高负载网络通信，特别是虚拟机恢复时的流量，将影响运行在同一网络上的服务，很有可能造成 Proxmox VE 集群崩溃。
建议认真估算网络带宽需求。单块硬盘可能不能压满 1Gb 链路，但多块硬盘组成的 OSD 就可以。主流 NVME SSD 完全可以压满 10Gbps 带宽。采用更高带宽性能的网络，可以确保网络任何时候都不会成为性能瓶颈。为此，25Gb，40Gb，100Gb 的网络都值得考虑。

存储盘

在规划 Ceph 集群时，需要重点考虑恢复时间因素。对于小规模集群，恢复时间可能会非常长。推荐在小规模集群中使用固态 SSD 盘代替 HDD 硬盘，以缩短恢复时间，降低恢复期间发生二次故障的风险。
通常情况下，SSD 的 IOPs 比传统磁盘高的多，但价格也更贵，可以参考 4.2.9 节组建不同类型的存储池，以提高恢复性能。也可以参考 4.2.7 节内容，使用高速存储盘作为 DB/WAL 设备，加速 OSDs。如果同时为多个 OSDs 配置了高速存储盘，需要考虑平衡 OSD 和 WAL/DB（卷）盘的配比，以避免高速存储盘成为相关 OSDs 的性能瓶颈。
除了选择合适存储盘类型，还可以选择为单一节点配置偶数个对称存储盘，以提高 Ceph 性能。例如，单一节点使用 4 块 500GB 存储盘时的性能就比混合使用 1 块 1TB 盘和 3 块 250GB 盘要好。
此外，还需要妥善平衡 OSD 数量和单一 OSD 容量。大容量 OSD 可以增加存储密度，但也意味着在 OSD 故障时，Ceph 需要恢复更多数据。

---

[b] Ceph 架构参见 http://docs.ceph.com/docs/luminous/architecture/
[c] Ceph 概念参见 http://docs.ceph.com/docs/luminous/glossary
[d] Ceph 网状网络配置参见 https://pve.proxmox.com/wiki/Full_Mesh_Network_for_Ceph_Server

不要使用硬 RAID

Ceph 直接处理数据对象冗余和多重并发磁盘（OSDs）写操作，因此使用硬 RAID 控制器并不能提高性能和可用性。相反，Ceph 需要直接控制磁盘硬件设备。硬件 RAID 控制器并非为 Ceph 所设计，其写操作管理和缓存算法可能干扰 Ceph 对磁盘的正常操作，从而把事情复杂化，并导致性能降低。

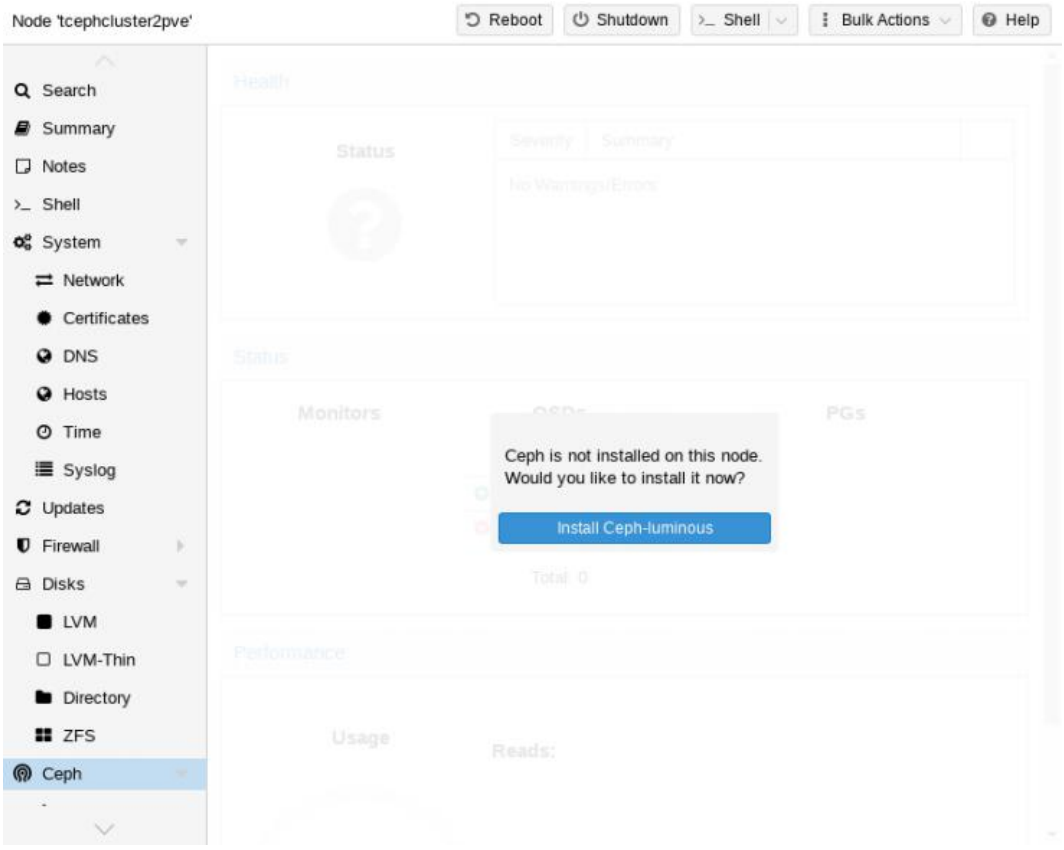
☒ 警告

不要使用硬件 RAID 控制器，可改用主机 HBA 卡。

➢ 提示

以上是关于硬件选型的一个粗略建议。具体还要结合需求特点，并对部署进行测试，以及对 Ceph 健康状况和性能进行持续观测，才能判定是否满足需要。

4.2.2 初始化 Ceph 安装和配置



Proxmox VE 提供了简单易用的 Ceph 安装向导。选中集群中的一个节点，然后在菜单树中打开 Ceph 菜单区，就可以开始安装 Ceph 了。

安装向导有多个步骤，每个步骤都需要执行成功才可以完成 Ceph 安装。开始安装操作后，

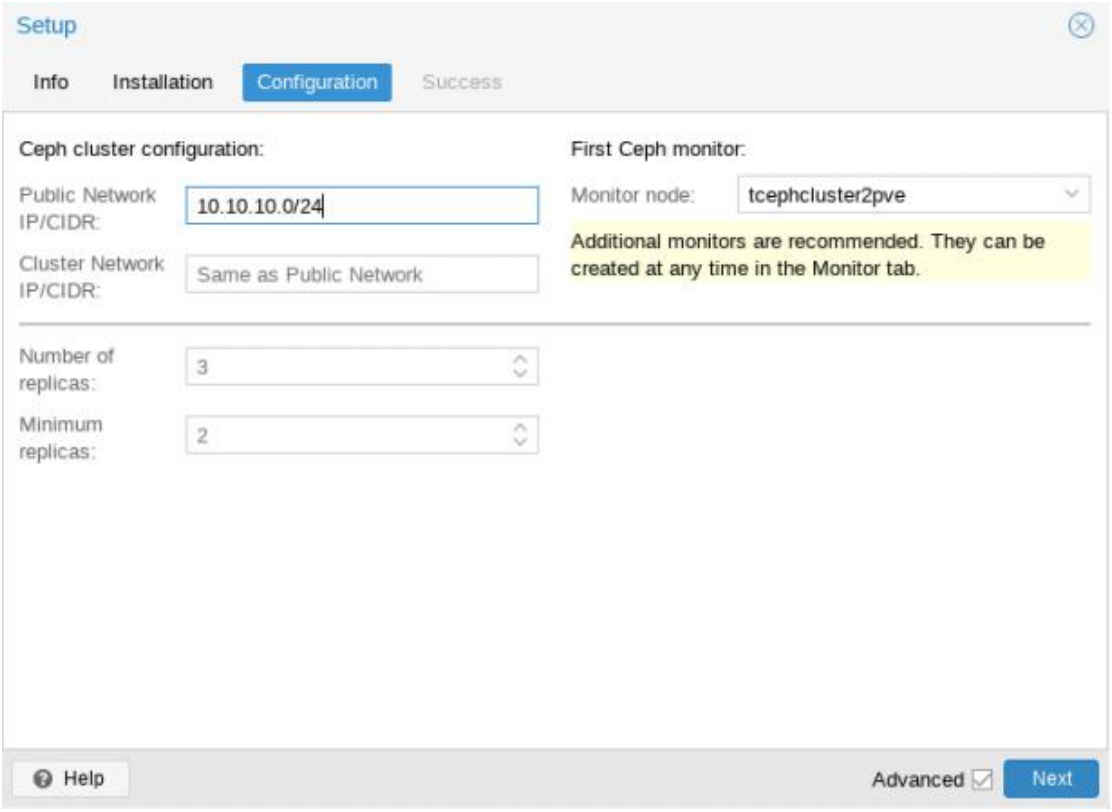
向导会自动从 Proxmox VE 的 ceph 软件源下载软件包并完成安装。
完成第一个步骤后，还需要创建配置。对每个集群，生成的配置信息会通过第 7 章所述 Proxmox VE 集群文件系统（pmxcfs）自动分发到其他节点，所以该操作只需要执行一次即可。
创建的配置包括以下信息：

- Public Network

为避免影响集群通信等其他对网络延迟敏感的服务，也为了提高 Ceph 性能，强烈建议为 Ceph 准备一个专门的独立网络，将 Ceph 流量隔离开来。

- Cluster Network

进一步，还可以设置 Cluster Network，将 OSD 复制和心跳流量隔离出来。这将有效降低 public network 的负载，并有效改善大规模 Ceph 集群的性能。



以下两个参数项属于高级配置功能，仅供专家级用户使用。

- Number of replicas

设置副本数量。

- Mininum replicas

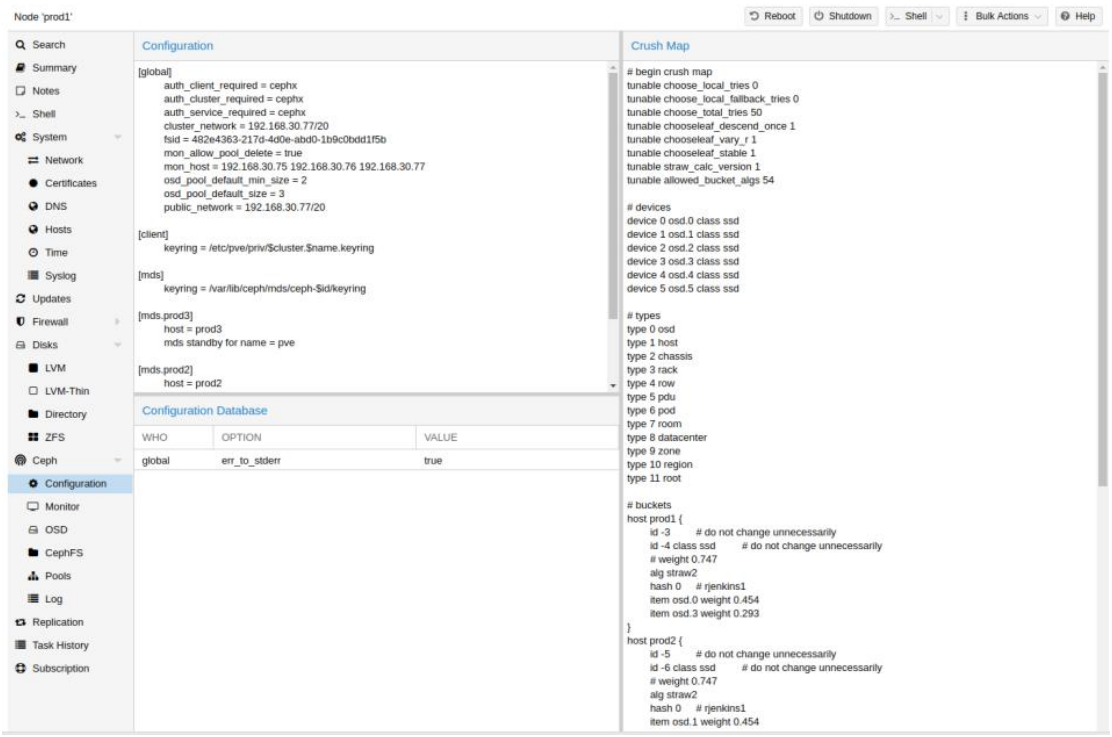
设置最小副本数量，副本数低于该阀值时，数据将会被标记为不完全状态。

此外，还需要选择第一个监视器节点。
所有配置完成后，系统会提示配置成功，并给出下一步安装指令。接下来，可以按 4.2.5 节创建 monitors，按 4.2.7 节创建 OSDs，按 4.2.8 节创建一个 pool，之后就可以使用 Ceph 了。本章后续内容将介绍如何在 Proxmox VE 中使用 Ceph，包括前述内容，以及 4.2.11 节的 CephFS。

4.2.3 安装 Ceph

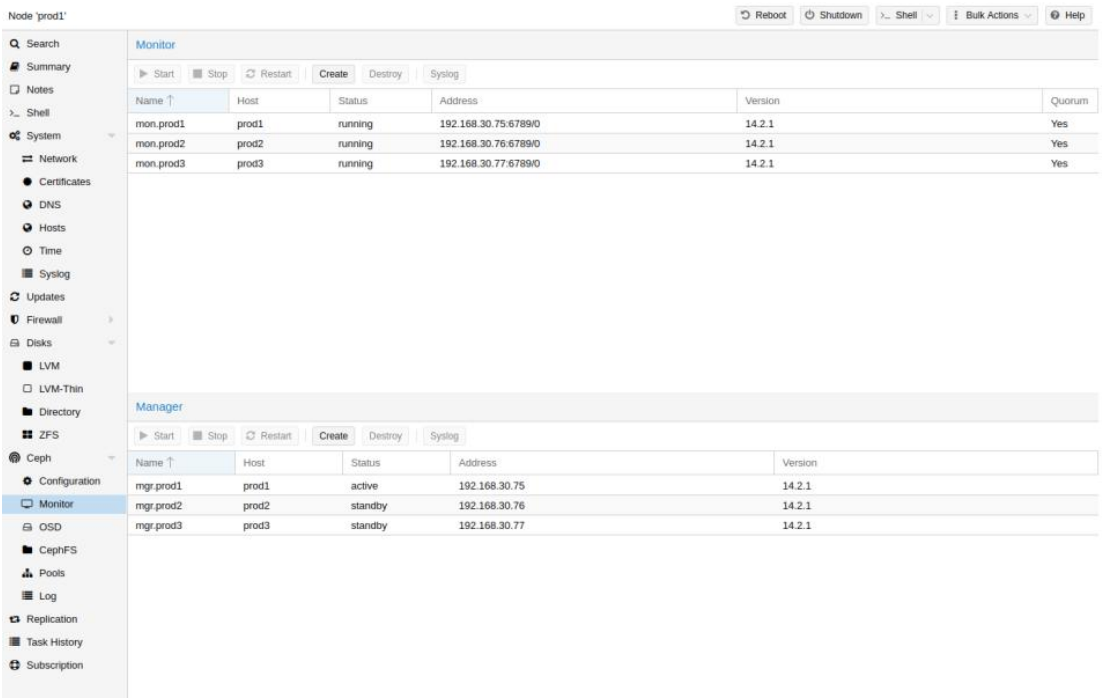
在每个节点运行如下脚本命令：
pveceph install
该命令将创建 apt 软件源/etc/apt/sources.list.d/ceph.list，并安装所需软件。

4.2.4 初始化 Ceph



使用 Proxmox VE Ceph 安装向导（推荐），或在一个节点上运行以下命令：
pveceph init –network 10.10.10.0/24
该命令将创建为 ceph 创建一个专用网络，并将初始配置写入文件/etc/pve/ceph.conf。该文件将通过第 7 章介绍的 pmxcfs 自动分发到所有 Proxmox VE 服务器节点。该命令还将创建符号链接/etc/ceph/ceph.conf 指向该配置文件，以便直接运行 Ceph 命令，无需另外创建配置文件。

4.2.5 创建 Ceph Monitor



Ceph Monitor（MON）[e]负责管理集群全局数据。如需要实现 HA，则至少需要创建 3 个 Monitor。安装向导会自动创建一个 monitor。对中小规模集群而言，最多 3 个 monitor 就够了，只有大型集群才需要更多的 monitor。

可以在你想要部署 monitor 的节点上（建议创建 3 个 monitor），可以在 GUI 界面依次选择 Ceph→Monitor 选项完成 monitor 创建，也可以运行如下命令。

pveceph createmon

运行该命令将默认同时安装 Ceph Manager（ceph-mgr）。如果不需要安装，可以在执行命令行增加选项-exclude-manager。

4.2.6 创建 Ceph Manager

Ceph Manager 是独立于 monitor 的另一个服务。主要提供监控 Ceph 集群运行状态的接口。从 luminous 版本开始，ceph-mgr[f]服务成为 Ceph 必选组件。Ceph manager 一般和 monitor 同时完成安装。

➢ 注意

建议将 Ceph Manager 部署在 monitor 节点上。并考虑安装多个 Cepnmanager，以满足高可用要求。

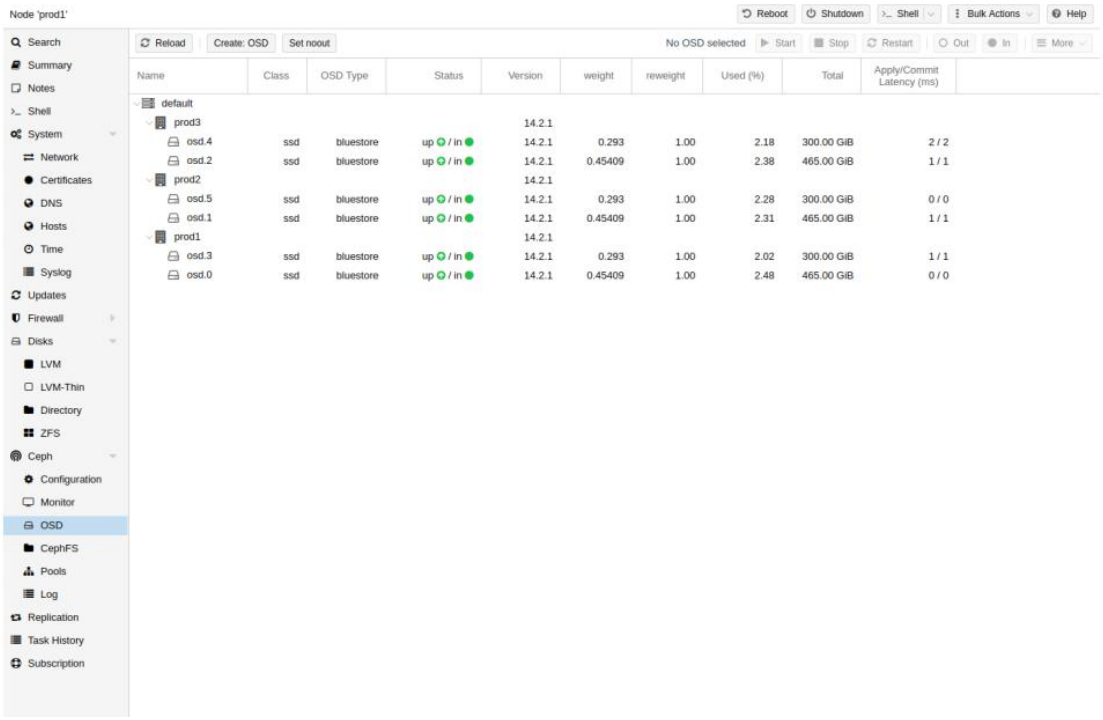
---

pveceph createmgr

[e] Ceph Monitor 详见 http://docs.ceph.com/docs/luminous/start/intro/

[f] Ceph Manager 详见 http://docs.ceph.com/docs/luminous/mgr/

4.2.7 创建 Ceph OSD



可以通过 GUI 或命令行创建 OSD。创建命令如下：

```
pveceph createosd /dev/sd[X]
```

> ➢ 提示
>
> 建议至少为 Ceph 集群创建 12 个 OSD，并平均分配到集群的各节点。在最小的 3 节点集群下，每个节点部署 4 个 OSD。

如磁盘之前已经被格式化过（如 ZFS/RAID/OSD），可用以下命令删除分区表、引导扇区和其他 OSD 遗留数据。

```
ceph-volume lvm zap /dev/sd[X] --destroy
```

> ☒ 警告
>
> 以上命令将删除磁盘上的所有数据。

Ceph Bluestore

从 Kraken 版本开始，Ceph 引入一种新的 Ceph OSD 存储类型，即 Bluestore[9]。而在 luminous 版 Ceph 中，该类型已成为默认 OSD 类型。

```
pveceph createdosd /dev/sd[X]
```

[9] Ceph Bluestore 详见 http://ceph.com/community/new-luminous-bluestore/

Block.db 和 block.wal

如果要为 OSD 配置专门独立的 DB/WAL 设备，可以通过-db_dev 和-wal_dev 选项指定设备。如果不指定专门设备，WAL 数据将保存在 DB 上。
pveceph createosd /dev/sd[X] -db_dev /dev/sd[Y] -wal_dev /dev/sd[Z]
你可以用-db_size 和-wal_size 参数直接设置相关设备大小。如果未明确设置，将依次尝试使用以下值：

- 使用 ceph 配置中的 bluestore_block_{db,wal}_size

-…数据库，osd 区块
-…数据库，global 区块
-…文件，osd 区块
-…文件，global 区块

- OSD 大小的 10%(DB)/1%(WAL)

➢ 注意

DB 保存了 BlueStore 的内部元数据。WAL 是 BlueStore 的内部卷，用于保存预写日志。建议采用高性能 SSD 或 NVRAM 作为 DB/WAL，以提高性能。

Ceph Filestore

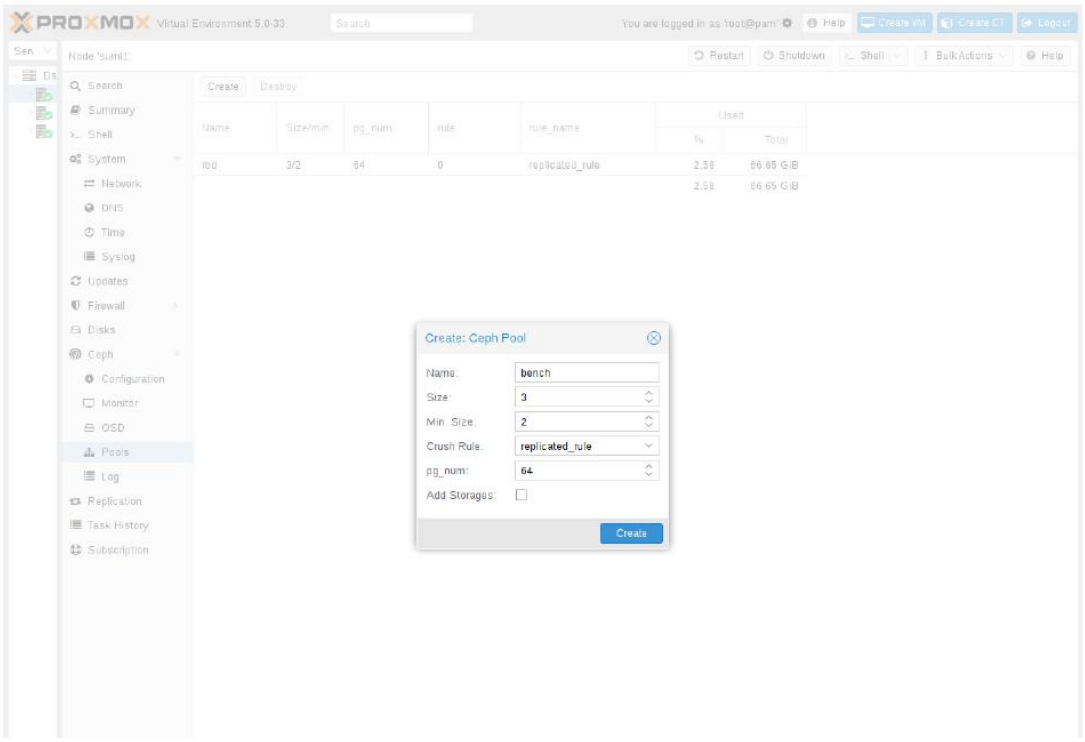
在 Ceph luminous 之前，Ceph OSD 都采用 Filestore 存储模式。从 Ceph Nautilus 开始，Proxmox VE 不再支持使用 pveceph 创建该类 OSD。如果确实需要创建 Filestore 类的 OSD，可以使用 ceph-volume 命令。
ceph-volume lvm create --filestore --data /dev/sd[X] --journal /dev/sd[Y]

4.2.8 创建 Ceph Pool

存储池 pool 是一组存储对象的逻辑集合。具体是由一组 **P**lacement **G**roups（PG，pg_num）的对象集合组成。
当不使用任何选项时，默认将创建 128 个 PG，并使用 3 副本模式，降级模式最低使用 2 副本模式。

➢ 注意

默认 PG 数量适用于 2-5 块磁盘的场景。当集群内 PG 数量太少或太多时，Ceph 将发出“HEALTH_WARNING”告警。



建议根据你的具体配置计算确定 PG 数量。互联网上可以找到 PG 数量计算公式或 PG 数量计算器[h]。Ceph 一旦投入运行，PG 数量只可以调整增加，但不能再减少。
可以在 GUI 主机管理界面选择 Ceph→Pools 或直接用命令行创建 pool。
pveceph createpool <name>

如果要自动获取 pool 的存储定义，可以在 GUI 上勾选“Add storages”框，或使用命令行创建 pool 时使用选项 –add_storages。
关于 Ceph pool 管理的进一步信息可以查看 Ceph pool 管理手册[i]。

4.2.9Ceph CRUSH 和设备类别

Ceph 是以算法 **C**ontrolled **R**eplication **U**nder **S**calable **H**ashing（CRUSH[j]）为基础创建的。CRUSH 算法用于计算数据存取的位置，且无需中心索引服务的支持。CRUSH 基于构成存储池 pool 的 OSD、buckets（设备位置）和 rulesets（数据复制规则）来完成计算。

➢ 注意

关于 CRUSH 图的进一步信息，可以查看 Ceph 官方文档中 CRUSH 图[k]一节。

调整该图可以反映不同层次的复制关系。对象副本可以分布在不同地方(例如，各故障区域)，并同时保持期望的分布。

[h] PG 计算机详见 http://ceph.com/pgcalc/
[i] Ceph pool 管理详见 http://docs.ceph.com/docs/luminous/rados/operations/pools/
[j] CRUSH 详见 https://ceph.com/wp-content/uploads/2016/08/weil-crush-sc06.pdf
[k] CRUSH map 参见 http://docs.ceph.com/docs/luminous/rados/operations/crush-map/

常见用法是为不同的 Ceph pool 配置不同类别的磁盘。为此，Ceph luminous 引入了设备类的概念，以简化 ruleset 的创建。
设备类信息可以用命令 ceph osd tree 查看。各个类代表了各自的根位置。命令如下。
ceph osd crush tree –show-shadow

以上命令的输出示例如下：

ID	CLASS	WEIGHT	TYPE	NAME
-16	nvme	2.18307	root	default~nvme
-13	nvme	0.72769	host	sumi1~nvme
12	nvme	0.72769		osd.12
-14	nvme	0.72769	host	sumi2~nvme
13	nvme	0.72769		osd.13
-15	nvme	0.72769	host	sumi3~nvme
14	nvme	0.72769		osd.14
-1		7.70544	root	default
-3		2.56848	host	sumi1
12	nvme	0.72769		osd.12
-5		2.56848	host	sumi2
13	nvme	0.72769		osd.13
-7		2.56848	host	sumi3
14	nvme	0.72769		osd.14

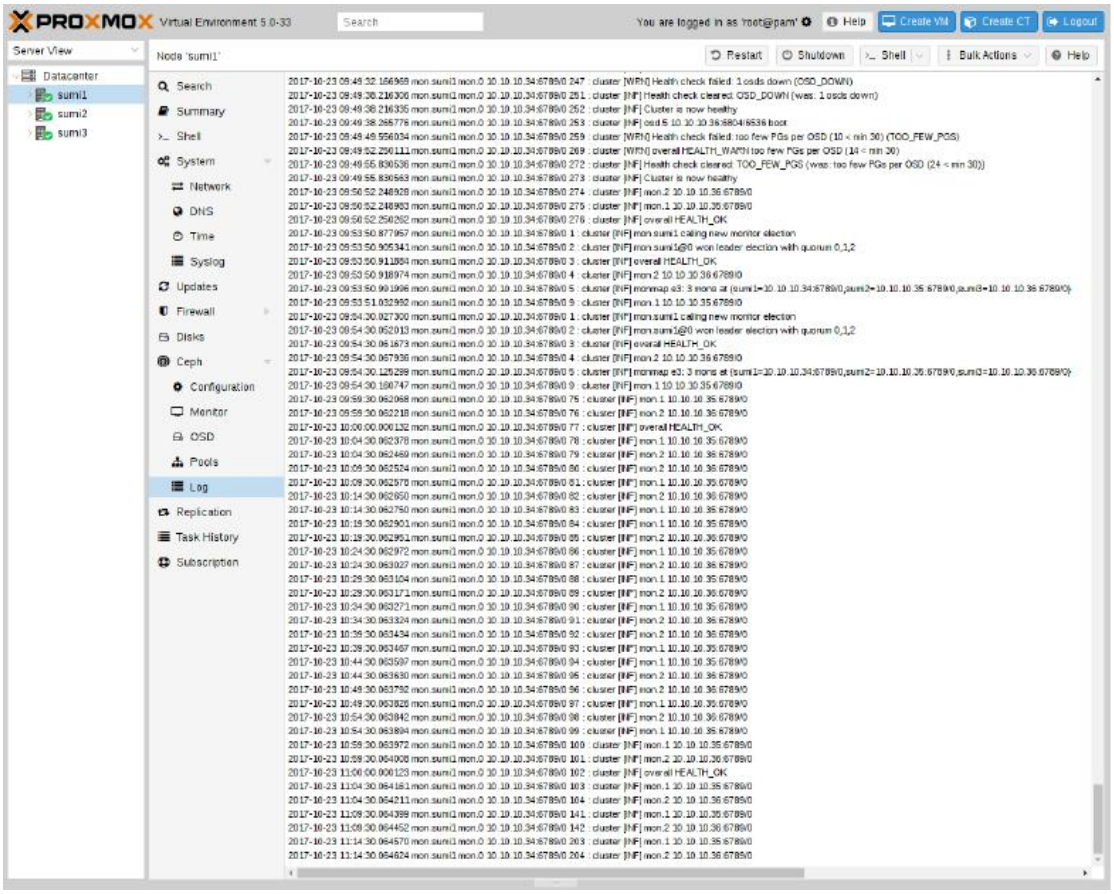
如果要让 pool 将对象保存在指定设备类上，需要用指定设备类创建 ruleset。
ceph osd crush rule create-replicated <rule-name> <root> <failure-domain> <class>

<rule-name> 规则名称，用于和 pool 关联（见 GUI 和 CLI）
<root> 规则所属的 CRUSH 根名称（默认 ceph root 为“default”）
<failure-domain> 对象所属的故障域（通常为 host）
<class> 要使用的 OSD 存储类名称（例如 nvme，ssd，hdd）
在 CRUSH 图中创建了规则后，可以指示 pool 启用该 ruleset。
ceph osd pool set <poot-name> crush_rule <rule-name>

➢ 提示

如果 pool 中已有数据，则现有数据将根据规则移动位置。这有可能对集群性能产生重大影响。你也可以新建一个存储池，然后将磁盘逐个迁移过去。

4.2.10Ceph 客户端



接下来可以配置 Proxmox VE 使用 pool 存储虚拟机或容器镜像。通过 GUI 增加 RBD 存储即可（参见 8.14 节“基于 Ceph RADOS 块设备的后端存储”）

也可以将 keyring 复制到外部 Ceph 集群指定位置。如果 Ceph 就安装在 Proxmox 节点，该操作将自动完成。

➢ 注意

文件名称需要采用<storage_id>+’.keyring 的格式。其中<storage_id>配置文件/etc/pve/storage.cfg 中 rbd:后面的存储名称。下面例子中采用 my-ceph-storage 的名称。

---

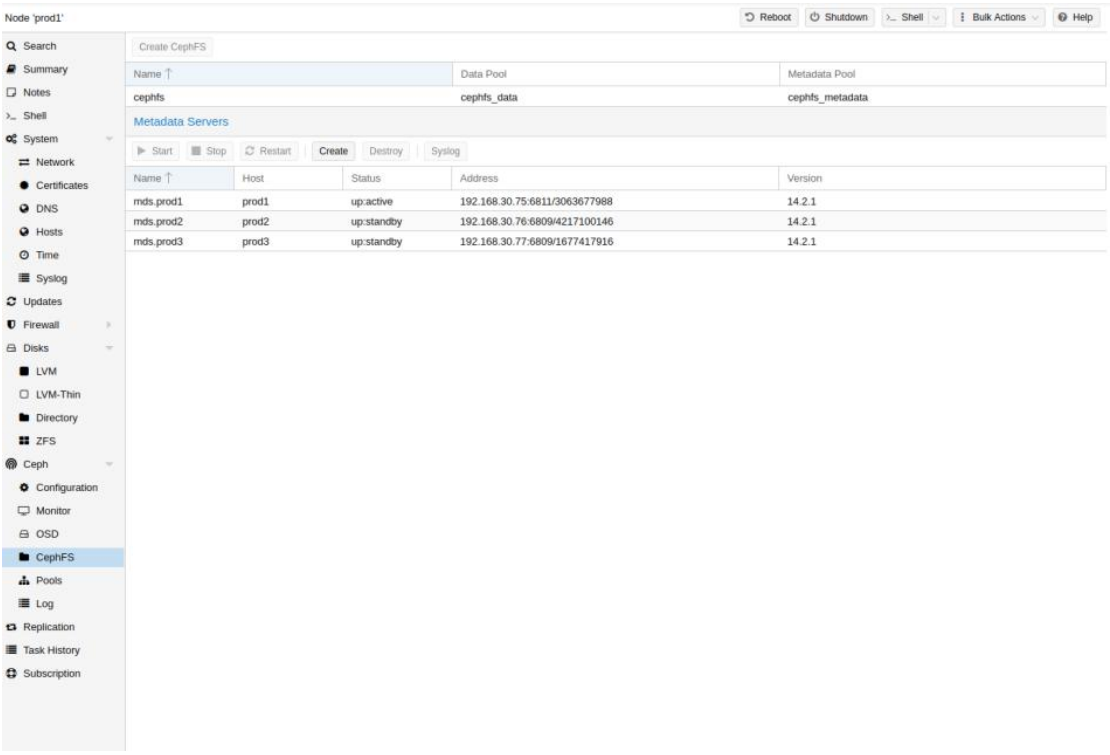
mkdir /etc/pve/priv/ceph

cp /etc/ceph/ceph.client.admin.keyring /etc/pve/priv/ceph/my-ceph-storage.keyring

4.2.11 CephFS

Ceph 也支持基于 RADOS 块设备的文件系统。元数据服务器（MDS）可以将 RADOS 块映射

为文件和目录，并提供兼容 POSIX 标准的多副本文件系统。用户可以很容易在 Ceph 上建立高可用的集群共享文件系统。元数据服务器可以确保文件平均分布在整个 Ceph 集群上，在高负载下也能有效避免单一节点过载，而这往往 NFS 等传统共享文件系统的一大痛点。



Proxmox VE 支持创建超融合的 CephFS，也支持挂载外部 CephFS，并可用于保存备份，ISO 文件，容器模板等。

元数据服务器（MDS）

为了使用 CephFS，至少需要配置一个元数据服务器。通过 Proxmox VE 的 GUI 界面，可以很容易创建元数据服务器，只需依次在打开 Node→CephFS 控制面板即可找到操作界面，也可以通过执行以下命令：

pveceph mds create

一个集群内也可以创建多个元数据服务器。但默认设置同一时间只能有一个元数据服务器处于活动状态。如果 MDS，或者其所在节点失去响应（或者崩溃），某个 standby 的 MDS 将自动转为 active。可以通过设置 hotstandby 参数加速主备切换，或者在对应 MDS 的 ceph.conf 文件中进行如下设置：

mds standby replay = true

启用该设置后，该备用 MDS 将持续轮训活动 MDS 的状态，相当于处于一种温备状态，能够在主 MDS 宕机后更快接管。当然，持续轮训会消耗一定资源，并对活动 MDS 的性能产生一定影响。

多主 MDS

从 Luminous（12.2.x）版本开始，可以有多个活动的元数据服务器同时运行，但这通常只在多个并发客户端的场景中有意义，MDS 很少成为性能瓶颈。如果想使用该特性，情参考 Ceph 文档。

创建 CephFS

在 Proxmox VE 下，可以通过 Web GUI、CLI、外部 API 接口等多种方式轻松创建 CephFS。
前置条件如下：
创建 CephFS 的前置条件：

- 按照 4.2.3 节安装 Ceph 软件包，如果之前已经安装了 Ceph，也可以再次安装升级到最新版，同时确保 CephFS 相关的包也被安装上。
- 按照 4.2.5 节内容设置 Monitors
- 按照 4.2.7 节内容设置 OSDs
- 按照 4.2.11 节创建至少一个 MDS

完成以上操作后，就可以通过 Web GUI 的 Node→CephFS 面板或者命令行工具 pveceph 创建 CephFS 了。示例如下：
pveceph fs create --pg_num 128 --add-storage

上面的命令将创建一个名为“cephfs”的 CephFS 存储池，数据存储名称为“cephfs_data”，配置 128 个数据集，元数据存储名称为“cephfs_metadata”，配置 32 个数据集，也就是数据存储的四分之一。可查看 4.2.8 节或 Ceph 文档以确定适当的存储集数量（pg_num）。此外，“--add-storage”参数将自动把创建成功的 CephFS 添加到 Proxmox VE 的存储配置文件中。

删除 CephFS

☒ 警告

删除操作后，CephFS 上所有数据都将不可继续使用。且该操作无法撤销。

如果确认需要删除 CephFS，首先需要停止或删除所有的元数据服务器（MDS）。该操作可通过 Web GUi 或如下命令进行：
pveceph mds destroy NAME
该命令需要在所有运行了 MDS 的 Promox VE 节点上执行。

然后可用如下命令删除 CephFS：
ceph rm fs NAME --yes-i-really-mean-it
该命令只需在某个运行了 Ceph 的节点执行一次即可。然后可以删除创建的数据存储和元数据存储。可在 Web GUI 操作或运行以下命令行：
pveceph pool destroy NAME

4.2.12 Ceph 监控和故障排查

最好从安装 Ceph 后就开始持续监控 Ceph 的健康状态。可以通过 ceph 自带工具，也可以通过 Proxmox VE API 监控。
以下命令可以查看集群是否健康（HEALTH_OK），或是否存在警告（HEALTH_WARN）或错误（HEALTH_ERR）。如果集群状态不健康，以下命令还可以查看当前事件和活动情况概览。

single time output
pve# ceph -s
continuously output status changes (press CTRL+C to stop)
pve# ceph -w
如果要查看进一步详细信息，可以查看/var/log/ceph/下的日志文件，每个 ceph 服务都会在该目录下有一个日志文件。如果日志信息不够详细，还可以进一步调整日志记录级别。
可以在官网查看 Ceph 集群故障排查的进一步信息。

第 5 章 图形用户界面

Proxmox VE 使用起来非常简单。你无需安装独立的管理工具，一切管理操作都可以通过浏览器完成（推荐使用最新版的 Firefox 或 Google Chrome 浏览器）。你既可以通过内置的 HTML5 控制台访问虚拟机和容器的桌面，也可以使用 SPICE 终端软件访问。
由于我们采用 Proxmox 集群文件系统（pmxcfs）管理集群，你可以从集群中任何一个节点管理整个集群。再次强调，每个节点都可以管理整个集群。无需安装任何独立的管理节点。
可以使用任何主流浏览器访问 web 管理接口。如果 Proxmox VE 检测到你在用移动终端设备访问 Web 管理界面，系统会自动跳转到一个专为触摸屏设备设计的轻量级的管理界面。
Web 管理界面的访问地址为 https://你的服务器 IP 地址:8006（默认用户为 root，口令为安装时设置的 root 口令）。

5.1 功能

- 无缝集成 Proxmox VE 集群管理功能。
- 基于 AJAX 的动态资源升级。
- 基于 SSL 加密的虚拟机和容器远程访问（https）。
- 快速搜索功能，可方便地管理几千台虚拟机。
- 基于 HTML5 的 SPICE 安全终端。
- 基于角色的访问权限管理（虚拟机，存储，节点等等）。
- 支持多种身份认证方式（例如本地操作系统用户，微软活动目录，LDAP 等）。
- 支持双因子认证（OATH，Yubikey）。
- 基于 Ext JS 6.x JavaScripte 框架技术开发。

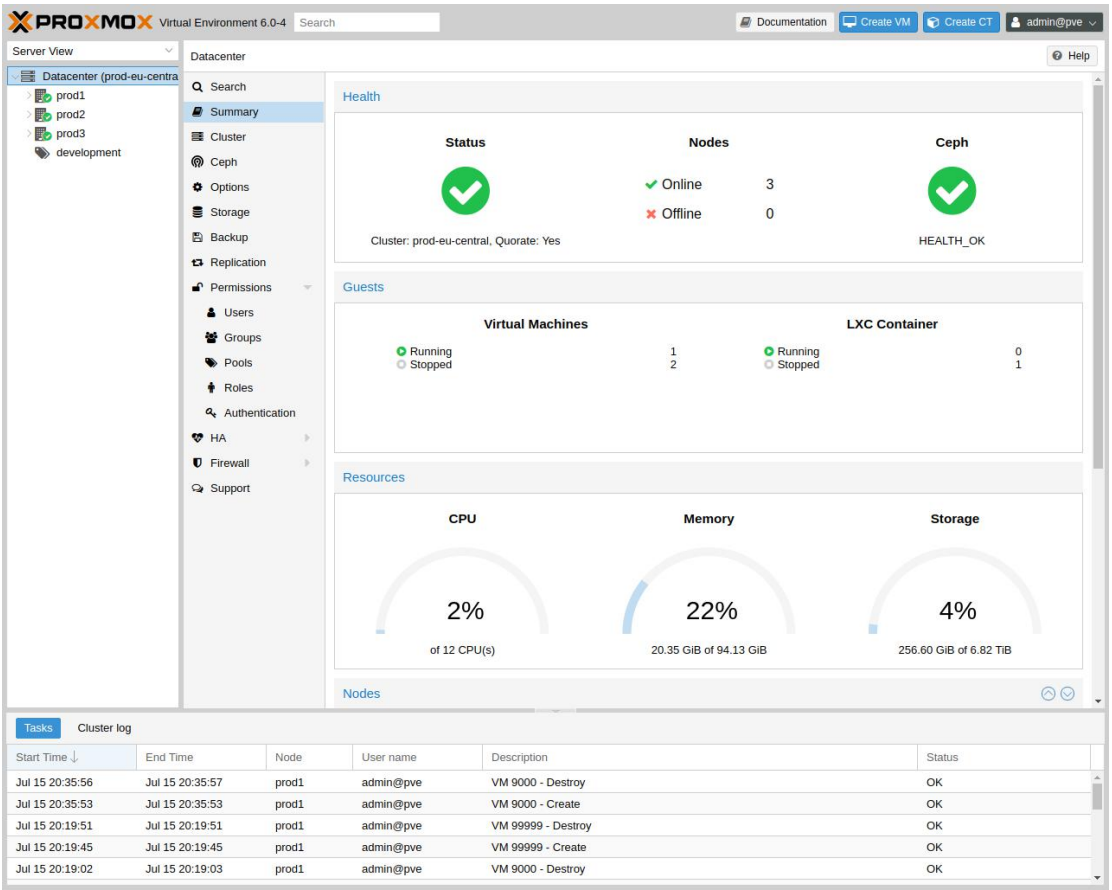
5.2 登录

连接到服务器时，首先看到的是登录窗口。Proxmox VE 支持多种身份认证方式（Realm），并支持多种可选择的语言。目前 GUI 支持多达 20 种语言。

➢ 注意

可以勾选对话框右下角复选框保存用户名，以便下次登录重复输入用户名。

5.3 GUI 概览



GUI 管理界面由 4 个区域组成：

- 标题栏　　位于正上方。用于展示状态信息，并包含有重要功能的操作按钮。
- 资源树　　位于左侧。以树状形式展示各类资源对象，并供管理员选择。
- 内容面板　　位于中间。用于展示左侧导航栏所选中对象的详细信息。
- 日志面板　　位于正下方。展示最近任务的日志信息。双击相关日志信息可以进一步获得详细信息，或中止任务。

➢ 注意

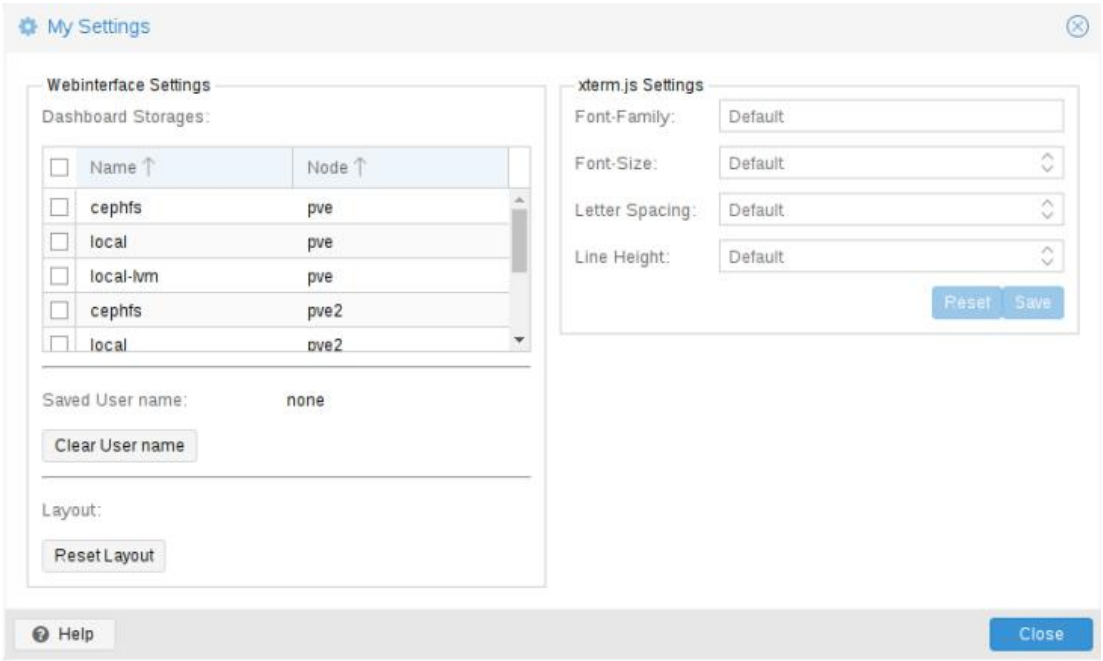
可以缩小或扩大资源树和日志面板大小，也可以彻底隐藏日志面板。这样的调整可以帮助你你合理使用较小尺寸的显示器，从而更好展示其他内容。

5.3.1 标题栏

左上方是 Proxmox 的 logo，其后是当前所安装的 Proxmox VE 版本。旁边的搜索栏可以搜索特定资源对象（虚拟机，容器，节点等）。这往往比在资源树中查找要更快。
搜索栏右侧显示的是用户名（登录名称）。齿轮形状的图标是一个按钮，用于打开“我的设置”对话框。你可以在对话框中进行用户界面设置（重置保存的用户名，重置保存的界面布局）。
标题栏最右侧是 4 个按钮：

- 帮助按钮　　点击可打开帮助文档。
- 创建虚拟机按钮　　点击可打开虚拟机创建向导对话框。
- 创建容器按钮　　点击可打开容器创建向导对话框。
- 退出按钮　　点击可退出管理界面，并回到登录对话框界面。

5.3.2 我的设置



在我的设置窗口里，可以设置本地存储参数。通过其中的存储设置面板可以启用停用指定存储设备，控制面板显示的可用存储空间也会相应实时变化。如果没有选中任何存储，则总的存储空间就是所有存储设备之和。

存储控制面板下方可以查看保存的用户名，还有清除保存用户按钮，以及将 GUI 布局重置为默认布局的按钮。

右侧是 xterm.js 设置界面，包含以下参数：

Font-Family	xterm.js 使用的字体（例如 Arial）
Font-Size	字体大小
Letter Spacing	增加或减小字符间距
Line Height	行高绝对值

5.3.3 资源树

资源树是最重要的导航界面。资源树最上方是视图下拉菜单，提供几种不同的资源树结构视图。默认视图为“服务器视图”，以下是该视图展示的资源对象类型：

- 数据中心　　展示集群级别设置信息（作用于所有节点）。
- 节点　　集群内单个物理服务器。物理服务器承载客户机的运行。
- 客户机　　指虚拟机、容器和模板。
- 存储　　指数据存储服务。
- 资源池　　指为便于管理而将若干客户机编成的一个组。

以下是可选的视图类型：

- 服务器视图　　展示所有类型对象，按节点分组展示。
- 文件夹视图　　展示所有类型对象，按对象类型分组展示。
- 文件夹视图　　仅展示存储对象，按节点分组展示。
- 资源池视图　　仅暂时虚拟机和容器，按资源池分组展示。

5.3.4 日志面板

日志面板主要用于展示集群当前正在运行的任务状态信息。后台执行的工作称为任务（task），如后台执行的新建虚拟机操作就是一个任务。
任务执行过程中输出的任何信息都会被保存到独立日志文件。双击任务日志条目即可查看详细的日志信息。同时也可以在日志查看界面中止正在运行的任务。
请注意，集群所有节点当前运行的任务在日志面板上展示出来，所以你可以实时查看其他用户在其他节点执行的任务运行状态。

➢ 注意

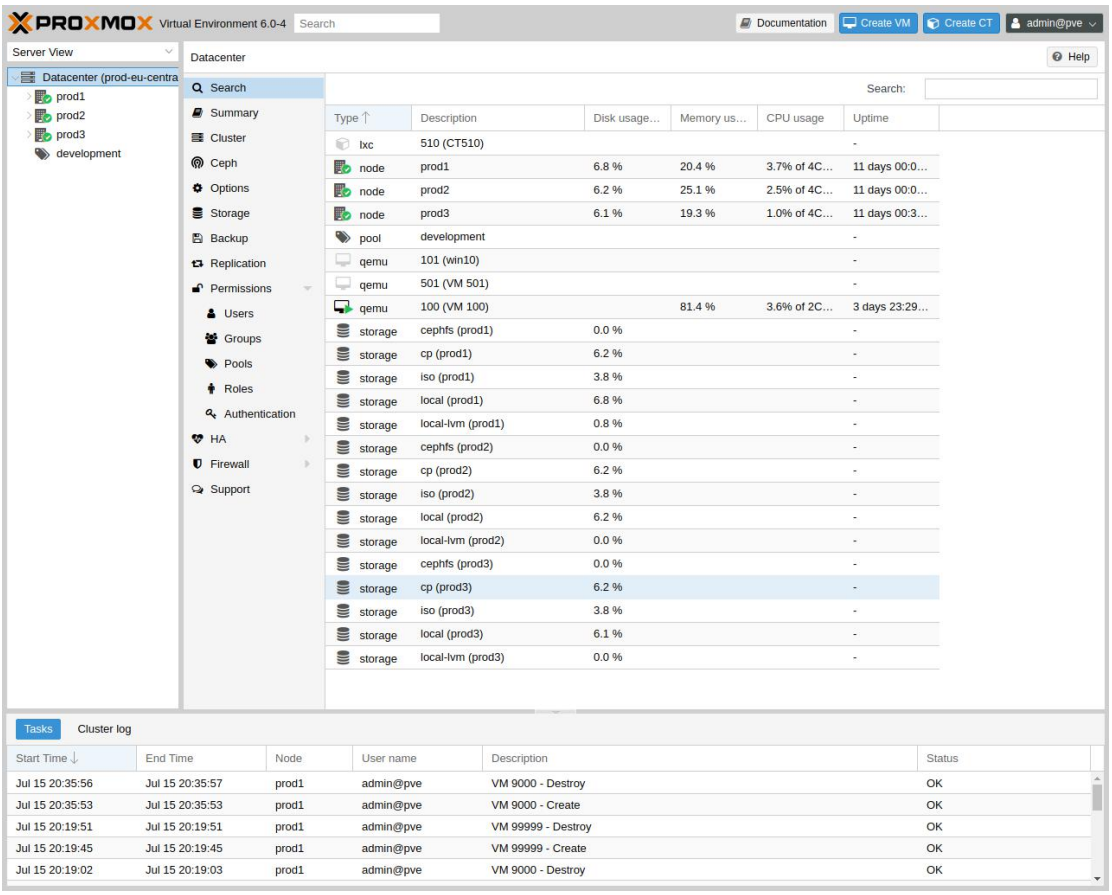
为确保日志列表简介，系统会自动删除旧的且已完成的任务日志。但你仍然可以在节点面板的历史日志栏目中找到这些任务日志信息。

部分小任务会将其日志发送给所有的集群成员。你可以在集群日志面板查看这些日志信息。

5.4 内容面板

选中资源树中某个资源后，其对应的配置信息和状态信息会自动显示在内容面板中。以下章节将简要介绍内容面板的功能。更详细信息可以参考本用户手册后续各个专门章节的内容。

5.4.1 数据中心



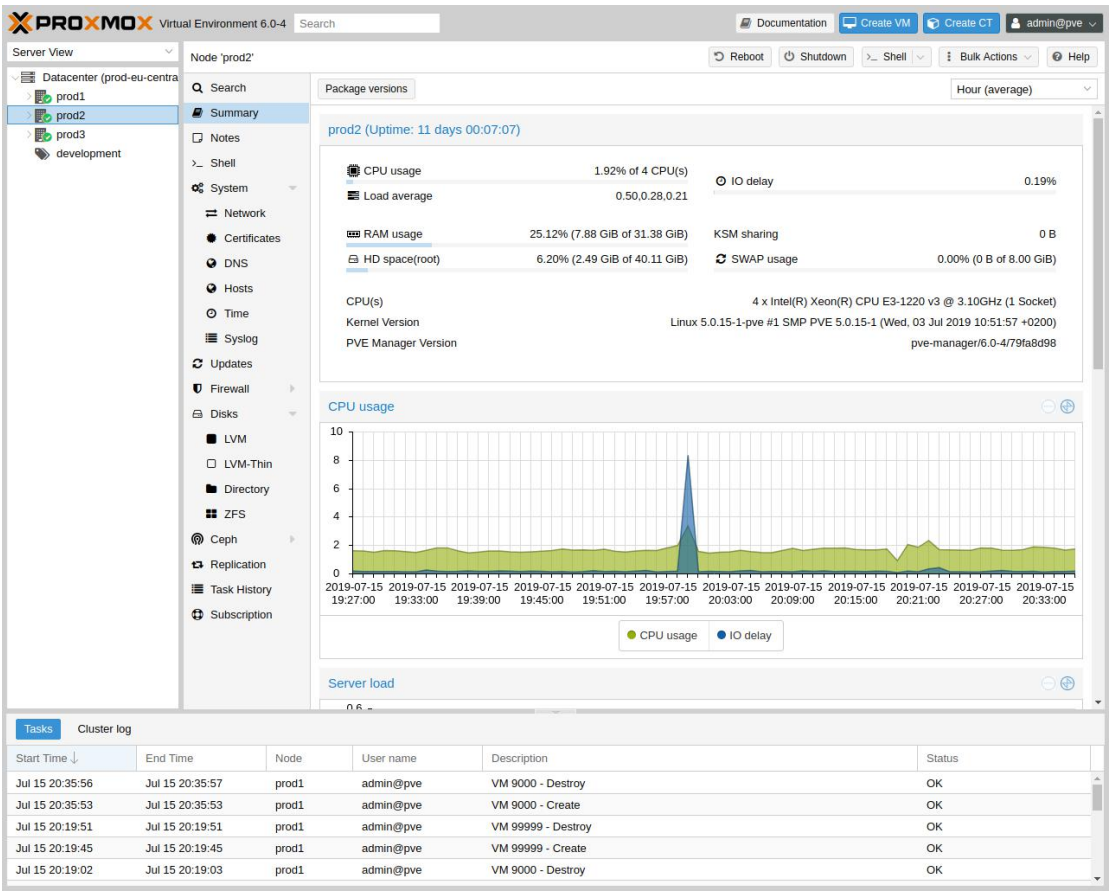
在数据中心级界面，你可以管理集群配置和信息。

- 搜索：用于在集群范围内搜索，可搜索的项目包括服务器节点，虚拟机，容器，存储或资源池。
- 概要：用于展示集群健康状态的概要信息。
- 集群：用于设置允许创建/加入集群，并展示相关信息。
- 选项：用于展示和设置集群范围内的公共参数项目。
- 存储：用于添加、管理、删除存储服务。
- 备份：用于管理调度备份任务。其配置在整个集群范围内均有效，所以你无须关心调度备份的虚拟机和容器具体在哪个节点运行，调度备份都会自动完成备份任务。
- 复制：展示复制任务信息，并创建新的复制任务。
- 权限：用于设置用户和用户组权限，此外还可以在此配置 LDAP、微软活动目录和双因子认证。

- HA：用于管理 Proxmox VE 中虚拟机的高可用性。
- 防火墙：用于设置集群范围内的防火墙策略和策略模板。
- 支持：用于查看你订阅的支持服务信息。

如果需要获得更多信息，可以查看相应章节。

5.4.2 节点

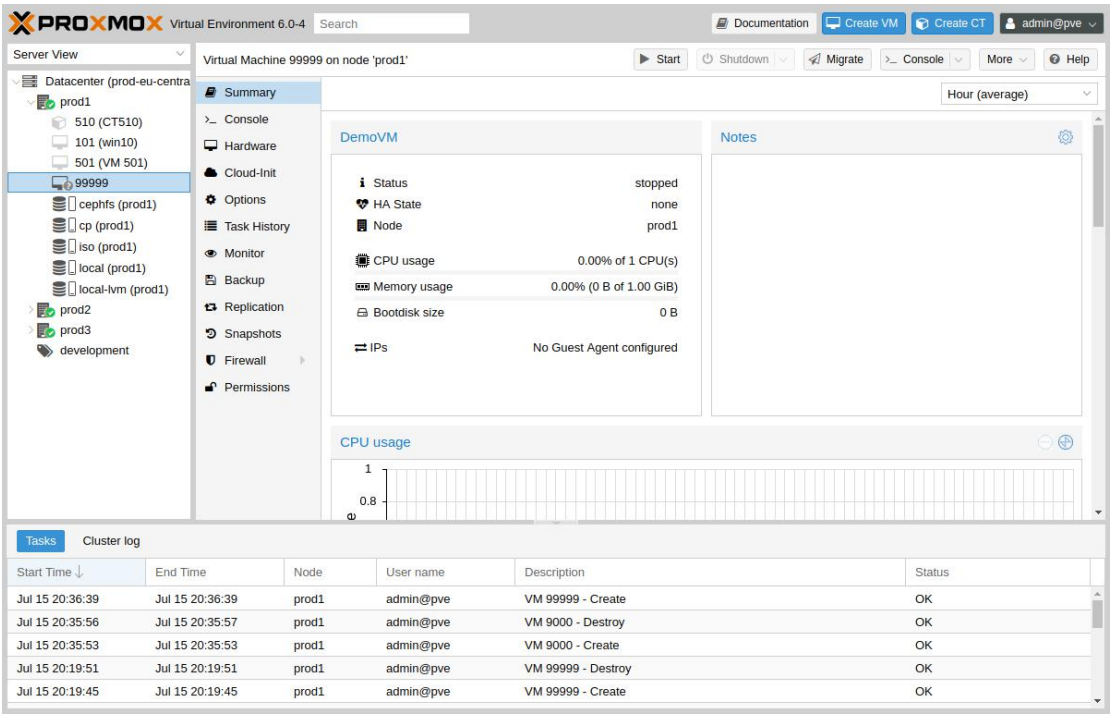


集群内的所有节点在该级别分别拥有独立的管理面板。
顶部是若干按钮，包括“Reboot”，“Shutdown”，“Shell”，“Bulk Actions”和“Help”。Shell 又有 noVNC, SPICE 和 xterm.js 等几种选项。Bulk Actions 包括 Bulk Start, Bulk Stop 和 Bulk Migrate 几个选项。

- 搜索：用于在选中节点内搜索，可搜索的项目包括虚拟机，容器，存储或资源池。
- 概要：用于展示资源使用情况的概要信息。
- 备注：用于记录用户备注信息。
- Shell：提供了一个登录节点服务器的 Shell 界面。

- System：用于配置服务器的网络、DNS、时区，也可查看 syslog 日志。
- 更新：用于查看可用的软件包更新，并升级系统软件。
- 防火墙：用于设置节点的防火墙策略。
- Disks：用于展示服务器硬盘的使用情况和概要信息。
- Ceph：只有在节点服务器安装 Ceph 服务器软件后才可使用。可从此管理 Ceph 集群，并查看 Ceph 运行状态。
- 复制：展示复制任务信息，并创建新的复制任务。
- 任务记录：展示历史任务日志信息。
- 认购：可在这里上传你订阅 Proxmox VE 时获取的密钥，并查看服务器当前的服务支持状态。

5.4.3 客户机

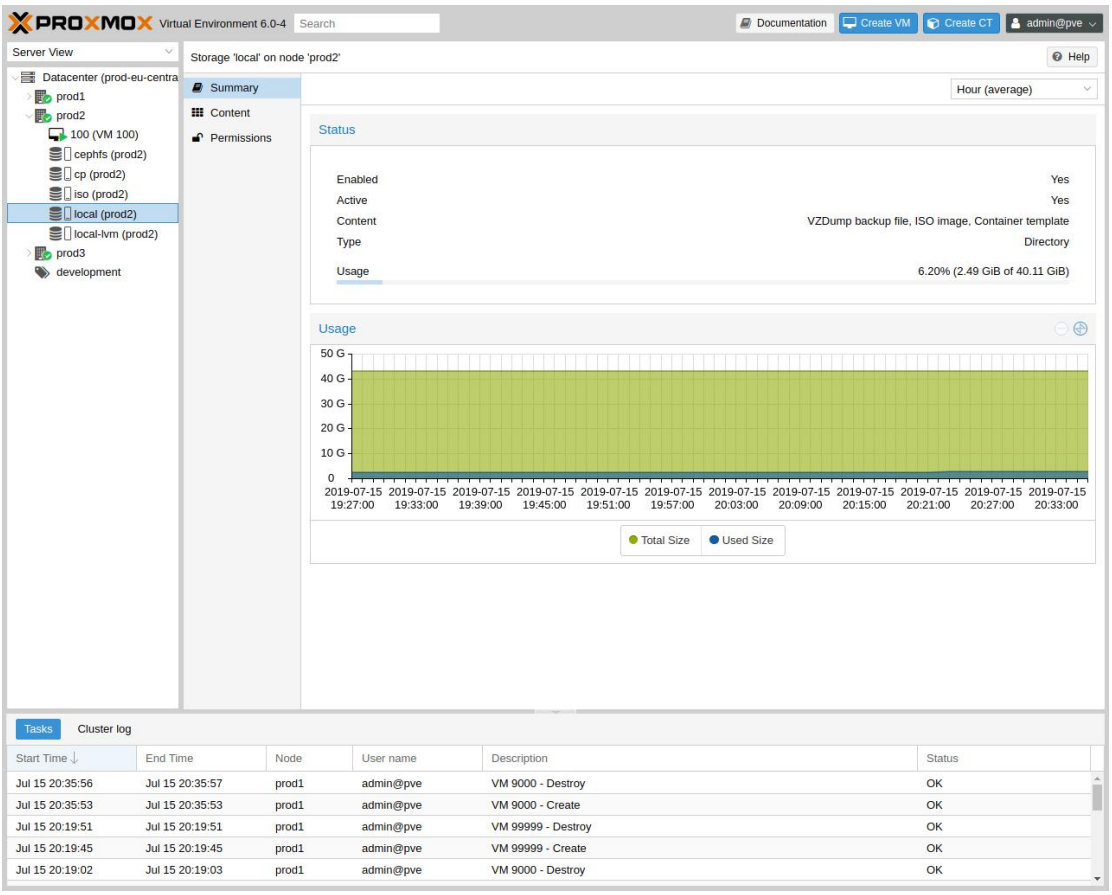


一共有两类客户机，均可以转换为模板。一是 Kernel-based 虚拟机（KVM），二是 Linux 容器（LXC）。资源树中对两类客户机的管理是一致的，只有少数配置参数不一样。
如果在左侧资源树选中虚拟机，中间主管理界面就会显示虚拟机相关信息。
主管理界面上方包含了重要的虚拟机操作命令按钮，如“启动“、”关机“、”重置“、”删除“、”迁移“、”控制台“和”帮助“。其中一些按钮还有隐藏按钮，如”关机“下隐藏有”停止“按钮，”控

制台“包含有几种不同类型控制台，如 SPICE、noVNC 和 xterm.js。
右侧显示内容根据选中的客户机选项而变化。
左侧依次显示所有可以选择的客户机选项。

- 概要：展示虚拟机概要信息。
- 控制台：虚拟机交互控制台界面。
- （KVM）硬件：展示和配置 KVM 虚拟机硬件配置信息。
- （LXC）资源：展示并配置 LXC 容器硬件配置信息。
- （LXC）网络：LXC 容器网络配置信息。
- （LXC）DNS：LXC 容器 DNS 配置信息。
- 选项：提供虚拟机选项配置界面，KVM 和 LXC 分别有自己的配置参数。
- 任务记录：展示虚拟机历史任务日志记录。
- （KVM）监视器：提供和 KVM 进程交互通信的控制界面。
- 备份：展示虚拟机可用备份，也可以创建新的虚拟机备份。
- 复制：展示虚拟机的复制任务，并允许创建新的复制任务。
- 快照：虚拟机快照管理界面。
- 防火墙：虚拟机防火墙管理界面。
- 权限：虚拟机访问权限管理界面。

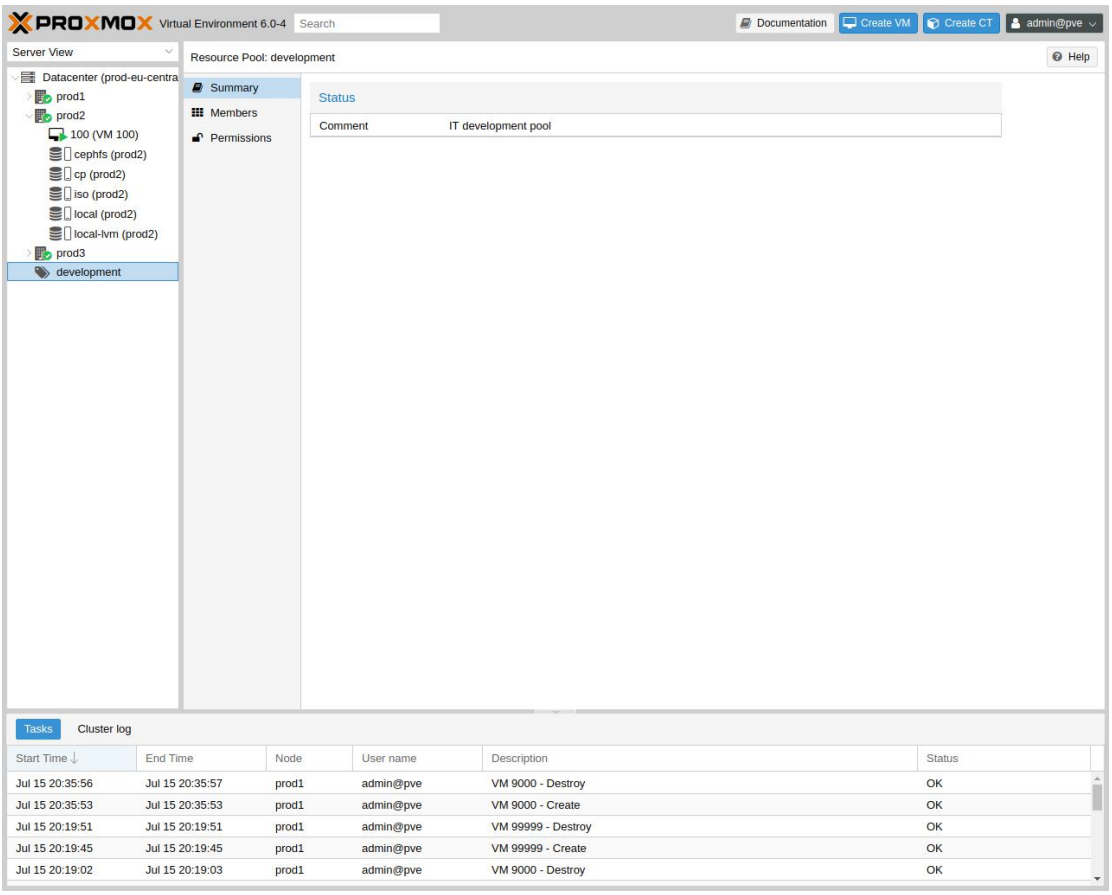
5.4.4 存储



该视图分为两个部分。左侧展示可供选择的存储资源选项，右侧展示选项对应的参数信息。

- 概要　展示存储参数信息，如使用率、类型、数据内容、运行状态、激活状态等。
- 内容　按数据类型分组展示所保存的数据内容。
- 权限　存储资源访问权限管理界面。

5.4.5 资源池



该视图分为两个部分。左侧展示可供选择的逻辑资源池选项，右侧展示选项对应的参数信息。

- 概要　　展示资源池描述信息。
- 内容　　资源池成员展示和管理界面。
- 权限　　资源池访问权限管理界面。

第 6 章 集群管理工具

Proxmox VE 集群管理工具 pvecm 用于创建一个由多个物理服务器节点构成的“组”。这样的一组服务器称为一个“集群”。我们使用 Corosync Cluster Engine 来确保集群通信的稳定可靠，目前一个集群最多可拥有 32 个物理节点（也可以更多，关键在于网络时延）。
使用 pvecm 可以创建新的集群，可以向集群新增节点，可以从集群删除节点，可以查看集群状态信息，也可以完成其他各种集群管理操作。Proxmox VE 集群文件系统（pmxcfs）用于确保配置信息透明地发送到集群中所有节点，并保持一致。
以集群方式使用 Proxmox VE 有以下优势：

- 集中的 web 管理
- 多主集群架构：从任何一个节点都可以管理整个集群
- pmxcfs：以数据库驱动的文件系统保存配置文件，并通过 corosync 在确保所有节点的配置信息实时同步。
- 虚拟机和容器可方便地在物理服务器节点之间迁移。
- 快速部署。
- 基于集群的防火墙和 HA 服务。

6.1 部署要求

- 所有节点必须可以互相访问彼此的 UDP 5404 和 5405 端口，以确保 corosync 正常工作。
- 各节点日期和时间需要保持同步。
- 各节点之间要能够在 TCP 22 端口建立 SSH 通信。
- 如果你需要配置 HA，则最少需要 3 个物理服务器节点，以保证集群多数票机制生效。此外，还需要保证所有节点使用同一版本的 Proxmox VE。
- 我们建议为集群通信分配专用网卡，特别是在配置共享存储的情况下，分配专用网卡能确保集群通信的稳定可靠。

- 新加节点时需要输入 root 口令。

➢ 注意

Proxmox VE 3.x 或更早版本不能和 Proxmox VE 4.x 混合组建集群。

➢ 注意

理论上，可以将 Proxmox VE 4.4 和 Proxmox VE 5.0 混编在同一集群，但不建议在生产环境中这样做。只是在进行 Proxmox VE 集群大版本升级的过程中，允许临时混编。

➢ 注意

Proxmox VE 6.x 不能和更早版本混编在同一集群。Proxmox VE 6.x 的集群通信协议（corosync）较更早版本有彻底的改变。针对 Proxmox VE 5.4 的 corosync 3 软件包仅用于帮助升级到 Proxmox VE 6.0。

6.2 节点服务器准备

首先需要在物理服务器节点安装 Proxmox VE。确保所有节点的主机名和 IP 地址都配置妥当。加入集群后将不允许再修改主机名和 IP 地址。
目前，创建集群操作可以在命令行控制台（ssh 登录）下进行，也可以通过 API 调用完成，GUI 界面就是通过调用 API 来创建集群的（Datacenter→Cluster）。
通常会在/etc/hosts 配置所有节点名称和 IP 地址解析记录（或使用其他的主机名解析技术），但这种配置对集群通信关系不大。对于节点之间互相通过 SSH 访问就显得比较有用，毕竟节点名称要比 IP 地址更容易使用（参见 6.7.3 节连接地址类型）。另外，请注意，在集群配置中，我们通常直接使用 IP 地址来标识节点。

6.3 创建集群

首先通过 ssh 远程登录一个 Proxmox VE 节点。然后为你的集群想一个名字，注意不要和已有的集群重名。一旦建立集群后，将不允许修改集群名称。集群命名规范和节点命名规范是一样的。创建命令如下：
hp1# pvecm create CLUSTERNAME

☒ 警告

集群名称用于计算生成集群多播通信地址。如果在网络里有多个 Proxmox VE 集群在运行，请务必确保集群名称不重复。为便人为混乱，即使不在同一网络，最好也不要重名。

创建集群后，可以用如下命令查看集群状态：
hp1# pvecm status

6.3.1 同一网络内创建多个集群

可以在同一物理网络或逻辑网络内创建锁个集群。每个集群必须使用不同的名字，这不仅有利于帮助管理员明确所操作的集群，也有利于避免集群通信故障。
尽管 corosync 集群通信占用的带宽并不高，但对网络数据包的延迟和每秒数据包吞吐量（PPS）有较高要求。而同一网络中的多个集群将互相争夺网络资源，所以在条件允许的情况下，还是尽量为大规模集群专门配置独立的物理网络设施。

6.4 新增集群节点

通过 ssh 远程登录要加入 Proxmox VE 集群的新节点。执行如下命令。
hp2# pvecm add IP-ADDRESS-CLUSTER
这里 IP-ADDRESS-CLUSTER 可以是已有集群中任意节点的 IP 地址或主机名。推荐使用 IP 地址（详见 6.7.3 节连接地址类型）。

☒ 警告

为避免虚拟机 ID 冲突，Proxmox VE 规定新节点加入集群前不能配置有任何虚拟机。此外，新加入节点/etc/pve 目录下的原有配置信息将被集群配置全部覆盖。如果节点上已有虚拟机，可以首先使用 vzdump 将所有虚拟机备份，然后删除节点上的虚拟机，待加入集群后再用新的虚拟机 ID 恢复原有虚拟机。

加入集群后可以查看集群状态：
pvecm status

加入 4 个节点后的集群状态

```
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
```

如果只需要查看节点列表，可运行如下命令：

```
# pvecm nodes
```

列出集群节点列表

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

6.4.1 添加位于不同网段的节点

如果要添加一个节点，而该集群网络和该节点在不同网段，你需要使用 LINK0 参数来指定节点在集群网络内使用的地址。

```
pvecm add IP-ADDRESS-CLUSTER -link0 LOCAL-IP-ADDRESS-LINK0
```

如果你要使用 kronosnet 的内置冗余环协议（见 6.8 节），你还需要设置 LINK1 参数。

6.5 删除节点

☒ 警告

删除节点前请仔细阅读删除操作步骤，不然很可能会发生你预料不到的情况。

首先将待删除节点上所有虚拟机都迁移到其他节点。确保待删除节点上没有任何你需要保留的数据和备份，或者相关数据已经被妥善备份。以下示例中，我们将演示如何删除集群中的 hp4 节点。
登录一个不同节点（非 hp4），执行 pvecm nodes 命令，确认待删除节点 ID。
hp1# pvecm nodes

Membership information
~~~~~~~~~~~~~~~~~~~~~~

Nodeid	Votes	Name
1	1	hp1 (local)
2	1	hp2
3	1	hp3
4	1	hp4

☒ 重要

如上所述，一定要在删除操作前先将待删除节点关闭，并确保不再启动该节点（特别在当前集群网络中）。如果你重新开启已删除的节点，很可能导致集群崩溃，并且很难恢复到一个正常的集群状态。

关闭 hp4 节点后，可以将其从集群内安全地删除。
hp1# pvecm delnode hp4

如果命令执行成功，将直接返回，而且不会有任何输出。可以运行 pvecm nodes 或者 pvecm status 检查删除节点后的集群状态。你会看到类似如下输出：
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

如果出于某种原因，你需要将被删除节点重新加入原集群，需要按如下步骤操作：

- **格式化被删除节点，并重新安装 Proxmox VE。**
- **如前一节所述步骤，将该节点重新加入集群。**

➢ **注意**

删除节点后，该节点的 SSH 指纹仍将保存在集群其他节点的 known_hosts 中。如果向集群再次添加相同 IP 或主机名的节点，并遭遇 SSH 错误，可以在新加入节点上运行 pvecm updatecerts，以更新其他节点上的 SSH 指纹信息。

6.5.1 隔离节点

☒ **重要**

我们不推荐使用隔离节点操作，按此方法操作时请务必小心。如果你对操作结果存有疑虑，建议使用删除节点的方法。

你可以将一个节点从集群中隔离出去，而无需格式化并重装该节点。但将节点从集群中隔离出去后，被隔离的节点仍然能够访问原 Proxmox VE 集群配置给它的共享存储。你必须在将节点隔离出去之前解决这个问题。由于存储锁只在一个集群内部起作用，所以 Proxmox VE 集群之间不能共享同一个存储设备，否则就有可能发生虚拟机 ID 冲突。
建议为待隔离节点专门创建一个独享的新存储服务。例如，可以为待隔离节点分配一个新的

NFS 服务或者 Ceph 存储池。必须确保该存储服务是独享的。在分配存储之后，可以将该节点的虚拟机迁移到新存储服务，接下来就可以开始进行隔离节点的操作。

☒ 警告

必须确保所有的资源都被已经彻底被隔离。否则将可能发生冲突或其他问题。

首先在待隔离节点上停止 pve-cluster 服务：
systemctl stop pve-cluster
systemctl stop corosync

然后将待隔离节点的集群文件系统设置为本地模式：
pmxcfs –l

接下来删除 corosync 配置文件：
rm /etc/pve/corosync.conf
rm /etc/corosync/*

最后重新启动集群文件系统服务：
killall pmxcfs
systemctl start pve-cluster

到此，该节点已经从集群中被隔离出去了。你可以在原集群中任意节点上执行删除命令：
pvecm delnode oldnode

如果因前面的隔离操作，原集群中剩余节点已经不满足多数票，节点删除命令就会失败。你可以将期望的多数票数量设置为 1，如下：
pvecm expected 1
然后重复节点删除命令即可。

接下来你可以重新登录被隔离出去的节点，删除原集群遗留下的各个配置文件。删除完成后，该节点即可重新加入任意其他集群。
rm /var/lib/corosync/*

被隔离节点的集群文件系统中仍然残留有和原集群其他节点有关的配置文件，这些也是需要删除的。你可以递归删除/etc/pve/nodes/NODENAME 目录清除这些文件。但在执行删除操作前请再三检查，确保删除操作无误。

☒ 警告

原集群中其他节点的 SSH 公钥仍会保留在 authorized_key 文件中。这意味着被隔离节点和原集群节点之间仍然可以用 SSH 公钥互相访问。为避免出现意外情况，可以删除/etc/pve/priv/authorized_keys 文件中的对应公钥。

6.6 多数票

Proxmox VE 采用了基于多数票（quorum）的机制确保集群节点状态一致。

多数票是指在一个分布式系统内一个分布式交易获准执行所必须得到的最低票数。

——Wikipedia 多数票（分布式计算）

在网络可能分裂为多个区域的情况下，修改集群状态需要得到大多数节点在线。如果集群内节点数量不足以构成多数票，集群将自动转为只读状态。

➢ 注意

默认情况下，Proxmox VE 集群内每个节点都有一票的投票权。

6.7 集群网络

集群网络是 Proxmox VE 集群的核心。集群网络必须确保可靠地将集群通信数据包按顺序送达所有节点。Proxmox VE 使用 corosync 来实现集群网络通信，确保集群网络通信的高性能，低延时，高可用。我们的分布式集群文件系统（pmxcfs）就基于此构建。

6.7.1 集群网络配置要求

Proxmox VE 集群网络只有在网络延时低于 2ms 时（局域网内）才可以正常工作。集群网络不应该同时承担其他重载业务，最理想的集群网络是专门设置的独立网络。特别注意不要将 corosync 网络和存储网络共享（除非使用特别低优先级的冗余网络设置，参见 6.8 节）。
最佳实践是在创建集群前先检测网络质量，确保网络能满足集群通信要求。为确保各节点之间能够通过集群网络互相访问，可以使用 ping 工具测试连通性。
如果启用了 Proxmox VE 防火墙，corosync 通信将被自动开启，无需手工设置。

➢ 注意

Corosync 3.0（Proxmox VE 6.0 引入）以前使用多播通信。Corosync 3.0 开始使用 Kronosnet 进行集群通信，目前仅支持 UDP 单播方式通信。

☒ 注意

仍然可以启用多播通信，或原来的单播通信，只需在 corosync.conf 中将传输方式设置为 udp 或 udpu 即可（见 6.10.1 节）。但要注意，这将同时停止通信加密和冗余支持。因此，并不建议这样使用。

6.7.2 独立集群网络

默认情况下，不带任何参数创建集群时，Proxmox VE 集群会和 Web GUI 以及虚拟机共享使用同一个网络。如果你配置不当，存储网络通信流量也有可能会通过集群网络传输。我们建议避免和其他应用共享使用集群网络，因为 corosync 是一个对时延非常敏感的实时应用。

准备一个新的网络

首先，你需要准备一个新的网络端口，该端口应连接在一个独立物理网络上。其次，需要确保这个网络满足 6.7.1 节集群网络配置要求。

创建集群时配置独立网络

可以用带 linkX 参数的 pvecm 命令创建拥有独立网络的 Proxmox VE 集群。
如果你想配置独立网卡用于集群通讯，而该网卡又配置了静态 IP 地址 10.10.10.1/25，那么可以使用如下命令：
pvecm create test --link0 10.10.10.1

然后可以使用如下命令检查集群通信是否正常：
systemctl status corosync
接下来，按 6.4.1 添加位于不同网段的节点一节的步骤继续操作。

创建集群后配置独立网络

即使在你创建集群后，你也可以配置集群改用其他独立网络进行通信，而无须重建整个集群。修改集群通信网络，各节点的 corosync 服务需要逐个重启，以便使用新网络通信，这可能会导致集群短时间处于丧失多数票的状态。
首先确认你了解 6.10.1 节编辑 corosync.conf 文件的方法。然后打开 corosync.conf 文件。配置文件 corosync.conf 的内容示例如下：

```
logging {
        debug: off
        to_syslog: yes
}

nodelist {
        node {
                name: due
                nodeid: 2
                quorum_votes: 1
                ring0_addr: due
        }

        node {
                name: tre
```

```
                nodeid: 3
                quorum_votes: 1
                ring0_addr: tre
        }

        node {
                name: uno
                nodeid: 1
                quorum_votes: 1
                ring0_addr: uno
        }
}

quorum {
        provider: corosync_votequorum
}

totem {
        cluster_name: testcluster
        config_version: 3
        ip_version: ipv4-6
        secauth: on
        version: 2
        interface {
                linknumber: 0
        }
}
```

➢ **注意**

该操作方法也适用于修改其他 ringX_addr 参数值，指定了 corosync 的链接地址，其中“ring”沿用了旧版 corosync 的说法，以保持向后兼容。

首先，如果 node 对象中缺少 name 属性，你需要手工增添该属性。注意 name 属性值必须和节点主机名一致。
然后，你需要将 ring0_addr 属性的值修改为节点在新集群网络内的地址。你可以使用 IP 地址或主机名设置 ring0_addr 属性。如果你使用主机名，必须确保所有的节点都能顺利解析该主机名（详见 6.7.3 节连接地址类型）。
在这里，我们计划将集群通信网络改为 10.10.10.1/25，所以需要相应修改所有的 ring0_addr 属性。

➢ **注意**

参数 ringX_addr 指定了 corosync 的链接地址，其中“ring”沿用了旧版 corosync 的说法，以保持向后兼容。

最后，你需要将 config_version 参数值增加 1。修改后的配置文件内容示例如下：

```
logging {
        debug: off
        to_syslog: yes
}

nodelist {

        node {
                name: due
                nodeid: 2
                quorum_votes: 1
                ring0_addr: 10.10.10.2
        }

        node {
                name: tre
                nodeid: 3
                quorum_votes: 1
                ring0_addr: 10.10.10.3
        }

        node {
                name: uno
                nodeid: 1
                quorum_votes: 1
                ring0_addr: 10.10.10.1
        }
}

quorum {
        provider: corosync_votequorum
}

totem {
        cluster_name: testcluster
        config_version: 4
        ip_version: ipv4-6
        secauth: on
        version: 2
        interface {
                ringnumber: 0
        }
}
```

最后你需要再次检查配置修改是否正确，然后可以根据 6.10.1 编辑 corosync.conf 文件一节的内容，启用新的配置。
修改后的配置可以即时生效，无需重启 corosync 服务。如果修改了其他配置，或 corosync 给出提示信息，可以选择重启 corosync 服务。
在一个节点上执行：
systemctl restart corosync
然后检查集群通信是否正常
systemctl status corosync

如果在所有节点上 corosync 服务都能顺利重启并正常运行，那么所有的节点都将逐个改接入新的集群网络。

6.7.3 Corosync 地址

Corosync 地址（也就是 corosync.conf 中的 ringX_addr 地址）可以用以下两种方式设置：

- **IPv4/v6 地址 建议使用该方式，设置为静态地址可以避免地址被意外修改。**
- **主机名 将通过 get addrinfo 获取 IP 地址，此时 IPv6 将被优先使用。要时刻牢记这一点，特别是将现有集群升级至 IPv6 时要格外注意。**

⊠ 注意

要小心使用主机名。因为主机名对应的 IP 地址很可能会在节点或 corosync 未被修改的情况下意外改变，从而对 corosync 产生意料之外的影响。

如果喜欢使用主机名，建议为 corosync 专门设置一个的固定的主机名，并且确保每一个节点都能正确解析所有节点主机名。
从 Proxmox VE 5.1 开始，节点在加入集群时，配置文件会记录其主机名对应的 IP 地址，而不使用主机名。
更早版本的 Proxmox VE 中，corosync.conf 仍将使用主机名。因此，最好手工改用 IP 地址或专门的固定主机名。

6.8Corosync 冗余性

Corosync 内部集成的 kronosnet，默认支持冗余网络（不在传统 udp/udpu 传输层实现）。可以指定多个 link 地址启用冗余网络功能，也可以在 pvecm 命令中使用--linkX 参数激活（当创建集群或添加新节点时），或者在 corosync.conf 中设置多个 ringX_addr。

☒ 注意

为实现可靠的网络故障转移，每个 link 都需要有自己独立的物理网络连接。

每个 link 都有优先级。可以在 corosync.conf 中设置 knet_link_priority 配置优先级。或者通过 pvecm 命令设置优先级参数，如下：

pvecm create CLUSTERNAME --link0 10.10.10.1,priority=20 --link1 10.20.20.1,priority=15

此时，集群将优先使用优先级数值更低的 link1。

如果未手工配置优先级（或两个 link 的优先级一样），将按 link 序号先后顺序启用，序号较小的将被优先启用。

即使所有 link 同时工作，只有最高优先级的 link 用于 corosync 流量。各 Link 优先级必须一致，不同优先级的 link 相互不能通讯。

由于低优先级 link 只有在高优先级 link 故障时才会被启用，可以为其他任务（虚拟机、存储等）的网络分配设置一个低优先级。这样在网络故障时，也可以尽最大可能确保集群工作，毕竟高延迟的拥堵网络也比完全没有的好。

6.8.1 为现有集群增加冗余 Link

为现有集群添加新的 link，首先要按 6.10.1 节编辑 corosync.conf 文件内容检查配置。

然后，为 nodelist 小节的每个节点添加一个新的 ringX_addr 参数。确保为每个节点添加的参数序号 X 都一样，且不和节点已有参数序号重复。

最后，在 totem 小节添加新的 interface，并将 X 替换为上面的 link 序号。

假定新添加 link 序号为 1，新的配置文件示例如下：

```
logging {
    debug: off
    to_syslog: yes
}
nodelist {
    node {
        name: due
        nodeid: 2
        quorum_votes: 1
        ring0_addr: 10.10.10.2
        ring1_addr: 10.20.20.2
    }
    node {
        name: tre
        nodeid: 3
        quorum_votes: 1
        ring0_addr: 10.10.10.3
        ring1_addr: 10.20.20.3
    }
    node {
```

```
        name: uno
        nodeid: 1
        quorum_votes: 1
        ring0_addr: 10.10.10.1
        ring1_addr: 10.20.20.1
    }
}
quorum {
    provider: corosync_votequorum
}
totem {
    cluster_name: testcluster
    config_version: 4
    ip_version: ipv4-6
    secauth: on
    version: 2
    interface {
        linknumber: 0
    }
    interface {
        linknumber: 1
    }
}
```

无需重启，只要按 6.10.1 节编辑 corosync.conf 文件相关步骤即可激活新增 link。可以用如下命令查看确认 corosync 加载了新添加的 link：

```
journalctl -b -u corosync
```

不妨暂时中断原有 link, 以测试一下新增 link 能够在原有 link 中断时能够无缝接管集群通信，保证集群状态正常。集群状态查看命令如下：

```
pvecm status
```

如果集群状态健康，就可以确认新建 link 工作正常。

6.9 外部 Corosync 投票节点

本节介绍如何为 Proxmox VE 集群部署一个外部投票节点。通过该技术，能够提高集群可靠性，在更多节点故障时，继续保持多数票安全要求，从而保持集群运转。

该技术涉及两个服务：

- 守护进程 qdevice，运行在每个 Proxmox VE 节点。
- 外部投票服务，运行在外部独立服务器。

这样就可以在小规模集群中获得更高可用性（例如 2+1 节点）。

6.9.1QDevice 技术概览

Corosync Quroum Deivce（QDevice）是一个服务进程，运行在集群的每个节点上。该服务会将预配置的投票数传递给集群投票子系统，预配置投票数包含了外部仲裁者提供的投票数。其主要作用是提高集群对故障节点数量的容忍度，允许比标准多数票更多的节点发生故障。外部设备能观测到所有节点的状态，从而可以选择部分节点参与投票。也就是说，只有节点接收到外部第三方投票时，才可以正常参与投票。
目前，支持的第三方仲裁者只有 QDevice Net 一种。该服务通过网络与集群成员相连接，并向这部分成员提供投票。任意时刻，它只能给集群的一部分投票。设计上，该服务支持多集群，且为无状态服务，无需额外配置。新集群可以动态加入，且无需在节点上增加任何配置文件，只需要运行 QDeivce 即可。
外部主机只要安装 corosync-qnetd 软件包，并能通过网络连接到集群即可。Proxmox VE 官方提供了 Debian 版本的软件包，其他 Linux 发行版可以用自己的软件包管理器安装。

➢ 注意

与 corosync 不同，QDeivce 通过 TCP/IP 连接到集群。实际上，该服务可以运行在集群网络之外，网络延迟稍微超过 2ms 也没关系。

6.9.2 部署方式

QDevice 支持偶数节点集群，最推荐在 2 节点集群使用，以真正提高可用性。不鼓励在奇数个节点的集群中使用 QDeivce。主要原因在于该服务对两种不同类型集群提供的投票数不一样。偶数节点集群中，该服务提供投票数为 1，从而可以提高可用性。也就是说，QDevice 自己故障时，就和没有 QDevice 时的情况一样。
对奇数节点集群，QDeivce 提供投票数为（N-1），其中 N 时集群节点数量。这主要是为了防止总票数为偶数时的脑裂风险。此时集群最多允许多数节点故障，只要有一个节点（当然还要包括 QDeivce）正常运行即可。这有两个问题：

- 如果 QNet 服务故障，就不能再有其他节点故障了，否则集群立刻就会失去多数票。例如，15 个节点的集群最多可以允许 7 个节点故障。但是，如果配置了 QDevice，一旦 QDevice 故障，就不允许再有其他节点发生故障。这时，QDevice 本身几乎就是一个单点故障。

- 只要 QDevice 加一个节点正常，即使所有其他节点故障也不会失去多数票，这听起来很美好，但实际上单个节点不可能撑住整个集群的负载，更不用说 ceph 服务会在正常节点数量少于((N-1)/2)时停止服务。

建议认真评估以上问题和潜在影响，然后再决定是否在奇数节点集群使用该技术。

6.9.3 安装 QDevice-Net

建议使用非特权用户运行为 corosync-qdevice 投票的服务进程。Proxmox VE 和 Debian 提供的软件包默认都配置成非特权用户运行。投票服务和集群之间的网络通信必须加密，以确保 QDevice 和 Proxmox VE 的安全集成。
第一步是在外部服务器上安装 corosync-qnetd 软件包，在集群所有节点安装 corosync-qdevice 软件包。
然后，确保所有节点正常在线。
在 Proxmox VE 集群的任意节点运行以下命令，即可轻松安装 QDevice：
pve# pvecm qdevice setup <QDEVICE-IP>
集群的 SSH 密钥将自动复制到 QDevice。中间可能需要输入 SSH 口令。
所有步骤成功完成后，将提示“Done”。可以运行以下命令查看当前状态：
pve# pvecm status
...
Votequorum information
~~~~~~~~~~~~~~~~~~~~~~
Expected votes: 3
Highest expected: 3
Total votes: 3
Quorum: 2
Flags: Quorate Qdevice
Membership information
~~~~~~~~~~~~~~~~~~~~~~
Nodeid Votes Qdevice Name
0x00000001 1 A,V,NMW 192.168.22.180 (local)
0x00000002 1 A,V,NMW 192.168.22.181
0x00000000 1 Qdevice
即可确认 QDevice 安装完成。

6.9.4 常见问题

平局打破

如果出现平局，也就是集群分裂为两个同样大小的部分，两个部分彼此网络中断，但都可以连接到 QDevice。此时 QDevice 将随机选择一个部分，并加入其投票。

潜在负面影响

对于偶数节点数量的集群，QDevice 不存在负面影响。如果 QDevice 故障，集群将正常工作，就和没有 QDevice 一样。

安装 QDevice 后增加/删除节点

在安装 QDevice 后，如果想增加或删除节点，需要先删除 QDevice，然后再增加或删除节点。

如果增加删除节点后，集群节点数量变成偶数，可以再次安装 QDevice。

删除 QDevice

如果前期使用 pvecm 工具增加 QDevice，可以用以下命令删除：
pve# pvecm qdevice remove

6.10 配置 Corosync

/etc/pve/corosync.conf 文件是 Proxmox VE 集群的核心配置文件。该文件控制着集群成员构成和网络通信。可以查看 corosync.conf 的 man 手册，以获取更多信息。
man corosync.conf
在查看节点成员时，你可以用 pvecm 命令。但在手工改变集群配置时，你可以直接编辑该 corosync.conf 配置文件。以下是编辑该文件时的一些最佳实践和提示。

6.10.1 编辑 corosync.conf

编辑配置文件 corosync.conf 并非看起来那么简单。Proxmox VE 服务器上一共有两个 corosync.conf 配置文件，一个是集群文件系统中的/etc/pve/corosync.conf，另一个是本地文件系统中的/etc/corosync/corosync.conf。对集群文件系统中的 corosync.conf 进行编辑，配置变更会自动同步到本地文件系统中的 corosync.conf，但反过来编辑本地文件系统中的 corosync.conf，配置变更不会同步到集群文件系统中的副本。
配置文件 corosync.conf 的内容变化会自动更新到服务配置中。这意味着，配置文件内容的变化会实时生效，并影响 corosync 服务的运行。为了避免编辑过程对 corosync 服务产生意想不到的影响，建议你先复制一个 corosync.conf 的副本，修改该副本而不是直接修改 corosync.conf。
cp /etc/pve/corosync.conf /etc/pve/corosync.conf.new
可以使用你喜欢的编辑器对 corosync.conf 副本进行修改。例如可以使用 Proxmox VE 预装的 nano 或者 vim.tiny。

➢ **注意**

编辑完成后，请记住增加 config_version 的值，否则可能会导致错误。

当编辑完成后，在启用新的配置之前，最好先备份当前配置文件，以便新配置文件启用失败或出现错误时能够回退到原配置。可执行如下命令进行备份：
cp /etc/pve/corosync.conf /etc/pve/corosync.conf.bak

现在可以用新的配置文件覆盖原配置文件，如下：
mv /etc/pve/corosync.conf.new /etc/pve/corosync.conf

启用新配置后，可以用以下命令查看服务状态及配置是否生效
systemctl status corosync

journalctl -b -u corosync

如果新配置未能自动生效，可以尝试重启 corosync 服务，如下：
systemctl restart corosync

如果发现错误，可以尝试按下一节介绍的方法进行排查。

6.10.2 故障排查

错误现象：quorum.expected_votes must be configured

如果 corosync 服务停止运行，而系统日志文件中有如下日志信息时：
[...]
corosync[1647]: [QUORUM] Quorum provider: corosync_votequorum failed to initialize.
corosync[1647]: [SERV] Service engine 'corosync_quorum' failed to load for reason 'configuration error: nodelist or quorum.expected_votes must be configured!'
[...]
原因是你在配置文件中设置的 ringX_addr 主机名不能正常解析。

失去多数票的状态下修改配置文件
如果你需要在一个失去多数票的节点上修改/etc/pve/corosync.conf 文件，你可以执行如下命令：
pvecm expected 1

将期望的多数票数目设置为 1，可以让当前节点重新满足多数票的要求，这样你就可以修改配置，或者是将配置恢复到的正常状态。
在 corosync 服务无法启动时，你还需要采取进一步措施以恢复集群。最好是修改本地文件系统中的配置文件/etc/corosync/corosync.conf，先确保 corosync 服务能正常启动。你需要特别注意，确保所有节点的 corosync.conf 内容都完全一致，避免集群出现脑裂的情况。如果你最终还是无法确认故障原因，可以向 Proxmox 社区寻求帮助。

6.10.3 Corosync 参数说明

ringX_addr

用于设置各个节点在不同的 totem 环内的地址，以用于进行 corosync 集群通信。

6.11 集群冷启动

很显然，当所有节点都断线时，集群是无法达到多数票要求的。例如，机房意外断电后，集群往往就处于这样的状态。

➢ 注意

使用不间断电源（UPS，也称为“后备电池电源”）是防止断电导致集群失去多数票的一个好办法，特别是在你需要实现 HA 效果的时候。

当节点重启启动时，pve-guests 服务会等待该节点加入集群并获得多数票状态。一旦获得多数票，该服务会启动所有设置了 onboot 标识的虚拟机。
因此，当你启动节点时，或者是意外断电后恢复供电时，你会发现一些节点启动速度会比其他节点快。另外要注意的是，在你的集群获得多数票之间，任何虚拟机都无法启动。

6.12 虚拟机迁移

能够把虚拟机从一个节点迁移到其他节点是集群的重要特性。Proxmox VE 提供了一些方法以便你控制虚拟机迁移过程。首先是 datacenter.cfg 提供了一些配置参数，其次是迁移命令行和 API 接口提供了相关控制参数。
客户机是否在线，以及是否使用本地资源（如本地磁盘）对迁移有很大影响。
KVM 虚拟机迁移详细内容参见 10.3 节。
容器迁移详细内容参见 11.9 节。

6.12.1 迁移类型

迁移类型是指迁移过程采用加密（secure）或不加密（insecure）通道传输虚拟机数据。将迁移类型设置为 insecure 后，在迁移过程中虚拟机内存数据将以明文方式传输，这有可能导致虚拟机上的敏感数据泄露（例如口令、密钥）。
因此，我们强烈建议使用安全通道迁移虚拟机，特别在你无法控制整个网络链路并无法保证网络不受窃听时。

➢ 注意

虚拟机磁盘迁移不受该配置影响。目前，虚拟机磁盘总是通过安全通道迁移。

由于数据加密会耗费大量计算资源，所以该虚拟机迁移时经常会选用“不安全”的传输方式，以节约计算资源。较新型的系统采用了硬件方式进行 AES 加密，受此影响较小。但在 10Gb 或更高速的网络中，该参数设置对于性能的影响会十分明显。

6.12.2 专用迁移网络

默认情况下，Proxmox VE 通过集群网络传输虚拟机迁移数据。但这样虚拟机迁移流量很可能会干扰到敏感的集群网络通信，并且集群网络也未必就是服务器节点所能利用的最大带宽网络。

通过设置迁移网络参数，可以选择一个专用网络进行虚拟机迁移。除传输内存数据，还可以在离线迁移时传输虚拟机磁盘镜像数据。
迁移网络参数值为 CIDR 格式的网络地址。这样的好处是，你无须分别设置迁移的源节点和目标节点地址。Proxmox VE 能够根据 CIDR 格式的网络地址自动决定目标节点应使用的 IP 地址。为确保该机制的正常工作，每个节点在迁移网络中都必须被分配一个且仅分配一个 IP 地址。

迁移网络配置示例

假定有一个 3 节点集群，节点之间通过 3 个不同的网络连接。其中一个是公共的互联网，一个是集群通信网络，一个是高带宽的快速网络。这里我们将选择该快速网络作为虚拟机迁移网络。
该集群的网络配置示例如下：

```
iface eth0 inet manual

# public network
auto vmbr0
iface vmbr0 inet static
        address 192.X.Y.57
        netmask 255.255.250.0
        gateway 192.X.Y.1
        bridge_ports eno1
        bridge_stp off
        bridge_fd 0

# cluster network
auto eno2
iface eno2 inet static
        address 10.1.1.1
        netmask 255.255.255.0

# fast network
auto eno3
iface eno3 inet static
        address 10.1.2.1
        netmask 255.255.255.0
```

这里，我们将选择快速网络 10.1.2.0/24 作为虚拟机迁移网络。对于命令行方式的迁移操作，可以使用 migration_network 设置迁移网络：

```
# qm migrate 106 tre --online --migration_network 10.1.2.0/24
```

如需将该快速网络设置为默认的虚拟机迁移网络，可利用/etc/pve/datacenter.cfg 中的 migration 属性进行设置：

```
# use dedicated migration network
migration: secure,network=10.1.2.0/24
```

➢ **注意**

在/etc/pve/datacenter.cfg 中设置虚拟机迁移网络时，必须同时设置迁移类型。

第 7 章 Proxmox 集群文件系统(pmxcfs)

Proxmox 集群文件系统是一个数据库驱动的文件系统，用于保存配置文件，并利用 corosync 在集群节点间实现配置文件的实时同步。我们利用这个文件系统来管理 PVE 的配置文件。该文件系统一方面将所有数据保存在磁盘上的一个数据库文件中，同时在内存中保存了一个拷贝。该设计引入了文件系统总容量的上限，目前该上限为 30MB，但仍然足以保存几千台虚拟机的配置信息。
该文件系统的优点如下：

- 在所有节点间透明地实时同步所有配置文件。
- 强一致性校验，避免虚拟机 ID 冲突。
- 节点失去多数票时自动进入只读状态。
- 自动更新所有节点上的 corosync 集群配置文件。
- 分布式锁机制。

7.1POSIX 兼容性

Pmxcfs 基于 FUSE 技术，其实现类似于 POSIX。但我们仅实现了必须的功能，因此 POSIX 标准中的部分功能并未实现。

- 仅支持普通文件和目录，不支持符号链接。
- 不能重命名非空目录（以便于确保虚拟机 ID 的独一性）。
- 不能修改文件权限（文件权限基于路径确定）。
- O_EXCL 创建不是原子操作（类似老的 NFS）。
- O_TRUNC 创建不是原子操作（FUSE 的限制）。

7.2 文件访问权限

所有的文件和目录都属于 root 用户和 www-data 用户组。只有 root 用户有写权限，www-data 用户组对大部分文件有读权限。以下路径的文件只有 root 有权访问。

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

7.3 技术

我们使用 Corosync 集群引擎实现集群通信，用 SQlite 管理数据库文件。文件系统用 FUSE 实现并运行在操作系统的用户空间。

7.4 文件系统布局

文件系统挂载点为：
/etc/pve

7.4.1 文件

corosync.conf	corosync 集群配置（Proxmox VE 4.x 之前为 cluster.conf）
storage.cfg	Proxmox VE 存储服务配置
datacenter.cfg	Proxmox VE 数据中心配置（键盘布局，代理…）
user.cfg	Proxmox VE 访问控制配置（users/groups/…）
domains.cfg	Proxmox VE 认证域
status.cfg	Proxmox VE 外部指标服务器配置
authkey.pub	票据签发系统的公钥
pve-root-ca.pem	集群 CA 的公共证书
priv/shadow.cfg	口令密文文件

priv/authkey.key	票据签发系统的私钥
priv/pve-root-ca.key	集群 CA 的私钥
nodes/<NAME>/pve-ssl.pem	Web 服务器的公开 SSL 证书(由集群 CA 签发)
nodes/<NAME>/pve-ssl.key	pve-ssl.pem 的私钥
nodes/<NAME>/pveproxy-ssl.pem	Web 服务器的公开 SSL 证书链（可由 pve-ssl.pem 覆盖）
nodes/<NAME>/pveproxy-ssl.key	pveproxy-ssl.pem 的私钥
nodes/<NAME>/qemu-server/<VMID>.conf	KVM 虚拟机的配置文件
nodes/<NAME>/lxc/<VMID>.conf	LXC 容器的配置文件
firewall/cluster.fw	集群级别的防火墙配置
firewall/<NAME>.fw	节点级别的防火墙配置
firewall/<VMID>.fw	虚拟机或容器级别的防火墙配置

7.4.2 符号链接

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

7.4.3 用于调试的特殊状态文件（JSON）

.version	文件版本（用于检测文件内容变更）
.members	集群成员的信息
.vmlist	虚拟机列表

.clusterlog	集群日志（最近 50 条）
.rrd	RRD 数据（最近的条目）

7.4.4 启用/禁用调试

运行如下命令可以启用 syslog 调试信息：
echo "1" >/etc/pve/.debug
运行如下命令可以禁用 syslog 调试信息：
echo "0" >/etc/pve/.debug

7.5 文件系统恢复

如果你的 Proxmox VE 服务器出现故障，例如硬件故障，你可以将 pmxcfs 的数据库文件/var/lib/pve-cluster/config.db 复制到一台新的 Proxmox VE 服务器。在新服务器上（没有配置任何虚拟机或容器），停止 pve-cluster 服务，覆盖 config.db 文件（需要设置权限为 0600），然后修改/etc/hostname 和/etc/hosts 和故障服务器对应文件一致，最后重启新服务器并检查是否恢复正常（不要忘记虚拟机/容器镜像数据）。

7.5.1 删除集群配置

将一个节点从集群中删除之后，推荐的做法是重新安装 Proxmox VE。这样可以确保所有的集群/ssh 密钥和共享配置数据都被彻底清除。
某些情况下，你也许不希望重装而直接将节点恢复到单机模式运行，此时可以参考 5.5.1 节"隔离节点"给出的方法。

7.5.2 从故障节点恢复/迁移虚拟机

对于 nodes/<NAME>/qemu-server（虚拟机）和 nodes/<NAME>/lxc（容器）中的虚拟机配置文件，Proxmox VE 认为<NAME>节点是对应目录下虚拟机的拥有者。这样就可以使用本地锁来防止并发的虚拟机配置文件修改操作，而不是使用代价高昂的分布式集群锁。
但由此导致的一个负作用是，当虚拟机所属的节点停止运行时（例如，意外断电，发生集群隔离事件，…），由于不能获取该节点（已停机）上的本地锁，无法用正常方式将该节点上的虚拟机迁移到其他节点（即使相关虚拟机的磁盘镜像保存在共享存储上）。对于配置使用 HA 的虚拟机而言，则不存在这样的问题，因为 Proxmox VE 的 HA 组件已包含了必要的锁机制（集群锁）和看门狗功能，可以确保相关虚拟机能够从故障节点自动迁移到其他节点运行。
对于未配置使用 HA 的虚拟机而言，如果其磁盘镜像保存在共享存储上（并且未使用其他依赖于故障节点本地资源的配置），可以通过将虚拟机配置文件从/etc/pve 下故障节点对应目

录手工移动到其他正常节点对应目录的方式（从而改变该虚拟机从属的节点），达到将虚拟机从故障节点手工迁移的目的。
例如，为将 ID 为 100 的虚拟机从故障节点 node1 迁移到正常节点 node2，可以使用 root 用户登录集群内任意正常节点，并运行如下命令：
mv /etc/pve/nodes/node1/qemu-server/100.conf /etc/pve/nodes/node2/

☒ 警告

使用以上方法迁移虚拟机之前，必须确保故障节点已经确实关机或者被隔离。否则 Proxmox VE 的锁机制将因为 mv 命令而被破坏，并导致不可预料的结果。

☒ 警告

以上方法无法迁移虚拟磁盘镜像保存在故障节点本地磁盘（或使用故障节点其他本地资源）的虚拟机。此时只能设法恢复故障节点重新加入集群，或利用之前的备份文件恢复虚拟机。

第 8 章 Proxmox VE 存储

Proxmox VE 提供了非常灵活的存储配置模型。 虚拟机镜像既可以保存在一种或多种服务器本地存储上，也可以保存在多种共享存储上，例如 NFS 或 iSCSI（NAS，SAN）。你可以自由地配置多种存储池，想配多少就配多少，完全没有任何限制。事实上，Debian Linux 支持的所有存储技术都可以拿过来用。
使用共享存储保存虚拟机镜像的最大好处就是可以在线迁移虚拟机，只要集群的所有节点都能直接访问虚拟机磁盘镜像，那么就无需关机随意迁移，而且迁移时无需复制虚拟机镜像数据，这样也大大提高了迁移的速度。
Proxmox VE 的存储库（libpve-storage-perl 包）具有非常灵活的插件式设计，并对所有存储技术提供了统一接口。这使 Proxmox VE 能够轻松兼容未来出现的新存储技术。

8.1 存储类型

Proxmox VE 将存储分为两种基本类型：

文件存储

文件存储允许访问全功能（POSIX）文件系统。这类存储方案比块存储（如下）更加灵活，允许保存所有类型的数据。ZFS 大概是目前最先进的文件存储方案，并且完全支持快照和克隆功能。

块存储

可用于存储 raw 格式的虚拟机镜像。但不可用于存储其他文件（ISO，虚拟机备份，…）。大部分较新的块存储方案自带了快照和克隆功能。RADOS 和 GlusterFS 是分布式存储，并将数据分散在多个节点保存。

表 8.1 Proxmox VE 支持的存储类型

名称	PVE 名称	级别	支持共享	支持快照	是否稳定
ZFS(local)	zfspool	文件	否	是	是
目录	dir	文件	否	否 1	是

NFS	nfs	文件	是	否 1	是
CIFS	cifs	文件	是	否 1	是
GlusterFS	glusterfs	文件	是	否 1	是
CephFS	cephfs	文件	是	是	是
LVM	lvm	块	否 2	否	是
LVM-thin	lvmthin	块	否	是	是
iSCSI/kernel	iscsi	块	是	否	是
iSCSI/libiscsi	iscsidir ect	块	是	否	是
Ceph/RBD	rbd	块	是	是	是
ZFS over iSCSI	zfs	块	是	是	是

1：在基于文件系统的存储上，可通过使用 qcow2 格式虚拟磁盘来实现快照。
2：可以在 iSCSI 存储上配置 LVM，从而获得共享 LVM 存储。

8.1.1 薄模式存储

一些存储方案，以及 Qemu 镜像文件 qcow2，支持薄模式（thin provisioning）。在薄模式下，只有虚拟机实际写入的数据才会占用物理存储空间。
例如，你创建了一个带有 32GB 磁盘的虚拟机，安装操作系统后，虚拟机根目录下有 3GB 数据。这时，薄模式存储上虚拟机磁盘仅使用 3GB 空间，而非你在虚拟机内查看磁盘容量时看到的 32GB。通过这种方式，薄模式存储允许你分配远大于当前实际可用存储空间的虚拟磁盘镜像。你可以为你的虚拟机创建很大的虚拟磁盘，当虚拟磁盘占用空间变大时，再向你的存储中增加物理硬盘设备，而无需重新调整虚拟磁盘的容量。

☒ 警告

当薄模式存储空间耗尽时，会造成其上所有虚拟机 IO 错误，进而导致文件系统不一致，甚至数据被破坏。建议不要超配存储空间，或者随时监控剩余空间，避免出错。

8.2 存储配置

Proxmox VE 所配置的存储配置信息全部保存在/etc/pve/storage.cfg 中。鉴于该文件在

/etc/pve 目录下，该文件会自动分发到集群的所有节点，所以所有节点都使用同样的存储配置信息。
共享存储配置信息对于共享存储非常有意义，因为共享存储本身就需要被所有节点访问。但对于本地存储来说也是很有用的，特别是在所有节点都配置了同一类本地存储的时候，尽管每个节点的本地存储都是不同的物理设备，其保存的数据也完全不一样。

8.2.1 存储池

每个存储池都有一个类型<type>，并唯一地被<STORAGE_ID>标识。 存储池的配置示例如下：

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

其中，<type>: <STORAGE_ID>是存储池配置的开始部分，其后是一组属性信息。大部分属性都需要配置参数值，但也有一部分使用默认值。使用默认值的情况下，可以省去<value>值。
以 Proxmox VE 安装后的默认存储配置文件为例，其中包含一个名为 local 的本地存储池，其路径为本地文件系统/var/lib/vz，并默认处于启用状态。Proxmox VE 安装程序也会根据安装时设置的存储类型创建其他存储服务。

默认存储配置文件（/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
```

8.2.2 公共存储服务属性

在 Proxmox VE 中，有一些公共的存储服务属性，在不同类型的存储服务中都有。

- nodes

用于配置能够使用/访问当前存储服务的节点名列表。通过该属性可将存储服务配置为仅能由部分节点访问。

- content

存储服务可用于保存多种不同类型的数据，例如虚拟磁盘镜像，光盘 ISO 镜像，容器模板或容器根文件系统。不是所有存储服务都可以存储所有类型的数据。可通过该属性设置存储服务所要保存的数据类型。可设置的属性值如下：

  - images KVM-Qemu 虚拟机镜像
  - rootdir 容器镜像数据
  - vztmpl 容器模板
  - backup 虚拟机备份文件
  - iso ISO 镜像
  - snippets Snippet 文件，例如客户机 hook 脚本

- shared

用于标示存储服务是共享存储服务。

- disable

设置该属性值可禁用该存储服务。

- maxfiles

用于设置每个虚拟机备份文件最大数量。设为 0 表示不限制备份文件数量。

- format

用于设置默认的虚拟机镜像格式（raw|qcow2|vmdk）。

☒ 警告

建议不要在不同 Proxmox VE 集群之间共享同一存储服务。由于某些存储服务访问操作有排他性，需要通过锁机制来防止并发访问。一个集群内可以通过锁机制防止并发访问，但两个集群之间就没办法禁止并发访问了。

8.3 存储卷

我们专门设计了一套存储空间命名规范。当你从存储池中分配了一块存储空间时，Proxmox VE 将返回一个存储卷标示符。存储卷标示符由多个部分组成，开头是存储服务标识<STORAGE_ID>，其后是冒号，最后是基于存储数据类型命名的卷名称。如下是一些合法卷标识符<VOMUME_ID>的示例：

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

可用如下命令获取<VOLUME_ID>对应的文件系统路径：
pvesm path <VOLUME_ID>

8.3.1 存储卷从属关系

每个 image 类型的存储卷都有一个属主。每个 iamge 类型的存储卷，都属于一个虚拟机或容器。例如存储卷 local:230/example-image.raw 由 230 号虚拟机拥有。大部分后端存储都会把这种从属关系用于编码生成存储卷名称。
当你删除虚拟机或容器时，Proxmox VE 会同时删除其拥有的全部存储卷。

8.4 命令行界面使用方法

建议你熟悉并掌握 Proxmox VE 中存储池和存储卷的概念，但实际应用中，你不一定非要在命令行界面去实践基于这些概念的底层操作。通常情况下，使用虚拟机和容器管理工具分配或删除存储卷更加方便。
尽管如此，Proxmox VE 还是提供了一个名为 pvesm（“Proxmox VE Storage Manager”）的命令行工具，可用于基本的存储服务管理操作。

8.4.1 示例

添加存储池
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>

禁用存储池
pvesm set <STORAGE_ID> --disable 1

启用存储池
pvesm set <STORAGE_ID> --disable 0

修改/设置存储属性
pvesm set <STORAGE_ID> <OPTIONS>
pvesm set <STORAGE_ID> --shared 1
pvesm set local --format qcow2
pvesm set <STORAGE_ID> --content iso

删除存储池。该操作并不删除任何数据，也不断开任何连接或卸载任何文件系统，而仅仅是删除配置文件中相关内容。
pvesm remove <STORAGE_ID>

分配存储卷
pvesm alloc <STORAGE_ID> <VMID> <name> <size> [--format <raw|qcow2>]

在 local 存储中分配 4GB 的存储卷。如果设置<name>为空，系统将自动生成存储卷名称。
pvesm alloc local <VMID> '' 4G

释放存储卷
pvesm free <VOLUME_ID>

☒ 警告

该操作将删除存储卷上的所有数据。

列出存储池状态
pvesm status

列出存储池中的存储卷
pvesm list <STORAGE_ID> [--vmid <VMID>]

列出某个虚拟机拥有的存储卷
pvesm list <STORAGE_ID> --vmid <VMID>

列出 iso 镜像
pvesm list <STORAGE_ID> --iso

列出容器模板
pvesm list <STORAGE_ID> --vztmpl

显示某个存储卷的文件系统路径
pvesm path <VOLUME_ID>

8.5 基于目录的后端存储

存储池类型：dir
Proxmox VE 可以使用本地目录或挂载在本地文件系统的共享存储作为存储服务。目录是文件系统级的存储服务，你可以在目录中保存任何类型的数据，包括虚拟机镜像，容器，模板，ISO 镜像或虚拟机备份文件。

➢ 注意

你可以通过 linux 配置文件/etc/fstab 挂载新增存储设备，然后将相应挂载点定义为目录存储服务，用这种方法就可以使用 Linux 支持的任意类型的文件系统。

Proxmox VE 目对录后端存储的唯一要求是兼容 POSIX 标准。这意味着你不能直接在目录存储服务上创建虚拟机快照，但可以使用 qcow2 格式自带的快照功能为保存在目录后端存储的虚拟机镜像创建快照。

➢ 提示

有些存储服务不支持 O_DIRECT，所以你不能在这些存储服务上配置使用 none 模式的缓存，而需要设置缓存模式为 writeback。

Proxmox VE 会在目录后端存储上自动创建预先定义好的子目录，以便存储不同类型的数据。

表 8.2 目录后端存储子目录

数据类型	子目录
虚拟机镜像	images/<VMID>/
ISO 镜像	template/iso/
容器模板	template/cache/
备份文件	dump/

8.5.1 配置方法

目录后端存储支持全部的公共存储服务属性，此外还支持名为 path 的附加属性，以指定路径。配置 path 属性时需要使用绝对路径。
配置示例（/etc/pve/storage.cfg）

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

以上配置定义了名为 backup 的存储池。该存储池可以用来保存最多 7 个虚拟机备份文件(指每个虚拟机最多 7 个备份）。备份文件的绝对路径为/mnt/backup/dump/…。

8.5.2 文件命名规范

目录后端存储有一套专门设计的虚拟机镜像文件命名规范，文件名格式如下：
vm-<VMID>-<NAME>.<FORMAT>

- <VMID>

镜像文件所属的虚拟机 ID.

- <NAME>

可以是任何不包含空白字符的字符串（ascii）。目录后端存储默认设置为 disk-[N]，其中[N]是一个不重复的整数序号。

- <FORMAT>

标识虚拟机镜像文件格式 （ raw|qcow2|vmdk）。

当你将一个虚拟机转换为虚拟机模板 时，Proxmox VE 会重新命名虚拟机镜像文件，以标明其处于只读状态，并仅供基础镜像或克隆使用。
base-<VMID>-<NAME>.<FORMAT>

➢ 注意

像虚拟机模板这样的基础镜像文件仅供用于克隆生成新的虚拟机。所以确保这类文件的只读属性非常重要。目录后端存储会将基础镜像文件的访问权限修改为 0444，并在文件系统支持的情况下设置不可修改标记（chattr +i）。

8.5.3 存储功能

如上所述，绝大部分文件系统本身不支持快照功能。如果要创建虚拟机快照，只能利用 qcow2 文件格式自带的快照功能。
同理，对于链接克隆操作，目录后端存储服务利用 qcow2 的基础镜像功能实现以链接克隆方式创建新虚拟机。

表 8.3 目录后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像 容器镜像 容器模板 ISO 镜像 虚拟机备份	raw qcow2 vmdk subvol	否	qcow2	qcow2

8.5.4 示例

如下命令用于在 local 存储池上创建一个 4GB 的磁盘镜像：
```
# 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'
```

➢ 注意

虚拟机镜像文件必须按照如前所述的规范进行命名。

如下命令用于查看镜像文件路径：
```
# pvesm path local:100/vm-100-disk10.raw
/var/lib/vz/images/100/vm-100-disk10.raw
```

如下命令用于删除镜像文件：
```
# pvesm free local:100/vm-100-disk10.raw
```

8.6 基于 NFS 的后端存储

存储池类型：nfs
基于 NFS 的后端存储服务实际上建立在目录后端存储之上，其属性和目录后端存储非常相似。其中子目录布局和文件命名规范完全一致。NFS 后端存储的优势在于，你可以通过配置 NFS 服务器参数，实现 NFS 存储服务自动挂载，而无需编辑修改/etc/fstab 文件。NFS 存储服务能够自动检测 NFS 服务器的在线状态，并自动连接 NFS 服务器输出的共享存储服务。

8.6.1 配置方法

NFS 后端存储支持全部的公共存储服务属性，但 shared 标识例外，因为 NFS 后端存储的 shared 属性值总是设置为启用状态。此外，NFS 后端存储还具有以下属性，以便于配置 NFS 服务器：

- server

设置 NFS 服务器的 IP 地址或 DNS 域名。建议直接配置为 IP 地址，以避免 DNS 查询带来的额外延迟——除非你的 DNS 服务器非常强大，或者以本地/etc/hosts 文件方式解析 DNS 域名。

- export

设置 NFS 服务器输出的共享存储路径（可用 pvesm nfsscan 命令扫描查看）。

此外，你还可以通过如下属性设置 NFS 存储挂载点：

- path

NFS 后端存储在 Proxmox VE 服务器上的挂载点(默认为/mnt/pve/<STORAGE_ID>/)。

- options

NFS 挂载选项（可查看 man nfs 获取更多信息）。

配置示例（/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
```

➢ 提示

在 NFS 连接请求超时后，NFS 默认会持续尝试建立连接。这有可能导致 NFS 客户端一侧的意外死机。对于保存只读数据的 NFS 存储，可以考虑使用 soft 选项，以限制尝试连接次数为 3。

8.6.2 存储功能

NFS 本身不支持快照功能，但可利用 qcow2 文件格式的支持进行虚拟机快照和链接克隆。

表 8.4 NFS 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像 容器镜像 容器模板 ISO 镜像 虚拟机备份	raw qcow2 vmdk subvol	是	qcow2	qcow2

8.6.3 示例

可用如下命令列出 NFS 共享路径：

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

8.7 基于 CIFS 的后端存储

存储池类型：cifs
基于 CIFS 的后端存储可用于扩展基于目录的存储，这样就无需再手工配置 CIFS 挂载。该类型存储可直接通过 Proxmox VE API 或 WebUI 添加。服务器心跳检测或共享输出选项等后端存储参数配置也将自动完成配置。

8.7.1 配置方法

CIFS 后端存储支持全部的公共存储服务属性，仅共享标识例外，而共享标识总是启用的。另外，CIFS 还提供以下特有属性：

- server

CIFS 存储服务器 IP 或 DNS 域名。必填项。

➢ 提示

为避免 DNS 域名查询带来的延时，直接配置 IP 地址较好—除非你的 DNS 服务器非常可靠，或者直接用/etc/hosts 文件进行域名解析。

- share

所使用的 CIFS 共享服务（可执行 pvesm cifsscan 或在 WebUI 查看获取可用 cifs 服务）存储服务器 IP 或 DNS 域名。必填项。

- username

CIFS 存储的用户名。可选项，默认为“guest”。

- password

用户口令。可选项。用户口令文件仅有 root 可读（/etc/pve/priv/<STORAGE_ID>.cred）。

- domain

设置 CIFS 存储的用户域（workgroup）。可选项。

- subversion

SMB 协议版本号。可选项。默认为 3。因安全原因，已不再支持配置 SMB1。

- path

本地挂载点。可选项。默认为/mnt/pve/<STORAGE_ID>/。

配置示例（/etc/pve/storage.cfg）

```
cifs: backup
        path  /mnt/pve/backup
        server 10.0.0.11
```

share VMData
content backup
username anna
smbversion 3

8.7.2 存储功能

CIFS 本身不支持快照功能，但可利用 qcow2 文件格式的支持实现虚拟机快照和链接克隆。

表 8.5 CIFS 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像 容器镜像 容器模板 ISO 镜像 虚拟机备份	raw qcow2 vmdk	是	qcow2	qcow2

8.7.3 示例

可用如下命令列出 CIFS 共享：
pvesm cifsscan <server> [--username <username>] [--password]
然后可用如下命令将 CIFS 存储添加到 Proxmox VE 集群：
pvesm add cifs <storagename> --server <server> --share <share>
[--username <username>] [--password]

8.8 基于 GlusterFS 的后端存储

存储池类型：glusterfs
GlusterFS 是一个可水平扩展的网络文件系统。GlusterFS 具有模块化设计，兼容常见硬件等优点，是一种低成本的高可用企业级存储解决方案。GlusterFS 能够支持扩容到数 P 字节容量，并可同时支持数千客户端连接。

➢ **注意**

在遭遇节点/brick 故障时，GlusterFS 会通过 rsync 重新同步数据，而大文件同步往往会需要很长时间，所以 GlusterFS 不适宜用于虚拟机镜像存储。

8.8.1 配置方法

GlusterFS 后端存储支持全部的公共存储服务属性，以及如下的 GlusterFS 特有属性：

- server

GlusterFS 存储服务器 IP 或 DNS 域名。

- server2

GlusterFS 备用存储服务器 IP 或 DNS 域名。

- volume

GlusterFS 卷名称。

- transport

GlusterFS 网络传输协议：tcp，unix 或 rdma。

配置示例（/etc/pve/storage.cfg）

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

8.8.2 文件命名规范

GlusterFS 后端存储的子目录布局和文件命名规范与目录后端存储完全一致。

8.8.3 存储功能

NFS 本身不支持快照功能，但可利用 qcow2 文件格式的支持进行虚拟机快照和链接克隆。

表 8.6 GlusterFS 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像 容器模板 ISO 镜像	raw qcow2 vmdk	是	qcow2	qcow2

虚拟机备份				

8.9 基于本地 ZFS 的后端存储

存储池类型：zfspool
该类型后端存储基于本地 ZFS 存储池（或 ZFS 存储池中的 ZFS 文件系统）建立。

8.9.1 配置方法

ZFS 后端存储支持公共存储服务属性 content、nodes、disable，以及如下的 ZFS 特有属性：

- pool

用于设置所使用的 ZFS 存储池/文件系统名称。所有的 Proxmox VE 存储卷都将在指定的存储池分配。

- blocksize

用于设置 ZFS 数据块大小。

- sparse

用于设置启用 ZFS 的薄存储模式。薄模式下，一个存储卷的大小等于其内部数据所占用的实际空间，而非分配给它的总空间。

配置示例（/etc/pve/storage.cfg）

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

8.9.2 文件命名规范

ZFS 后端存储采用如下虚拟机镜像文件命名规范：

```
vm-<VMID>-<NAME>      // 普通虚拟机镜像
base-<VMID>-<NAME>    // 虚拟机模板（只读）
subvol-<VMID>-<NAME>       // 容器镜像（使用 ZFS 文件系统存储容器）
```

- <VMID>

镜像文件所属的虚拟机 ID.

- <NAME>

可以是任何不包含空白字符的字符串（ascii）。目录后端存储默认设置为 disk-[N]，其中[N]是一个不重复的整数序号。

8.9.3 存储功能

在快照功能和克隆功能方面，ZFS 大概是最强大的后端存储方案。ZFS 后端存储同时支持虚拟机镜像（raw 格式）和容器镜像（subvol 格式）的存储。ZFS 的配置继承自上级存储池，所以你只需配置上级存储池使用默认属性值即可。

表 8.7 ZFS 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像 容器镜像	raw subvol	否	是	是

8.9.4 示例

推荐创建另外的 ZFS 文件系统以存储虚拟机镜像：
zfs create tank/vmdata
如下命令在新建文件系统开启数据压缩功能：
zfs set compression=on tank/vmdata
如下命令用于列出可用的 ZFS 文件系统：
pvesm zfsscan

8.10 基于 LVM 的后端存储

存储池类型：lvm
LVM 是建立在硬盘设备和分区之上的一个轻量级存储层软件。LVM 可将硬盘空间划分为多个小的逻辑卷。LVM 在 Linux 上得到了广泛应用，并大大简化了硬盘管理操作。
另一种方式使用 LVM 管理大的 iSCSI LUN。这样可以轻松实现 iSCSI LUN 的空间分配，否则，在 iSCSI 本身不提供空间分配接口的情况下，这将是一个不可能完成的任务。

8.10.1 配置方法

LVM 后端存储支持公共存储服务属性 content、nodes、disable，以及如下的 LVM 特有属性：

- vgname

用于设置 LVM 的卷组（VG）名称。必须设置为已有卷组的名称。

- base

用于标识基本卷。基本卷必须在访问存储之前就自动激活。该属性常用在远端 iSCSI 服务器

上的 LVM 卷组。

- saferemove

用于标识删除逻辑卷时同步擦除数据。设置该属性后，删除逻辑卷时，LVM 将确保所有数据被物理擦除。

- saferemove_throughput

用于设置擦除数据块大小。（ 即 cstream –t 参数值).

配置示例（/etc/pve/storage.cfg）

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

8.10.2 文件命名规范

LVM 后端存储的命名规范与 ZFS 后端存储基本一致。

vm-<VMID>-<NAME>　　//普通虚拟机镜像

8.10.3 存储功能

LVM 是典型的块存储解决方案，但 LVM 后端存储本身不支持快照和链接克隆功能。更不幸的是，在创建普通 LVM 快照期间，整个卷组的写操作都会受到影响而变得非常低效。
最大的好处是你可以在共享存储上建立 LVM 后端存储服务。例如可以在 iSCSI LUN 上建立 LVM。LVM 后端存储自带 Proxmox VE 集群锁以有效防止并发访问冲突。

➢ 提示

最新的 LVM-thin 后端存储提供了快照和链接克隆功能，但不支持在共享存储上使用。

表 8.8 LVM 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像 容器镜像	raw	可能	否	否

8.10.4 示例

以下命令列出所有可用的 LVM 后端存储：

```
# pvesm lvmscan
```

8.11 基于 LVM-thin 的后端存储

存储池类型：lvm-thin
LVM 是在逻辑卷创建时就按设置的卷容量大小预先分配所需空间。LVM-thin 存储池是在向卷内写入数据时按实际写入数据量大小分配所需空间。LVM-thin 所用的存储空间分配方式允许创建容量远大于物理存储空间的存储卷，因此也称为“薄模式”。
创建和管理 LVM-thin 存储池的命令和 LVM 命令完全一致（参见 man lvmthin）。假定你已经有一个 LVM 卷组 pve，如下命令可以创建一个名为 data 的新 LVM-thin 存储池（容量 100G）：
lvcreate -L 100G -n data pve
lvconvert --type thin-pool pve/data

8.11.1 配置方法

LVM-thin 后端存储支持公共存储服务属性 content、nodes、disable，以及如下的 LVM 特有属性：

- vgname

用于设置 LVM 的卷组（VG）名称。必须设置为已有卷组的名称。

- thinpool

LVM-thin 存储池名称。

配置示例（/etc/pve/storage.cfg）
lvmthin: local-lvm
        thinpool data
        vgname pve
        content rootdir,images

8.11.2 文件命名规范

LVM-thin 后端存储的命名规范与 ZFS 后端存储基本一致。
vm-<VMID>-<NAME>　　//普通虚拟机镜像

8.11.3 存储功能

LVM-thin 属于块存储解决方案，同时支持快照和链接克隆功能。新创建的卷数据默认为全 0。
必须强调，LVM-thin 存储池不能被多个节点同时共享使用，只能用于节点本地存储。

表 8.9 LVM-thin 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像 容器镜像	raw	否	是	是

8.11.4 示例

以下命令列出卷组 pve 上所有可用的 LVM-thin 后端存储：
pvesm lvmthinscan pve

8.12 基于 Open-iSCSI 的后端存储

存储池类型：iscsi
iSCSI 是一种广泛应用于服务器和存储设备之间连接的协议。几乎所有的存储厂商都有兼容 iSCSI 的设备。目前也有多种开源的 iSCSI target 解决方案，例如基于 Debian 系统的 OpenMeidaVault。
你需要先手工安装 open-iscsi 软件包才可以使用基于 Open-iSCSI 的后端存储服务。Proxmox VE 默认不安装该 Debian 软件包。
apt-get install open-iscsi
底层的 iscsi 管理任务可以通过 iscsiadm 命令完成。

8.12.1 配置方法

Open-iSCSI 后端存储支持公共存储服务属性 content、nodes、disable，以及如下的 iSCSI 特有属性：

- portal

用于设置 iSCSI Portal（可设置为 IP 地址或 DNS 域名）。

- target

用于设置 iSCSI target。

配置示例（/etc/pve/storage.cfg）

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

➢ 提示

如果需要在 iSCSI 上创建 LVM 存储服务，最好启用 content none。这样就防止直接在 iSCSI LUN 上创建虚拟机镜像。

8.12.2 文件命名规范

iSCSI 协议本身未定义分配空间或删除数据的接口，而是将这部分工作交由各存储厂商自行实现。一般情况下，iSCSI 上分配的存储卷都以 LUN 序号的形式输出，所以 Proxmox VE 就以 Linux 内核获取的 LUN 信息命名 iSCSI 存储卷。

8.12.3 存储功能

iSCSI 属于块存储解决方案，但并未提供任何管理接口。所以，最佳时间是配置并输出一个很大的 iSCSI LUN，然后再配置创建 LVM 进行管理。你可以使用 Proxmox VE 的 LVM 插件直接管理 iSCSI LUN 的存储空间。

表 8.10 iscsi 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像 none	raw	是	否	否

8.12.4 示例

以下命令用于扫描远端的 iSCSI portal，并列出可用的 target：
pvesm iscsiscan -portal <HOST[:PORT]>

8.13 基于用户空间 iSCSI 的后端存储

存储池类型：iscsidirect
用户空间 iSCSI 和 Open-iSCSI 后端存储功能相近，其主要区别在于使用用户空间库（libiscsi2）实现。
需要强调的是，iscsidirect 未使用内核组件。由于省去了内核空间切换，所以其性能更加优秀，但代价是不能其创建的 iSCSI LUN 上配置使用 LVM，你只能在存储服务器端完成 iSCSI LUN 的划分和管理。

8.13.1 配置方法

用户空间 iSCSI 后端存储的属性和 Open-iSCSI 后端存储完全一致。
iscsidirect: faststore
portal 10.10.10.1
target iqn.2006-01.openfiler.com:tsn.dcb5aaaddd

8.13.2 存储功能

➢ 提示

用户空间 iSCSI 后端存储仅能用于 KVM 虚拟机镜像存储，不能存储容器镜像。

表 8.11 iscsidirect 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像	raw	是	否	否

8.14 基于 Ceph RADOS 块设备的后端存储

存储池类型：rbd
Ceph 是一种同时支持对象存储和文件存储的高性能分布式存储解决方案，其设计兼顾高性能、可靠性、可扩展性。RADOS 块设备是一种功能强大的块级别存储设备，优势如下：

- 薄模式存储
- 存储卷容量可调
- 分布式存储及多副本存储（基于多个 OSD 的条带）
- 支持快照和链接式克隆
- 数据自修复
- 无单点故障
- 容量可扩展至数 E 字节。
- 支持内核空间和用户空间实现

系统用户空间 iSCSI 和 Open-iSCSI 后端存储功能相近，其主要区别在于使用用户空间库（libiscsi2）实现。
需要强调的是，iscsidirect 未使用内核组件。由于省去了内核空间切换，所以其性能更加优秀，但代价是不能其创建的 iSCSI LUN 上配置使用 LVM，你只能在存储服务器端完成 iSCSI LUN 的划分和管理。

➢ 注意

小规模部署场景下，也可以直接在 Proxmox VE 服务器上运行 Ceph 存储服务。近些年服务器的 CPU 和内存配置足以支持同时运行存储服务和虚拟机应用。

8.14.1 配置方法

rbd 后端存储支持公共存储服务属性 content、nodes、disable，以及如下的 rbd 特有属性：

- monhost

用于设置监控服务绑定的 IP 地址。

- pool

用于设置 Ceph 存储池名称。

- username

Ceph 用户 ID。

- krbd

设置强制通过内核模块 krbd 访问 rbd 存储服务。可选。

➢ 注意

容器将自动通过 krbd 访问 rbd，不受该参数设置影响。

配置外部 Ceph 集群示例（/etc/pve/storage.cfg）

```
rbd: ceph-external
        monhost 10.1.1.20 10.1.1.21 10.1.1.22
        pool ceph-external
        content images
        username admin
```

➢ 提示

Ceph 底层管理任务可以使用 rbd 命令完成。

8.14.2 认证方式

如选择使用 cephx 认证方式，需要将密钥文件从外部 Ceph 集群复制到 Proxmox VE 服务器。
首先运行如下命令创建目录/etc/pve/priv/ceph
mkdir /etc/pve/priv/ceph
然后复制密钥文件
scp <cephserver>:/etc/ceph/ceph.client.admin.keyring /etc/pve/priv/ceph/<STORAGE_ID>.keyring

密钥文件名称需要和<STORAGE_ID>一致。注意复制操作需要 root 权限才能完成。
如果 Ceph 就安装在 PVE 集群本地，可使用 pveceph 命令，或在 GUI 操作，将自动完成密钥文件复制过程。

8.14.3 存储功能

rbd 属于块存储解决方案，并支持快照和链接克隆。

表 8.12 rbd 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
虚拟机镜像 容器镜像	raw	是	是	是

8.15 基于 Ceph 文件系统（CephFS）的后端存储

存储池类型：cephfs
CephFS 是一种兼容 POSIX 标准的文件系统，后台使用 Ceph 集群保存数据。CephFS 基于 Ceph 技术，兼具 Ceph 大部分特性，包括冗余性，横向扩展，自我修复和高可用性。

➢ 提示

Proxmox VE 提供 ceph 安装功能，见 4.2 节，能够简便快捷地配置 CephFS。当前主流硬件的 CPU 和内存资源已经足够强大，完全可以同时支持虚拟机和 CephFS 的运行。

如需使用 CephFS 存储插件，需要升级 Ceph 客户端。按 3.1.4 节内容增加 Ceph 软件源。然后运行 apt update 和 apt dist-upgrade，即可升级到最新软件版本。
必须确认没有配置使用其他的 Ceph 软件源，否则安装将失败，节点上的软件包版本也将来自不同软件源，并导致未知后果。

8.15.1 配置方法

CephFS 后端存储支持公共存储服务属性 nodes，disable，content，以及如下的 cephfs 特有属性：

- monhost

用于设置监视器进程地址列表。本参数为可选参数，仅在使用外部 Ceph 存储时需要配置。

- path

用于设置本地挂载点。本参数为可选参数。默认为/mnt/pve/<STORAGE_ID>/。

- username

用于设置 Ceph 用户 ID。本参数为可选参数。仅在用外部 Ceph 存储时需要配置。默认为 admin。

- subdir

用于设置待挂载的 CephFS 子目录。本参数为可选参数。默认为/。

- fuse

用于设置通过 FUSE 访问 CephFS。未启用时默认通过内核客户端访问。本参数为可选参数。默认为 0。

示例：外部 Ceph 集群配置样例（/etc/pve/storage.cfg）

```
cephfs: cephfs-external
    monhost 10.1.1.20 10.1.1.21 10.1.1.22
    path /mnt/pve/cephfs-external
    content backup
    username admin
```

➢ 提示

如未关闭 cephx，请务必记住配置客户端密钥。

8.15.2 认证方式

默认使用 cephx 认证，如需使用该认证方式，需要把外部 Ceph 集群密钥复制到 Proxmox VE 主机。

创建目录/etc/pve/priv/ceph，命令如下：

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

然后复制密钥，命令如下：

```
scp cephfs.secret <proxmox>:/etc/pve/priv/ceph/<STORAGE_ID>.secrett
```

密钥名称必须与<STORAGE_ID>一致。密钥复制操作一般需要提供 root 权限才能完成。文件必须只包含密钥本身。这一点与 rbd 后端密钥不一致，rbd 密钥还包含了[client.userid]小节。

密钥可以从外部 ceph 集群（使用 ceph 管理员用户）提取得到，命令如下。注意使用能够访问集群的真实客户端 ID 替换 userid。关于 ceph 用户管理的进一步信息，可以查看 Ceph 文档。

ceph auth get-key client.userid > cephfs.secret

如果 Ceph 安装在 Proxmox VE 集群本地，也就是通过 pveceph 命令安装，以上步骤会在安装时自动完成。

8.15.3 存储功能

cephfs 属于兼容 POSIX 标准的文件系统，其底层采用 Ceph 集群存储。

表 8.13 cephfs 后端存储功能

数据类型	镜像格式	支持共享	支持快照	支持链接克隆
容器模板 ISO 镜像 虚拟机备份	none	是	是	否

注意：由于快照功能缺乏充分测试，尽管未发现有 bug，但仍不能保证稳定可靠。

第 9 章 存储复制

命令行工具 pvesr 用于管理 Proxmox VE 存储复制框架。存储复制能够提高使用本地存储的客户机的冗余性，同时降低客户机迁移时间。
该工具能够将客户机的虚拟磁盘复制到其他节点，使得客户机数据在其他节点也可以访问，而无需共享存储。存储复制使用快照技术减少网络传输数据量。因此，在首次全量同步后只需传输新的增量数据即可。当节点发生故障时，你的客户机可以在复制节点上启动运行。
复制操作按照配置的时间间隔自动执行。最小复制时间间隔为 1 分钟，最大为 1 周。时间间隔配置采用 systemd 日历事件的子集来实现，具体可以参考 9.2 节“调度格式”。
每个客户机都可以同时复制到多个目标节点，但客户机不能两次同时复制到同一目标节点。可以对每个复制的带宽进行限速，从而防止服务器或存储负载过重。
目前，配置了存储复制的虚拟机还不能进行在线迁移。但离线迁移是肯定没问题的。如果你将虚拟机迁移到复制节点，只要将最后一次同步复制后的增量数据（因此也称为 delta）传输过去即可，从而大大缩短迁移时间。当迁移完成后，复制方向也会在两个节点间自动反转。例如：VM100 当前运行在 nodeA 节点，并被配置复制到 nodeB 节点。当你迁移 VM100 到 nodeB 之后，系统将自动调整复制方向，开始把 VM100 最新状态从 nodeB 再复制到 nodeA 节点。
如果你把虚拟机迁移到一个非复制节点，则需要将全部磁盘数据传输过去。迁移完成后，复制任务将继续把该虚拟机复制到原配置的复制节点。

☒ 重要

利用存储复制功能可以实现客户机高可用，但需要注意以下两点：

- **故障节点恢复在线后，将客户机迁移回去会导致错误。**
- **存储复制可实现故障恢复，但在最后一次同步和故障时间点之间的数据会丢失。**

9.1 支持的存储类型

表 9.1 存储类型

描述	PVE 类型	支持快照	稳定可靠
ZFS（本地）	zfspool	是	是

9.2 调度格式

Proxmox VE 支持多种灵活的复制任务调度方式。各调度方式均基于 systemd time 服务的日历事件格式。[1]
日历事件可以在一个表达式中表示一个或多个时间点。
日历事件格式示例如下：
[day(s)] [[start-time(s)][/repetition-time(s)]]
上面的表达式可以用于定义一组日期，以便调度任务执行。也可以设置一个或多个开始执行的时间，存储复制调度器将根据设置的时间调度复制任务执行。表达式'mon, tue, wed, thu, fri 22'可用于调度任务在每天晚上 10 点执行，该表达式也可以简写为'mon..fri 22'。大部分调度任务都可以用这样直观的方式来配置。

➢ 注意

小时采用 24 小时制表示。

可以使用一个或多个重复时间来进一步简化调度任务配置。这种调度表达式一般由一个开始时间和重复复制任务执行次数组成。例如想在上午 8 点到 9 点间每隔 15 分钟调度复制任务，可以使用：'8:00/15'。
这里你可以注意到，如果未使用小时分隔符（:），则对应的单位就是分钟。如果使用了分隔符，则左侧表示小时，右侧表示分钟。此外，还可以用星号*来匹配所有可能的对应值。更多信息可以参考 9.2.2 节。

9.2.1 配置说明

- **日期**

日期使用英语单词缩写表示：sun，mon，tue，wed，thu，fri 和 sat。你可以使用逗号分隔符连接配置多个日期。也可以用起始日期和终止日期来指定一个日期区间，起始日期和终止日期之间使用".."连接，例如 mon..fri。这些配置方式可以混合使用。如果未配置，默认配置为'*'。

- **时间格式**

时间格式由小时和分钟区间列表组成。小时和分钟之间使用':'分隔。小时和分钟均可以使用列表和区间的方式设置，配置格式与日期配置一致。配置时，小时在前，分种在后，如不需要可以省略小时，此时相当于将小时设置为'*'。小时的合法取值为 0-23，分钟合法值为 0-59。

[1] 更多信息可以执行 man 7 systemd.time 查看

9.2.2 示例

表 9.2 调度配置示例

调度配置字符串	其他表达形式	解释
mon,tue,wed,thu,fri	mon..fri	每个工作日的 0:00 调度
sat,sun	sat..sun	周末的 0:00 调度
mon,wed,fri	—	周一、周三、周五的 0:00 调度
12:05	12:05	每天下午 12:05 调度
*/5	0/5	每 5 分钟调度
mon..wed 30/10	mon,tue,wed 30/10	周一、周二、周三的每个整点后 30 分、40 分、50 分调度
mon..fri 8..17,22:0/15	—	每个工作日的上午 8 点到下午 6 点，晚上 10 点到 11 点间，每隔 15 分钟调度
fri 12..13:5/20	fri 12,13:5/20	周五的 12:05 ,12:25 ,12:45，13:05，13:25，13:45 调度
12,14,16,18,20,22:5	12/2:5	每天的 12:05 到 22:05 之间，每 2 小时调度
*	*/1	每分钟调度（最小间隔）

9.3 错误处理

如果复制任务发成错误，将会被置于错误状态。此时配置的调度任务间隔将被暂时挂起。同时系统将每隔 30 分钟重试失败的复制任务，一旦成功，将恢复原配置的调度任务间隔。

9.3.1 可能发生的故障

这里列出的仅是日常最常遇到的故障，实际中很可能遇到其他原因导致的故障。

- 网络连接中断。
- 复制目标节点空间耗尽。
- 目标节点存储 ID 重复。

➢ 注意

遇到故障时可以查看复制日志来分析故障原因。

9.3.2 虚拟机故障转移

在发生严重错误时，客户机可能会在故障节点卡死，无法继续运行。这时可以将客户机手工迁移到正常节点继续运行。

9.3.3 示例

假定有两个客户机（VM100 和 CT200）在节点 A 运行，并配置为复制到节点 B，且 A 节点发生故障并无法恢复。这时可以将客户机手工迁移到节点 B 运行。

- 通过 ssh 连接到节点 B，或通过 WebUI 打开节点 B 的 Shell 界面。
- 检查集群的投票状态

  #pvecm status

- 如果集群不具备多数票，则务必首先修复集群投票状态。只有在彻底无法修复的状态下才可以考虑用如下命令强制当前节点恢复多数票：

  #pvecm expected 1

☒ 警告

手工调整期望票数后，要尽一切可能避免影响集群状态的操作（如增/删节点、存储、客户机）。实际上，只应该在修复多数票或紧急启动重要客户机时才手工调整期望票数。

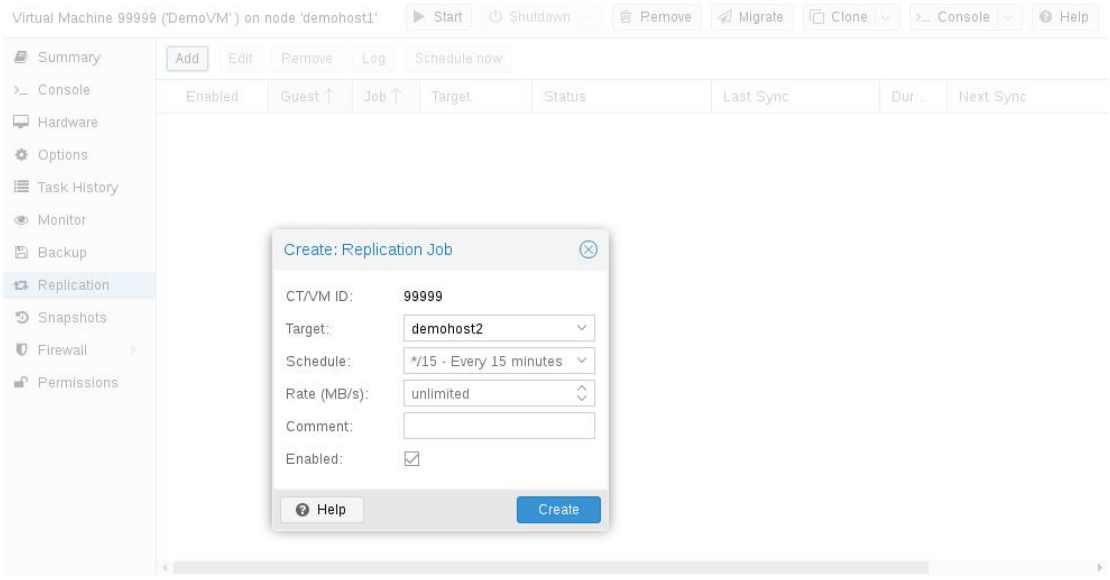
- 将两个客户机的配置文件从节点 A 复移动到节点 B：

```
# mv /etc/pve/nodes/A/qemu-server/100.conf/etc/pve/nodes/B/qemu-server/100.conf
# mv /etc/pve/nodes/A/lxc/200.conf /etc/pve/nodes/B/lxc/200.conf
```

- 启动客户机

```
# qm start 100
# pct start 200
```

9.4 调度任务管理



可以在 Web GUI 上创建、调整或删除复制调度任务。此外，也可以使用命令行（CLI）工具 pvesr 管理调度任务。

在 Web GUI 的各层级（数据中心、节点、虚拟机）都有复制任务管理面板。不同层级的控制面板主要在于所显示的调度任务数量：所有调度任务，当前节点的调度任务或者是特定虚拟机的调度任务。

新增调度任务时，需要指定虚拟机（如果未选定的话）和目标节点。默认调度时间为每隔 15 分钟同步一次 all 15 minutes，也可以根据 9.2 节内容自行设置。还可以对复制任务设置速度上限，从而避免导致存储负载过重。

复制调度任务在集群内有唯一 ID。该 ID 由虚拟机 VMID 和一个任务序号构成。该 ID 由系统自动生成，只有在使用命令行工具时才需要手工指定。

9.5 命令行工具示例

为 ID 100 的虚拟机新增调度任务，每 5 分钟执行一次，复制带宽上限为 10mbps（兆字节每秒）

```
# pvesr create-local-job 100-0 pve1 --schedule "*/5" --rate 10
```

禁用 ID 为 100-0 的调度任务

```
# pvesr disable 100-0
```

恢复被禁用的 ID 为 100-0 的调度任务

```
# pvesr enable 100-0
```

将 ID 为 100-0 的调度任务间隔修改为 1 小时

```
# pvesr update 100-0 --schedule '*/00'
```

第 10 章 Qemu/KVM 虚拟机

Qemu（Qemu 模拟器的简称）是一个开源的虚拟机管理软件，主要功能是模拟物理计算机。在运行 Qemu 的主机看来，Qemu 就是一个普通的用户进程，将主机拥有的硬盘分区、文件、网卡等本地资源虚拟成物理硬件设备并映射给模拟计算机使用。
模拟计算机的操作系统访问这些虚拟硬件时，就好像在访问真正的物理硬件设备一样。例如，当你设置 Qemu 参数向模拟计算机映射一个 ISO 镜像时，模拟计算机的操作系统就会看到一个插在 CD 驱动器里的 CDROM 光盘。
Qemu 能够模拟包括从 ARM 到 sparc 在内的一大批硬件设备，但 Proxmox VE 仅仅使用了其中的 32 位和 64 位 PC 平台模拟硬件，而这也是当前绝大部分服务器所使用的硬件环境。此外，借助 CPU 的虚拟化扩展功能，Qemu 模拟相同架构硬件环境的速度可以被大大提高，虚拟 PC 硬件也是当前 Qemu 支持的运行速度最快的虚拟硬件环境。

➢ 注意

后续章节你可能会看到 KVM（Kernel-based Virtual Machine）一词。这是指 Qemu 借助 Linux 的 kvm 内核模块在 CPU 虚拟化扩展的支持下运行。在 Proxmox VE 里，Qemu 和 KVM 这两个词完全可以互换使用，因为 Qemu 总是尝试使用 kvm 模块。

为方便访问块存储设备和 PCI 硬件，在 Proxmox VE 里 Qemu 进程总是以 root 权限运行。

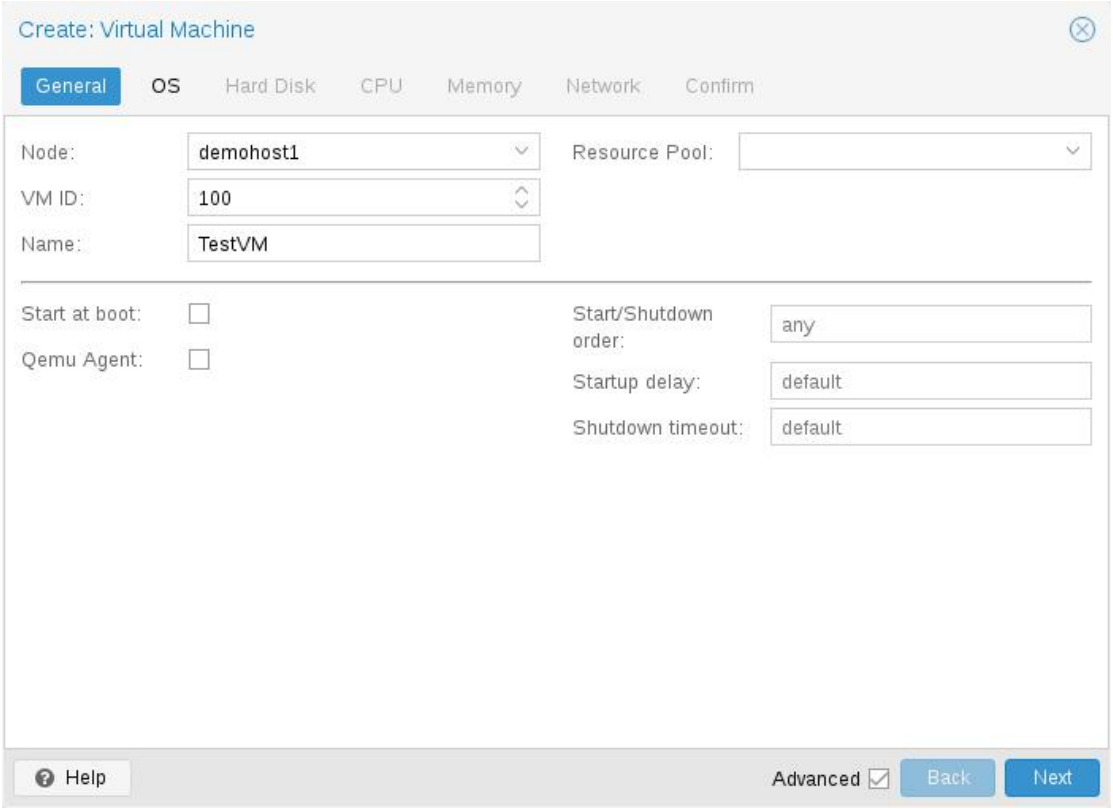
10.1 虚拟化硬件和半虚拟化硬件

Qemu 模拟的 PC 硬件设备包括主板、网卡控制器、scsi 控制器、ide 控制器、sata 控制器、串口等（完整列表参见 man kvm(1)手册），这些都是以软件模拟方式实现的虚拟化硬件。换句话说，这些虚拟化硬件都是和对应硬件设备完全相当的软件，如果客户机操作系统安装了对应的驱动程序，客户机就可以像驱动真实物理硬件一样驱动这些虚拟化硬件。这样，Qemu 就可以直接运行客户机而无需修改客户机操作系统。
但这种方式的缺点就是性能损耗较大，因为 CPU 必须耗费大量计算能力才能以软件方式模拟硬件操作。为提高性能，可以 Qemu 还提供有半虚拟化硬件，这时客户机操作系统会感知到 Qemu 环境的存在，并直接和虚拟机管理器配合工作。
Qemu 的半虚拟化硬件采用了 virtio 标准，并以 virtio 半虚拟化硬件形式实现，具体包括半虚拟化硬盘控制器，半虚拟化网卡，半虚拟化串口，半虚拟化 SCSI 控制器等。
鉴于其所提供的高性能，我们强烈推荐优先使用 virtio 半虚拟硬件。在使用 bonnie++(8)进行的连续写测试中，virtio 半虚拟磁盘控制器的性能是模拟 IDE 控制器的 2 倍。而在基于 iperf 的测试中，virtio 半虚拟网卡的性能是模拟 Intel E1000 虚拟网卡的 3 倍。

10.2 虚拟机配置

一般来说，Proxmox VE 默认提供的虚拟机硬件配置就是最佳选择。当你确实需要改变 Proxmox VE 默认的虚拟机配置时，确保你确实清楚修改的原因及后果，否则可能会导致性能下降或者数据丢失风险。

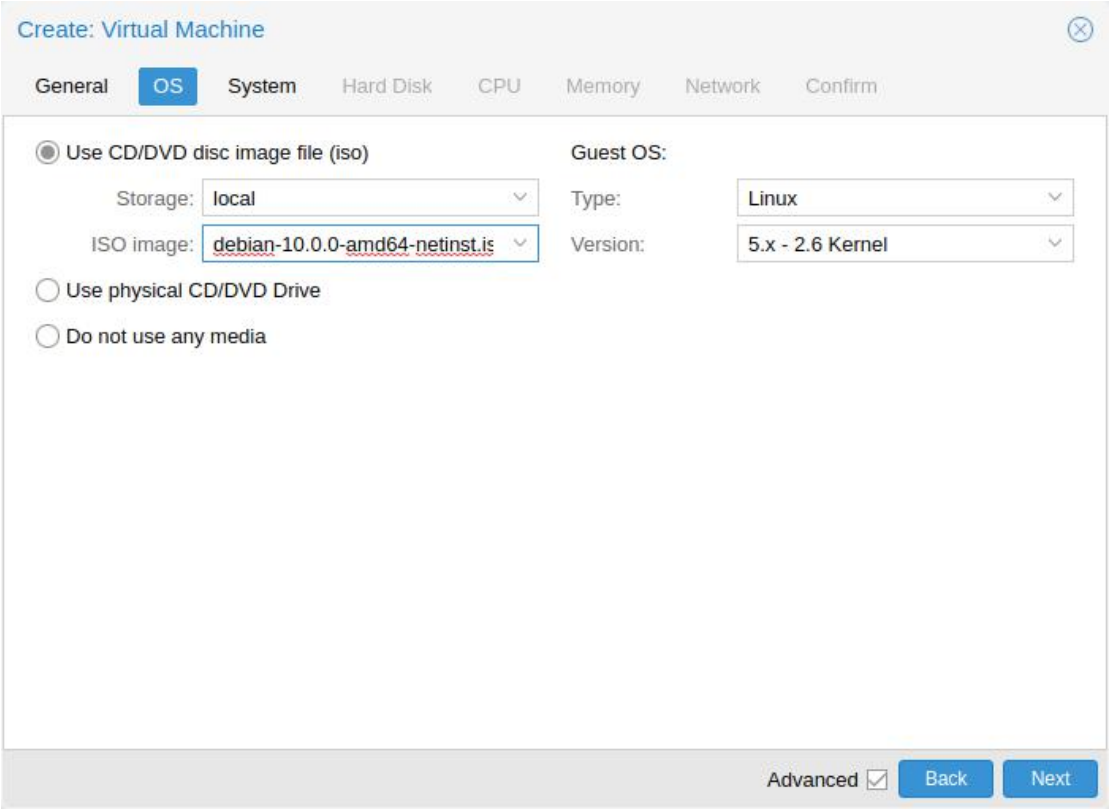
10.2.1 通用配置



虚拟机通用配置包括：

- 节点：虚拟机所处的物理服务器名。
- VM ID：Proxmox VE 用于标识虚拟机的一个唯一编号。
- 名称：虚拟机名称，用于描述虚拟机的字符串。
- 资源池：虚拟机所处的逻辑组。

10.2.2 操作系统配置

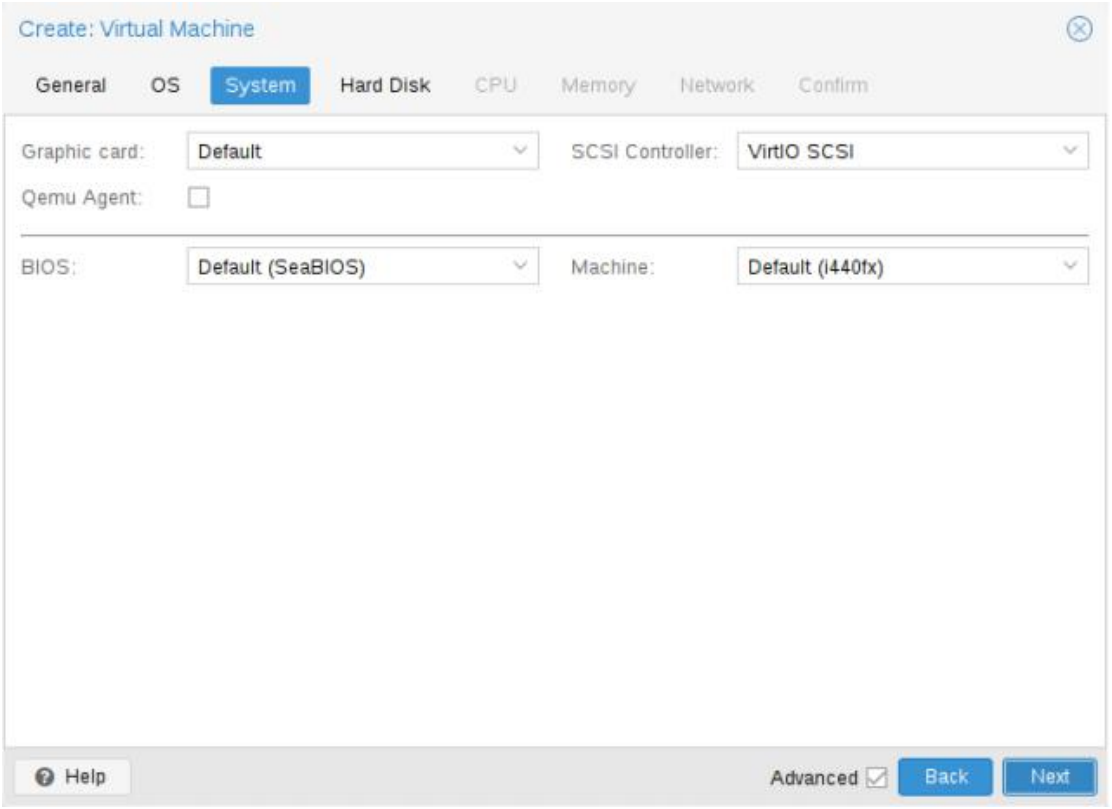


在创建虚拟机时，设置合适的操作系统版本能够帮助 Proxmox VE 优化虚拟机底层配置。例如，Windows 操作系统将期望 BIOS 时钟基于本地时间，而 Unix 类操作系统将期望 BIOS 时钟使用 UTC 时间。

10.2.3 系统设置

创建虚拟机时，可以修改虚拟机的部分系统配置。比如可以指定 10.2.8 节所述的显示类型。此外，还可以改变 10.2.4 节所述 SCSI 控制器类型。如过计划安装 QEMU Guset Agent，或选择使用 ISO 镜像自动安装系统，你可以勾选 Qemu Agent 复选框，以便 Proxmox VE 显示更多信息或自动完成某些操作（例如，关机或快照）。

Proxmox VE 支持多种 BIOS 固件和机器类型，见 10.2.10 节关于 SeaBIOS 和 OVMF 相关内容。多数情况下，你可能希望使用 OVMF 而非传统 SeaBIOS。除非你想使用 10.9 节所述的 PCI 直通技术。机器类型决定了虚拟机主板的硬件布局，具体有 Intel 440FX 和 Q35 两种可选。Q35 提供了虚拟 PCIe 总线，是进行 PCIe 直通的必备之选。



10.2.4 硬盘

Qemu 能够模拟多种存储控制器：

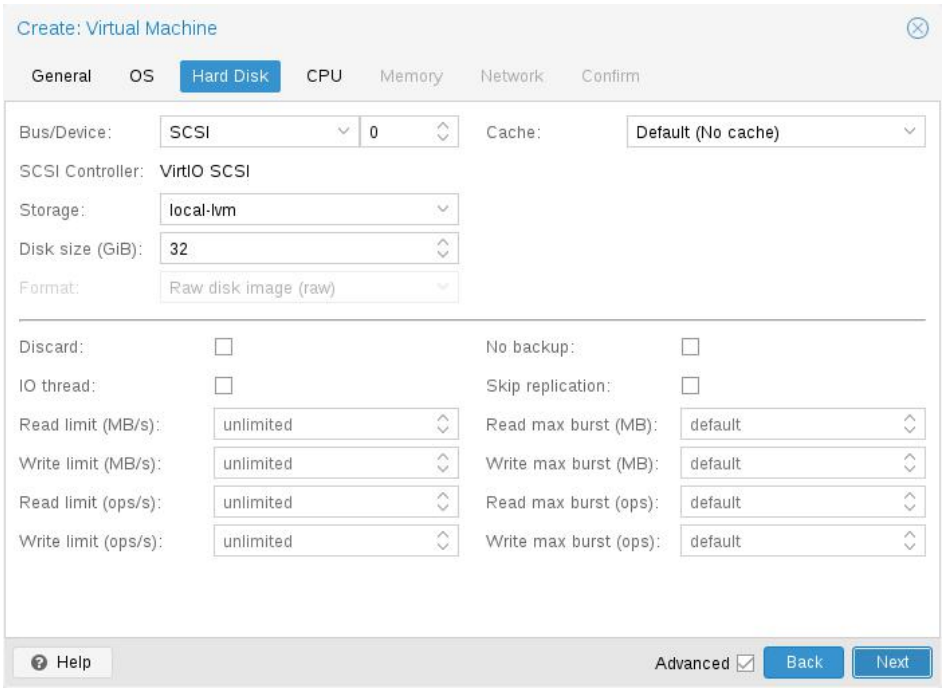
- IDE 控制器，最早可追溯到 1984 年的 PC/AT 硬盘控制器。尽管后来又出现了更多更新的控制器设计，但基本上你能想到的每种操作系统都会支持 IDE 控制器。当你的虚拟机使用 2003 年以前开发的操作系统时，使用 IDE 控制器将是最佳选择。该控制器上最多可挂载 4 个设备。

- SATA 控制器出现于 2003 年，采用了更为现代化的设计，不仅提供了更高的数据传输速率，并且支持挂载更多的设备。该控制器上最多可挂载 6 个设备。

- SCSI 控制器设计于 1985 年，通常用于服务器级硬件，最多可挂载 14 个设备。默认情况下 Proxmox VE 模拟的 SCSI 控制器型号为 LSI 53C895A。

如果你想追求更高的虚拟硬盘性能，可以选择使用 VirtIO SCSI 类型的 SCSI 控制器。事实上，Proxmox VE 4.3 开始将该类型 SCSI 控制器用于 Linux 虚拟机的默认配置。Linux 于 2012 年开始支持该控制器，而 FreeBSD 则于 2014 年开始支持。对于 Windows 类操作系统，你需要在安装操作系统时使用专门的驱动光盘安装驱动程序后才可以使用。如果你想追求最极致的性能，可以选用 VirtIO SCSI single，并启用 IO Thread 选项。在选用 VirtIO SCSI single 时，Qemu 将为每个虚拟磁盘创建一个专用控制器，而不是让所有磁盘共享一个控制器。

- Virtio Block 控制器，也常称为 VirtIO 或 virtio-blk 控制器，是一种老式的半虚拟化控制器。现在已经被 Virtio SCSI 控制器取代。



以上每种控制器都支持同时挂载多个虚拟硬盘设备，虚拟硬盘可以基于一个文件，也可以基于某种存储服务提供的块存储设备。而所选择的存储服务类型决定了虚拟硬盘镜像能采用的数据格式。块存储服务（LVM，ZFS，Ceph）上只能保存 raw 格式虚拟硬盘，文件系统存储服务（Ext4，NFS，CIFS，GlusterFS）则允许你选择使用 raw 格式或 QEMU 镜像格式。

- QEMU 镜像格式是一种基于“写时复制”的虚拟硬盘格式，支持虚拟硬盘快照和薄模式存储。

- Raw 格式硬盘镜像是一种逐个 bit 存储数据的硬盘镜像格式，具体和你在 Linux 上用 dd 命令创建的镜像格式很像。这种镜像格式本身不具有创建快照或薄模式存储的功能，而需要下层存储服务支持才可以实现这些功能。但是其速度可能比 QEMU 镜像格式快 10%。

- VMware 镜像格式仅供用于从其他类型虚拟机系统导入/导出硬盘镜像时使用。

虚拟硬盘的 Cache 模式设置会影响 Proxmox VE 主机系统向虚拟机操作系统返回数据块写操作完成通知的时机。设置为 No cache 是指在所有数据块都已写入物理存储设备写队列后，再向虚拟机发出写操作完成通知，而忽略主机页缓存机制。该方式将能较好地平衡数据安全性和写入性能。

如果你希望在备份某个虚拟机时指定 Proxmox VE 备份管理器跳过某个虚拟硬盘，可以在该虚拟硬盘上启用 No backup 配置。

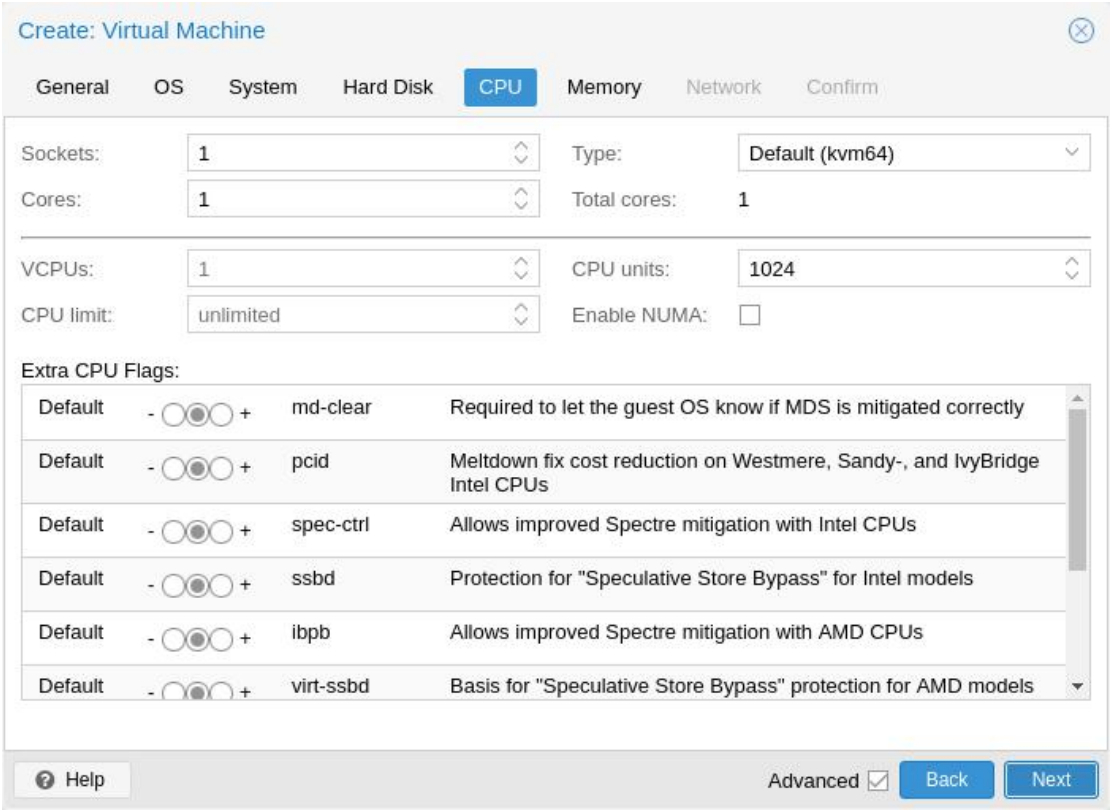
如果你在配置 Proxmox VE 存储复制时希望忽略某些磁盘，可以在该磁盘上启用忽略复制（Skip replication）选项。对于 Proxmox VE 5.0，存储复制功能只能用于 zfspool 上的虚拟

磁盘，所以在其他类型存储上为配置了复制任务的虚拟机新增虚拟磁盘时，需要启用忽略复制。
如果存储服务支持薄模式存储（参见存储服务一章），可以启用 Discard 配置。启用 Discard 配置后，并且虚拟机操作系统支持 TRIM 功能，当在虚拟机中删除文件后，虚拟机文件系统会将对应磁盘扇区标识为未使用，磁盘控制器会根据该信息压缩磁盘镜像。为了支持虚拟机发出的 TRIM 命令，必须使用 VirtIO SCSI 控制器（或 VirtIO SCSI Single），或者在虚拟机磁盘上设置启用 SSD emulation 选项。注意，Discard 参数在 VirtIO Block 设备商是不能生效的。如果希望虚拟机磁盘表现为固态硬盘而非传统磁盘，可以在相应虚拟磁盘上设置 SSD emulation。该参数并不需要底层真的使用 SSD 盘，任何类型物理介质均可使用该参数。但在 SSD emulation 在 VirtIO Block 设备上是不能生效的。

IO Thread

当使用 VirtIO SCSI single 控制器时，对于启用 Virtio 控制器或 Virtio SCSI 控制器时的磁盘可以启用 IO Thread。启用 IO Thread 后，Qemu 将为每一个虚拟硬盘分配一个读写线程，与之前所有虚拟硬盘共享一个线程相比，能大大提高多硬盘虚拟机的性能。注意，IO Thread 配置并不能提高虚拟机备份的速度。

10.2.5 CPU



CPU Socket 指 PC 主板上的 CPU 芯片插槽。每个 CPU 可以有一个或多个核心（core），每个核心都是一个独立的处理单元。为虚拟机配置 1 个 4 核心虚拟 CPU 和配置 2 个 2 核心 CPU

在性能上区别不大。但某些软件是基于 Socket 授权，这时按照软件授权设置 Socket 数量就显得比较有意义了。
通常增加虚拟机的虚拟 CPU 数量都可以改善性能，但最终改善程度还依赖于虚拟机对 CPU 的使用方式。每增加 1 个虚拟 CPU，Qemu 都会在 Proxmox VE 主机上增加一个处理线程，从而改善多线程应用的性能。如果你不确定虚拟机的具体负载，可以先为虚拟机配置 2 个虚拟 CPU，通常情况下这是比较安全的配置方法。

☒ 注意

在 Proxmox VE 主机上可以为多个虚拟机分配总数超过物理 CPU 核心数量的虚拟 CPU（比如在 8 核心主机上同时运行 4 台 4 核心的虚拟机）。Proxmox VE 主机会自动平衡各物理 CPU 核心上的 Qemu 处理线程数量，其效果就和调度一般的多线程应用程序一样。但是，Proxmox VE 不允许为一台虚拟机分配超过物理 CPU 核心数的虚拟 CPU，这只会白白增加不必要的任务调度，反而降低性能。

资源限制

除了可以设置虚拟 CPU 数量，还可以设置一个虚拟机能够占用的物理 CPU 时间比例，以及相对其他虚拟机占用 CPU 时间的比例。通过设置 cpulimit（“主机 CPU 时间“）参数，可以限制虚拟机能占用的主机 CPU 时间。该参数是一个浮点数，1.0 表示占用 100%,2.5 表示占用 250%并以此类推。如果单进程充分利用一个 CPU 核心，就是达到 100%的 CPU 时间占有率。对有 4 个虚拟 CPU 的虚拟机，在充分利用所有核心的情况下，可以达到的最大理论值为 400%。由于 Qemu 还为虚拟外部设备启用其他线程，因此虚拟机真实的 CPU 占有率会更高一些。这个设置对于有多个虚拟 vCPU 的虚拟机最有用，因为可以有效避免同时运行多个进程的虚拟机 vCPU 利用率全部达到 100%。举个极端的例子：对于有 8 个 vCPU 的虚拟机，任何时候都不能让其 8 个核心同时全速运行，因为这样会让服务器负载过大，导致服务器上其他虚拟机和容器无法正常运行。这时，可以设置 cpulimit 为 4.0（=400%）。这时，所有核心同时运行重载任务时，最多占有为服务器 CPU 核心 50%时间资源。但是，如果只有 4 个核心运行重载任务，仍然有可能导致 4 个物理 CPU 核心利用率达到 100%。

➢ 注意

根据具体设置，虚拟机有可能启动其他线程，例如处理网络通信、IO 操作、在线迁移等。因此，虚拟机的实际占用的 CPU 时间会比虚拟 CPU 所占用的要多。为确保虚拟机占用的 CPU 时间不超过所分配给虚拟 CPU，可以设置 cpulimit 为所有核心数量总数。

第二个 CPU 资源限制参数是 cpuunits（常称为 CPU 份额或 CPU 权重），可用于控制虚拟机占用 CPU 资源相对其他虚拟机的比例。这是一个相对的份额权重，默认值为 1024，增加某个虚拟机的 cpuunits，将导致调度器调低其他虚拟机的 CPU 分配权重。例如，虚拟机 VM 100 权重为默认值 1024，虚拟机 VM 200 权重调整为 2048 后，分配给 VM 200 的 CPU 时间将是 VM 100 的两倍。
更多信息可查看 man systemd.resource-control，文档中的 CPUQuota 对应于 cpulimit，CUPShares 对应于 cpuunits。Notes 小节中有具体的参考文档和实现细节。

CPU 类型

Qemu 可以模拟包括从 486 到最新 Xeon 处理器在内的多种 CPU 硬件。模拟更新的 CPU 意味着模拟更多功能特性，比如硬件 3D 渲染，随机数生成器，内存保护等等。通常，你应该选择和主机 CPU 最接近的虚拟机 CPU 类型，这可以让你的虚拟机访问使用主机 CPU 的功能特性（也称为 CPU flags），你也可以将 CPU 类型设置为 host，这样虚拟机的虚拟 CPU 就和主机物理 CPU 完全一致。
这种配置方法最大的问题在于，如果你需要将一个虚拟机在线迁移到另一台物理服务器，虚拟机可能会因为两台物理服务器的 CPU 类型不同而崩溃。如果 CPU flag 不一致，Qemu 进程会直接停止运行。为避免该问题，Qemu 专门提供了一种名为 kvm64 的虚拟 CPU，这也是 Proxmox VE 默认使用的 CPU。大致上 kvm64 是一种类似于 Pentium 4 的虚拟 CPU，具有较少的 CPU flag，但具有最好的兼容性。
简而言之，如果你需要确保虚拟机的在线迁移能力，最好使用默认的 kvm64 虚拟 CPU。如果不在乎在线迁移，或者集群内所有节点硬件型号完全一样，可以设置虚拟 CPU 类型为 host，以获得最好的性能。

Meltdown / Spectre 相关 CPU 标识

有几个 CPU 标识与 Meltdown 和 Spectre 脆弱性相关，除非虚拟机的 CPU 类型已经默认启用，否则需要进行手工设置以确保安全。
启用这两个 CPU 标识，需要满足以下先决条件：

- **主机 CPU 必须支持相关特性，并传递给客户虚拟机的虚拟 CPU。**
- **客户虚拟机操作系统已升级到最新版本，能够利用这两个标识缓解攻击。**

否则，需要先在 Web GUI 调整虚拟 CPU 类型或修改虚拟机配置文件中的 cpu 选项 flag 属性，确保虚拟 CPU 支持相关 CPU 标识。
对于 Spectre v1，v2，v4 补丁，还需要升级从 CPU 制造商下载并升级 CPU 微码。
可以用 root 权限执行以下命令，检测你的 Proxmox VE 服务器是否存在漏洞：

```
/sys/devices/system/cpu/vulnerabilities/*; do echo "${f##*/} -" $( ←-
    cat "$f"); done
```

也可以执行安全社区提供的脚本，检测主机安全性。

Intel 处理器

- pcid

pcid 用于降低 Meltdown（CVE-2017-5754）补丁 Kernel Page Table Isolation（KPTI）对性能的影响。由于 KPTI 将内核空间与用户空间分离并隐藏，关闭 PCID 的情况下，KPTI 将严重降低系统性能。
可以 root 权限执行如下命令，检测 Promxox VE 服务器是否支持 PCID：

```
# grep ' pcid ' /proc/cpuinfo
```

如命令返回不为空，则证明主机 CPU 支持 pcid。

- spec-ctrl

spec-ctrl 用于配合 Spectre v1（CVE-2017-5753）和 Spectre v2（CVE-2017-5755）补丁使

用，以弥补 retpoline 的不足。对于带有-IBRS 标识的 Intel CPU，默认包含了该特性。对于没有-IBRS 标识的 Intel CPU，需要升级 CPU 微码（intel-microcode>=20180425），并显式开启该功能。

- ssbd

Ssbd 参数和 Spectre V4（CVE-2018-3639）补丁配合使用。Intel CPU 默认不启用该特性。必须升级 CPU 微码（intel-microcode>=20180703），并显式启用该功能。

AMD 处理器

- ibpd

ibpd 用于配合 Spectre v1（CVE-2017-5753）和 Spectre v2（CVE-2017-5755）补丁使用，以弥补 retpoline 的不足。对于带有-IBRS 标识的 AMD CPU，默认包含了该特性。对于没有-IBRS 标识的 AMD CPU，需要升级 CPU 微码，并显式开启该功能。

- virt-ssbd

virt-ssbd 参数和 Spectre V4（CVE-2018-3639）补丁配合使用。AMD CPU 默认不启用该特性。必须显式启用该功能。即使不启用 amd-ssbd，也应当启用该功能并提供虚拟机使用，以便改善虚拟机兼容性。在虚拟机使用“host”类型 cpu 时，该功能必须显式启用。

- amd-ssbd

amd-ssbd 参数和 Spectre V4（CVE-2018-3639）补丁配合使用。AMD CPU 默认不启用该特性。必须显式启用该功能。启用 amd-ssbd 后，能在 virt-ssbd 基础上进一步改善虚拟机性能。因此，只要主机 CPU 支持，就应该启用该功能并提供给虚拟机使用。启用 virt-ssbd 能改善虚拟机兼容性，因为某些版本的内核只能识别 virt-ssbd。

- amd-no-ssb

amd-no-ssb 标识用于表示 CPU 不存在 Spectre V4 漏洞（CVE-2018-3639）。默认不包含在任何 AMD CPU 中。但在未来的 CPU 修补 CVE-2018-3639 漏洞后，可以通过设置启用 amd-no-ssb 标识通知虚拟机无需启用相关补丁。该参数不能和 virt-ssbd 和 amd-ssbd 同时使用。

NUMA

此外还可以选择在虚拟机上启用 NUMA 架构模拟功能。NUMA 架构的基本设计是，抛弃了以往多个内核共同使用一个大内存池的设计，而将内存按照 Socket 分配个每个 CPU 插槽。NUMA 能有效解决共用一个大内存池时的内存总线瓶颈问题，大大改善系统性能。如果你的物理服务器支持 NUMA 架构，我们推荐启用该配置，从而更合理地在物理服务器上分配虚拟机工作负载。此外，如果要使用虚拟机的 CPU 和内存热插拔，也需要启用该项配置。如果启用了 NUMA，建议为虚拟机分配和物理服务器一致的 Socket 数量。

vCPU 热插拔

现代操作系统开始支持 CPU 热插入功能，并在一定程度上支持 CPU 热拔出。虚拟化环境下，CPU 热插拔较真实物理服务器更为简单，因为无需考虑物理 CPU 插拔带来的各类硬件问题。但是，CPU 热插拔仍然是一个复杂且不成熟的功能特性，所以除非绝对需要，应严格限制使用该功能。但 10.2.5 节介绍的其他大部分功能都经过充分测试，且相对简单，可以放心使用。

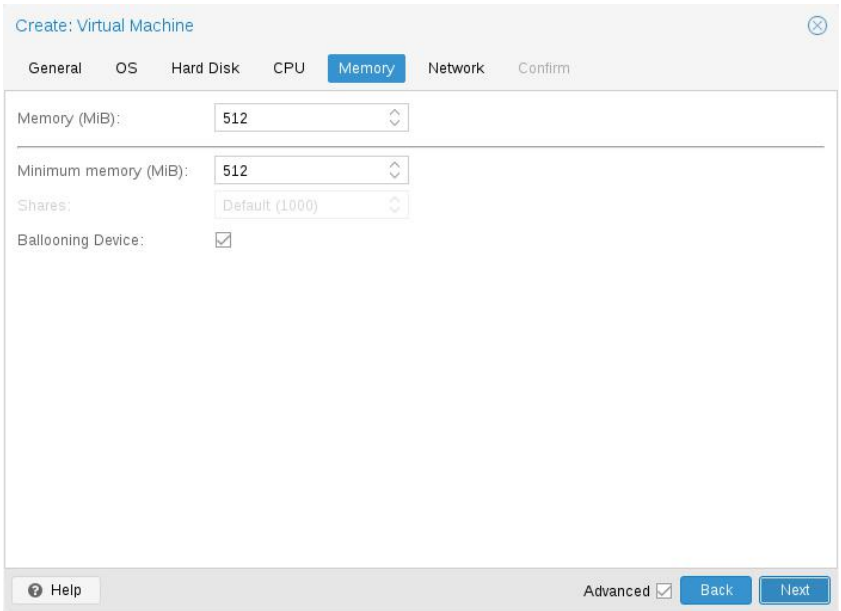
在 Proxmox VE 下，可热插拔的最大 CPU 数量为 cores*sockets 的乘积。对于一个虚拟 CPU

数量低于 COU 总数的虚拟机，可以启用 vpus 设置，以控制虚拟机启动时可启用的虚拟 CPU 数量。
目前，仅有 Linux 可以使用该特性，且 Linux 内核版本必须高于 3.10，推荐使用 4.7 以上 Linux 内核。
可以在 Linux 中按以下示例配置 udev 规则，在虚拟机中完成 CPU 热插入自动检测：
SUBSYSTEM=="cpu", ACTION=="add", TEST=="online", ATTR{online}=="0", ATTR{online}="1"
将上面的配置保存在/etc/udev/rules.d 下的配置文件中，配置文件后缀名为.rules 即可生效。
注意：CPU 热拔出依赖于设备硬件，并需要客户机操作系统的支持。CPU 删除命令并不一定真正移除 CPU，一般还需要将该 CPU 删除请求发送给虚拟机做进一步处理，CPU 删除请求的发送机制因硬件平台而异，如 x86/amd64 下就是 ACPI 机制。

10.2.6 内存

为虚拟机分配内存时，你可以选择分配固定容量内存或让 Proxmox VE 根据主机内存使用情况动态分配内存。

分配固定容量内存



当设置内存容量和最小内存容量为相同值时，Proxmox VE 将为虚拟机分配固定容量内存。即使使用固定容量内存，也可以在虚拟机启用 ballooning 设备，以监控虚拟机的实际内存使用量。通常情况下，应该启用 ballooning 设备，如需禁用，可以取消 ballooning 设备的勾选，或者在虚拟机配置文件中进行如下设置
balloon: 0

自动分配内存

当设置的最小内存容量低于设置的内存容量值时，Proxmox VE 将为虚拟机至少分配设置的最小容量内存，并在物理服务器内存占用率达到 80%之前根据虚拟机需要动态分配内存，直到达到设置的最大内存分配量。

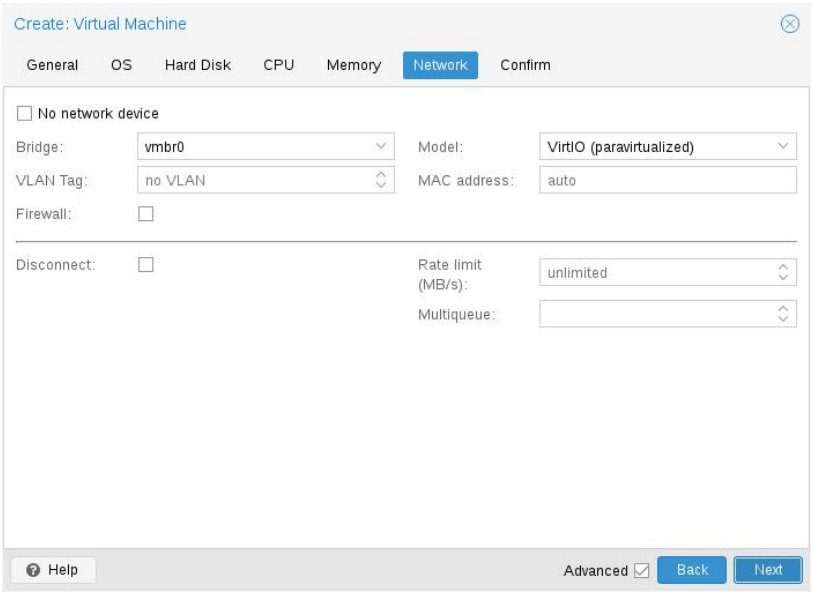
当物理服务器内存不足时，Proxmox VE 将开始回收分配给虚拟机的内存，并在必要时启动 SWAP 分区，如果仍然不能满足需要，最终将启动 oom 进程杀掉部分进程以释放内存。物理服务器和虚拟机之间的内存分配和释放通过虚拟机内的 balloon 驱动完成，该驱动主要用于从主机抓取或向主机释放内存页面。

当有多台虚拟机使用自动内存分配方式时，可以通过配置“shares”参数，在多个虚拟机之间分配可用内存份额。比如，假定现在有 4 台虚拟机，其中 3 台为 HTTP 服务器，1 台为数据库服务器。为了让数据库服务器能够使用更多内存缓存数据库数据，你会希望能优先给数据库服务器分配更多内存。为达此目的，可以设置数据库虚拟机的 Shares 为 3000，并设置其他 3 个 HTTP 虚拟机的 Shares 为默认值 1000。如果物理服务器有 32GB 内存，且目前已使用 16GB，那么可以提供给这 4 台虚拟机分配使用的物理内存有 32*80/100-16=9GB。数据库虚拟机能获得的内存容量为 9*3000/(3000+ 1000+1000+1000)=4.5GB，而每个 HTTP 虚拟机能获得 1.5GB。

2010 年以后，所有的 Linux 发行版默认都安装了 balloon 驱动。对于 Windows 系统，则需要手工安装 balloon 驱动，并且可能会导致系统性能降低，所以我们不建议在重要的 Windows 系统上安装 balloon 驱动。

当为虚拟机分配内存时，至少要为主机保留 1GB 可用内存。

10.2.7 网卡



虚拟机可以配置多个网卡，共有以下四种类型虚拟网卡可以选择使用：

- Intel E1000 是默认配置的网卡类型，模拟了 Intel 千兆网卡设备。
- VirtIO 是半虚拟化网卡，具有较高的性能。但和其他 VirtIO 虚拟设备一样，虚拟机必须安装 virtio 驱动程序。
- Realtek 8139 模拟了旧的 100Mb/s 的网卡。当虚拟机使用旧版操作系统（2002 年以前发行）时，可以使用该类型虚拟网卡。

- Vmxnet3 是另一种半虚拟化网卡。可用于从其他类型虚拟化平台导入的虚拟机。

Proxmox VE 会为每一块虚拟网卡生成一个随机的 MAC 地址，以便虚拟机网络通信使用。
虚拟网卡的工作模式分为以下两种：

- 桥接模式下，每个虚拟网卡的底层都使用物理服务器上的 tap 设备（软件实现的 loopback 物理网卡设备）实现。该 tap 设备被添加到虚拟交换机上，如 Proxmox VE 默认的 vmbr0，以便虚拟机直接访问物理服务器所连接的局域网 LAN。

- NAT 模式下，虚拟网卡将只能和 Qemu 的网络协议栈通信，并在内嵌的路由服务和 DHCP 服务的帮助下进行网络通信。内嵌的 DHCP 服务会在 10.0.2.0/24 范围内分配 IP 地址。由于 NAT 模式的性能远低于桥接模式，所以一般仅用于测试环境。该模式仅支持通过 CLI 或 API 使用，不能直接在 WebUI 编辑配置。

你可以在创建虚拟机时通过选择 No network device 跳过网络设备添加环节。

Multiqueue

如果你配置了 VirtIO 网卡，可以同时配置使用 Multiqueue 功能。启用 Multiqueue 可以让虚拟机同时使用多个虚拟 CPU 处理网络数据包，从而提高整体网络数据包处理能力。
在 Proxmox VE 下使用 VirtIO 网卡时，每个虚拟网卡的收发队列都传递给内核处理，每个收发队列的数据包都由虚拟主机驱动创建的一个内核线程负责处理。当启用 Multiqueue 后，可以为每个虚拟网卡创建多个收发队列交由主机内核处理。
使用 Multiqueue 时，推荐设置虚拟机收发队列数量和虚拟 CPU 数量一致。此外，还需要为每个虚拟 VirtIO 网卡设置多功能通道数量，命令如下：
ethtool -L eth0 combined X
其中 X 指虚拟机的虚拟 CPU 数量。
需要注意，当设置 multiqueue 参数值大于 1 时，网络流量增大会引发主机 CPU 和虚拟机 CPU 负载的升高。我们推荐仅在虚拟机需要处理大量网络数据包时启用该配置，例如用作路由器、反向代理或高负载 HTTP 服务器时。

10.2.8 显示器

QEMU 支持多种的虚拟化 VGA 硬件。如下：

- std，默认显卡，模拟基于 Bochs VBE 扩展的显卡。

- cirrus，以前的默认显卡，模拟一种非常古老的显卡硬件，缺陷较多。建议只在万不得已时再考虑使用该类型显卡，例如在使用 Windows XP 或更老版本操作系统时。

- vmware，模拟 VMWare 的 SVGA-II 类显卡。

- qxl，模拟 QXL 半虚拟化显卡。选择该类型显卡将同时为虚拟机启用 SPICE 显示器。

可以设置 memory 参数，调整虚拟 GPU 显存容量。设置更大显存能够帮助提高虚拟机所能达到的分辨率，特别在使用 SPICE/QXL 时。

由于显存是为显卡设备专门预留的，在 SPICE 下启用多显示器模式（例如，qx12 双显示器）时，需要注意以下事项：

- **Windows 需要为每个显示器配置一个显卡 ,如果 ostype 设置为 Windows ,Proxmox VE 将为虚拟机的每个显示器分配一个额外的显卡。每个显卡都会分配指定容量的显存。**
- **Linux 虚拟机默认可以拥有多个虚拟显示器，选择启用多显示器模式时，会根据显示器数量自动为显卡分配多份显存。**

选择使用 serialX 类型显卡时，会自动禁用 VGA 输出，并将 Web 控制台输出重定向到指定的串口。此时 memory 参数设置将不再生效。

10.2.9 USB 直通

Proxmox VE 支持两种 USB 直通方法：

- **基于主机的 USB 直通**
- **基于 SPICE 协议的 USB 直通**

基于主机的 USB 直通是将主机上的一个 USB 设备分配给虚拟机使用。具体可以通过指定厂商 ID 和设备 ID 分配，也可以通过指定主机总线号和端口号分配。
厂商/设备 ID 格式为：0123:abcd。其中 0123 为厂商 ID，abcd 为设备 ID，这意味着同样型号的 USB 设备将具有同样的 ID。
总线/端口编号格式为：1-2.3.4。其中 1 为总线号，2.3.4 为端口路径。合起来标识了主机上的一个物理端口（取决于 USB 控制器的内部顺序）。
即使虚拟机配置中的 USB 直通设备并未连接到物理服务器，虚拟机也可以顺利启动。在主机上指定的直通设备不可访问时，虚拟机会做跳过处理。

☒ 警告

由于 USB 直通设备只在当前主机上具备，所以使用 USB 直通的虚拟机将无法在线迁移到其他物理服务器。

第二种直通方式基于 SPICE 协议。这种直通方式需要 SPICE 客户端的支持。如果你给虚拟机添加了 SPICE USB 端口，那么就可以直接将 SPICE 客户端上的 USB 设备直通给虚拟机使用（例如输入设备或硬件加密狗）。

10.2.10 BIOS 和 UEFI

为了完美模拟计算机硬件，QEMU 使用了固件。也就是传统 PC 中的 BIOS 或(U)EFI，用于虚拟机的初始启动，完成基本的硬件初始化，并为操作系统提供硬件和固件访问接口。QEMU 默认使用开源 x86 BIOS 固件 SeaBIOS。大多数情况下，SeaBIOS 都是不错的选择。
当然，也有 BIOS 不能正常引导启动测场景。比如，在配置 VGA 直通时。此时，使用开源

UEFI 固件 OVMF 更好。
使用 OVMF 有以下几点需要注意：
为了保存启动顺序等配置，需要为虚拟机添加一个 EFI 硬盘，并纳入备份和快照管理范围，并且只能有一块 EFI 硬盘。
EFI 硬盘添加命令如下：
qm set <vmid> -efidisk0 <storage>:1,format=<format>
其中<storage>是 Proxmox VE 存储服务名，<format>是存储格式。你也可以在 Web 控制台提供的虚拟机硬件配置界面通过添加 EFI 硬盘来完成该操作。
在 OVMF 下使用虚拟显示器时（而非通过 VGA 直通），你需要在 OVMF 菜单（在虚拟机启动时按 ESC 键可调出该菜单）中配置终端显示器的分辨率，或者选择使用 SPICE 显示器。

10.2.11 内部虚拟机共享内存

可以通过内部虚拟机共享内存设备（ivshmem）实现主机和虚拟机之间的内存共享，或多个虚拟机之间的内存共享。
可以使用 qm 命令增加该类设备：
qm set <vmid> -ivshmem size=32,name=foo
其中 size 参数的单位是 MiB。产生的设备文件位于/dev/shm/pve-shm-$name（默认名称为 vmid）。

➢ 注意

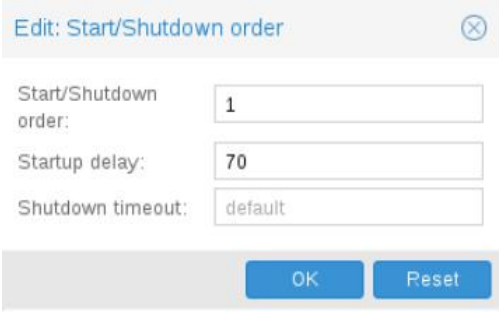
虚拟机关闭或停止时，该设备会自动删除。已有的打开会被保持，但新打开请求将会被拒绝。

该设备的一个应用场景是 Looking Glass 项目，用于在主机和客户机之间实现高性能、低延时的镜像显示功能。

10.2.12 虚拟机自启动和自关闭

创建虚拟机后，如果需要虚拟机在 Proxmox VE 物理服务器开机后自动运行，需要在 Web 控制台的虚拟机 Option 选项卡中选择“Start at boot“，或者运行以下命令：
qm set <vmid> -onboot 1

开机顺序和关机顺序



某些场景下，你可能需要仔细调整各个虚拟机的启动顺序，比如为其他虚拟机提供防火墙或 DHCP 服务的虚拟机应该先启动。可以设置以下参数调整开关机顺序。

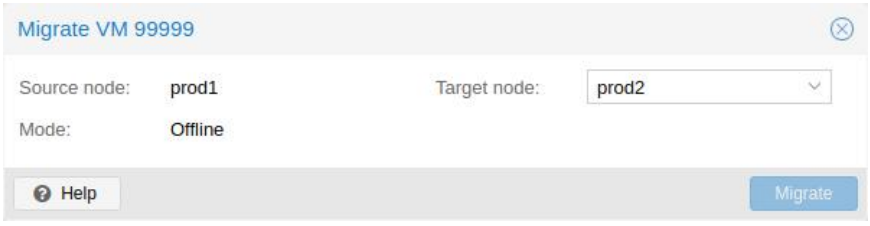
- Start/Shutdown order：用于设置开机优先级。例如，设置为 1 表示该虚拟机需要第一个被启动（关机顺序和开机顺序相反，所以设置为 1 的虚拟机会最后被关闭）。如果同一物理服务器上的多个虚拟机设置相同优先级，将按照其 VMID 升序依次启动。
- Startup delay：用于设置当前虚拟机开机后到下一个虚拟机开机前的时间间隔。例如，设置为 240 表示时间间隔为 240 秒。
- Shutdown timeout：用于设置关机命令发出后 Proxmox VE 等待虚拟机关机的时间间隔。该参数默认值为 180，也就是说 Proxmox VE 发出关机命令后会花 180 秒时间等待虚拟机完成关机操作，如果 180 秒后虚拟机仍未完成关机，Proxmox VE 会强制关机。

➢ 注意

启用 HA 管理的虚拟机，其开机自启动以及启动顺序设置将不再生效。开机关机算法自动忽略这些虚拟机，而由 HA 管理器负责开机关机操作。

需要注意，未设置 Start/Shutdown order 参数的虚拟机总是在设置了该参数的虚拟机之后启动，而且该参数仅能影响同一 Proxmox VE 服务器上虚拟机的启动顺序，其作用域局限于单一 Proxmox VE 服务器内部，而非整个集群。

10.3 虚拟机迁移



在 Promxox VE 集群中，可以将虚拟机迁移到其他服务器运行。命令如下：
qm migrate <vmid> <target>
迁移方式共有两种

- 在线迁移（也称为实时迁移）
- 离线迁移

10.3.1 在线迁移

如果虚拟机没有配置使用 Proxmox VE 服务器的本地资源（例如 local 存储上的虚拟磁盘，直通物理设备等），你可以增加-online 参数发起在线迁移命令，也就是在虚拟机开机运行状态下进行迁移操作。

工作原理

在线迁移时，目标服务器将启动一个 Qemu 进程，该进程设置有 incoming 标识，启动后将等待接收来自源虚拟机的内存数据和设备状态信息（由于其他资源，如磁盘数据等都在共享存储上，所以只需传输内存数据和设备状态即可）。
一旦建立连接，源虚拟机会以异步方式将内存数据传输给目标 Qemu 进程。如果在传输过程中内存数据发生了改变，相应的内存段会被标记成脏数据，并被再次传输。该过程将反复进行，直到剩余待传输数据量变得非常小，此时在线迁移将暂时冻结源虚拟机运行，并将剩余数据传输给目标，然后在目标节点恢复虚拟机继续运行，一般虚拟机中断运行时间不超过 1 秒钟。

先决条件

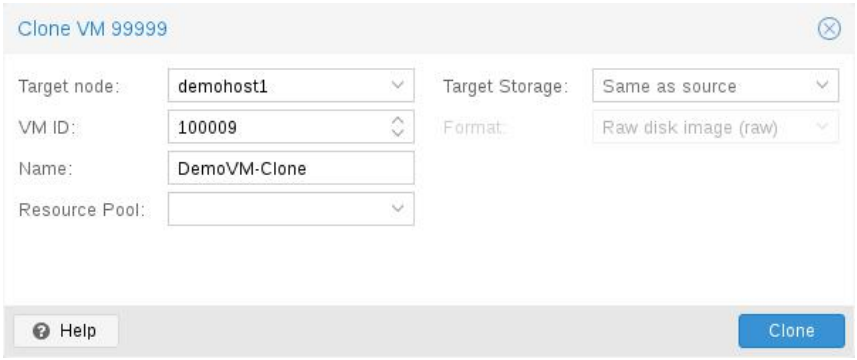
使用在线迁移需要以下先决条件：

- 虚拟机未使用本地资源（例如：直通设备，本地磁盘等）
- 源主机和目标主机在同一个 Proxmox VE 集群中。
- 源主机和目标主机有（可靠的）网络连接。
- 目标主机 Proxmox VE 版本不低于源主机（从高版本主机向低版本迁移有可能也可以，但不保证一定成功）

10.3.2 离线迁移

即使虚拟机使用了 Proxmox VE 服务器本地资源，但只要虚拟硬盘所处的存储服务在源服务器和目的服务器都有配置，仍然可以离线迁移虚拟机。迁移操作中，Proxmox VE 会通过网络将虚拟机硬盘镜像复制到目标服务器。

10.4 复制与克隆



通常虚拟机操作系统需要使用安装光盘（CD-ROM）手工安装。这往往是一个非常耗时的操作。

部署多个同类型虚拟机时，可以采用复制原有虚拟机的方式。这种复制操作称为 clone，并分为 linked（链接克隆）和 full（完整克隆）两类。

完整克隆

全克隆可以创建一个完全独立的虚拟机，新虚拟机与原虚拟机之间不存在任何共享存储资源。

可以选择目标存储，从而将虚拟机复制到一个完全不同的存储设备。同时可以根据存储支持的情况选择改用其他磁盘格式。

➢ 注意

完整克隆需要读取并复制虚拟机全部镜像数据。因此耗时往往较链接克隆长的多。

某些类型的存储支持复制指定快照，也就是当前的虚拟机数据。这也意味着最终的虚拟机不包含原虚拟机的其他快照。

链接克隆

现代存储驱动支持快速链接克隆技术。链接克隆生成一个可写副本，其初始内容和原数据一致。生成链接克隆的速度非常快，几乎可以瞬间完成，且刚创建时几乎不消耗存储空间。

顾名思义，链接克隆产生的新镜像仍然链接到源镜像。其核心技术称为 Copy-on-wirte，如果数据块被改写（然后再读取），将写到一个新位置，如果数据块未被修改过，将直接从源镜像读取。

➢ 注意

你不能删除创建有链接克隆的源模板。

创建链接克隆时不能改变目标存储，因为该技术依赖于存储内部功能特性。

通过设置目标节点选项，可以用链接克隆在其他节点创建新虚拟机。唯一需要确保的是，虚拟机保存在共享存储上，且共享存储已经挂载到目标节点。
为避免冲突，链接克隆虚拟机的所有网卡 MAC 地址都重新随机生成，虚拟机 BIOS（smbios1）的 UUID 也会重新生成。

10.5 虚拟机模板

可以将虚拟机转换为模板。模板是只读的，并可用于创建链接克隆。

- **注意**

模板不能再被启动，因为这将改变模板数据。如果想修改模板，可以先创建一个链接克隆，然后再修改。

10.6 虚拟机生成 ID

Proxmox VE 支持虚拟机生成 ID（vmgenid）功能。虚拟机操作系统可利用该功能检测操作系统时间漂移事件，比如备份恢复虚拟机或快照回滚等。
在新建虚拟机时，会自动生成 vmgenid 并写入虚拟机配置文件。
对于已有虚拟机，如果要新增 vmgenid，可以向虚拟机传递特殊值‘1’，Proxmox VE 就会自动创建。也可以手工指定 UUID 为 vmgenid 值。示例如下：
qm set VMID -vmgenid 1
qm set VMID -vmgenid 00000000-0000-0000-0000-000000000000

- **注意**

首次向已有虚拟机添加 vmgenid 时，虚拟机有可能将该操作理解为生成值改变，从而做出类似对快照回滚或备份恢复的响应。

如果确实有特殊原因，不希望启用 vmgenid，可以在创建虚拟机时设置值‘0’，或者在创建虚拟机后再执行删除该特性的命令，如下：
qm set VMID -delete vmgenid
微软 Windows 操作系统是使用 vmgenid 的典型场景，能通过该特性有效避免快照回滚，备份恢复或虚拟机克隆时导致时间敏感服务（例如，数据库，域控制器）异常。

10.7 虚拟机和磁盘镜像导入

其他虚拟机管理器导出的虚拟机一般由一个或多个磁盘镜像和一个虚拟机配置文件（描述内存，CPU 数量）构成。

如果虚拟机由 VMware 或 VirtualBox 导出，磁盘镜像有可能是 vmdk 格式，如果从 KVM 管理器导出，可能是 qcow2 格式。最流行的虚拟机导出格式是 OVF 标准，但实际上由于 OVF 标准本身不完善，以及虚拟机管理器导出的众多非标准扩展信息，跨管理器使用 OVF 往往受很多限制。
除了格式不兼容之外，如果虚拟机管理器之间的虚拟硬件设备差别太大，也可能导致虚拟机镜像导入失败。特别是 Windows 虚拟机，对于硬件变化特别敏感。为解决这一问题，可以在导出 Windows 虚拟机之前安装 MergeIDE.zip，并在导入后启动前将虚拟磁盘改为 IDE 类型。
最后还需要考虑半虚拟化驱动因素。半虚拟化驱动能够改善虚拟硬件性能，但往往针对特定虚拟机管理器。GNU/Linux 和其他开源 Unix 类操作系统默认已经安装所有必要的半虚拟化驱动，可以在导入虚拟机后直接改用半虚拟化驱动。对于 Windows 虚拟机，还需要自行安装 Windows 版本的半虚拟化驱动软件。
GNU/Linux 和其他开源 Unix 虚拟机通常可以直接导入。但由于以上提到的因素，不能保证所有 Windows 虚拟机均能够顺利导入成功。

10.7.1 Windows OVF 导入步骤示例

Microsoft 为 Windows 开发提供了虚拟机下载服务。以下将利用这些镜像演示 OVF 导入功能。

下载虚拟机镜像压缩包

在选择同意用户协议后，选择基于 VMware 的 Windows 10 Enterprise（Evaluation-Build），下载 zip 压缩包。

从 zip 压缩包提取磁盘镜像

使用 unzip 或其他工具解压缩 zip 压缩包，通过 ssh/scp 将 ovf 和 vmdk 文件复制到 Proxmox VE 服务器。

导入虚拟机

执行以下命令可以创建新虚拟机，虚拟机的 CPU、内存和名称沿用 OVF 配置文件中的设置，磁盘镜像将导入 local-lvm 存储。网络配置可以手工完成。

```
qm importovf 999 WinDev1709Eval.ovf local-lvm
```

至此，虚拟机导入完成，可以启动使用。

10.7.2 向虚拟机增加外部磁盘镜像

也可以将磁盘镜像直接添加到虚拟机。磁盘镜像可以是外部虚拟机管理器导出的，也可以是你自己创建的。
首先使用 vmdebootstrap 工具创建 Debian/Ubuntu 磁盘镜像：

```
vmdebootstrap --verbose \
--size 10GiB --serial-console \
--grub --no-extlinux \
```

```
--package openssh-server \
--package avahi-daemon \
--package qemu-guest-agent \
--hostname vm600 --enable-dhcp \
--customize=./copy_pub_ssh.sh \
--sparse --image vm600.raw
```
然后创建一个新的虚拟机。
```
qm create 600 --net0 virtio,bridge=vmbr0 --name vm600 --serial0 socket \
--bootdisk scsi0 --scsihw virtio-scsi-pci --ostype l26
```
将磁盘镜像以 unused0 导入虚拟机，存储位置为 pvedir：
```
qm importdisk 600 vm600.raw pvedir
```
最后将磁盘连接到虚拟机的 SCSI 控制器：
```
qm set 600 --scsi0 pvedir:600/vm-600-disk-1.raw
```
至此，虚拟机导入完成，可以启动使用。

10.8Cloud-Init 支持

Cloud-Init 兼容多个 Linux 发行版，主要用于虚拟机初始化配置。通过 Cloud-Init，虚拟机管理器可以直接配置虚拟机网络设备和 ssh 密钥。当虚拟机首次启动时，Cloud-Init 能够在虚拟机内部启用相关配置。

很多 Linux 发行版都提供了可直接使用的 Cloud-Init 镜像，多数都为 OpenStack 创建。这些镜像也可以直接用于 Proxmox VE。尽管可以直接使用官方镜像，但最好还是自己创建 Cloud-Init 镜像。自己创建镜像的好处是可以完全控制所安装的软件包，并可以按照自己的需求进行定制。

建议将创建的 Cloud-Init 镜像转换为虚拟机模板，并用该模板链接克隆快速创建虚拟机。启动虚拟机之前只需要完成网络配置（或许还有 ssh 密钥）即可。

推荐使用 Cloud-Init 提供的基于 SSH 密钥认证的方式登录虚拟机。当然也可以使用用户名口令的方式进行登录，但由于 Cloud-Init 会保存加密后的口令，所以基于 SSH 密钥的认证方式更安全。

Proxmox VE 通过 ISO 镜像方式向虚拟机传递配置数据。因此所有 Cloud-Init 虚拟机需要配置一个虚拟 CDROM 驱动器。很多 Cloud-Init 镜像会假定拥有串口控制台，为此推荐增加一个串口控制台并用于显示虚拟机信息。

10.8.1 准备 Cloud-Init 镜像

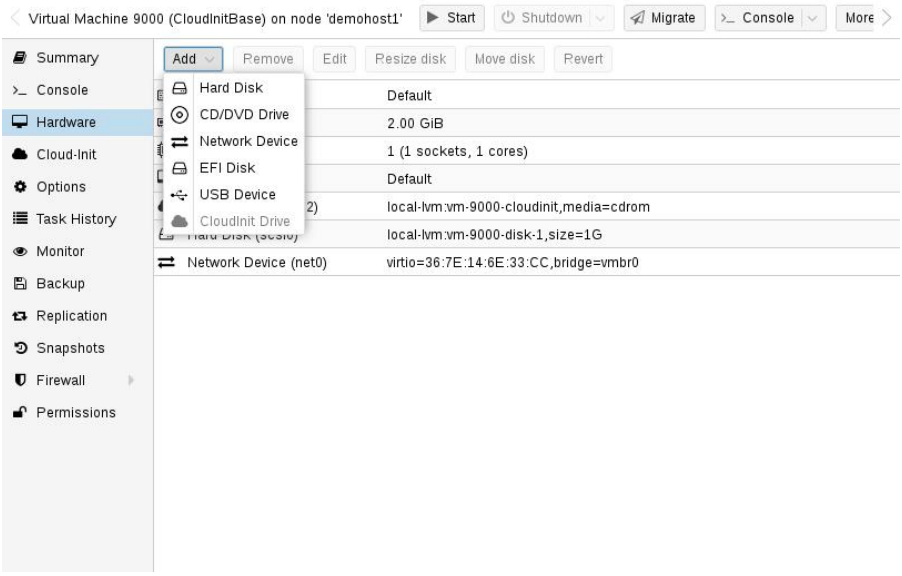
使用 Cloud-Init 的第一步是准备虚拟机。理论上可以使用任何虚拟机。只需在虚拟机内部安装 Cloud-Init 软件包即可。例如在基于 Debian/Ubuntu 的虚拟机上，执行以下命令即可：
```
apt-get install cloud-init
```
很多 Linux 发行版都提供可直接使用的 Cloud-Init 镜像（以.qcow2 文件形式），因此也可以直接下载并导入这类镜像。下面的例子就使用 Ubuntu 在 https://cloud-images.ubuntu.com 提供的云镜像。
```
# download the image
```

```
wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.img
# create a new VM
qm create 9000 --memory 2048 --net0 virtio,bridge=vmbr0
# import the downloaded disk to local-lvm storage
qm importdisk 9000 bionic-server-cloudimg-amd64.img local-lvm
# finally attach the new disk to the VM as scsi drive
qm set 9000 --scsihw virtio-scsi-pci --scsi0 local-lvm:vm-9000-disk-1
```

➢ 注意

在 Ubuntu 的 Cloud-Init 镜像中使用 SCSI 磁盘时，需要配置 virtio-scsi-pci 控制器。

增加 Cloud-Init CDROM 驱动器



接下来要为虚拟机配置 CDROM 驱动器，以便传递 Cloud-Init 配置数据。

```
qm set 9000 --ide2 local-lvm:cloudinit
```

为直接启动 Cloud-Init 镜像，需要将 bootdisk 设置为 scsi0，并设置仅从磁盘启动。这可以省去虚拟机 BIOS 的自检并加速启动过程。

```
qm set 9000 --boot c --bootdisk scsi0
```

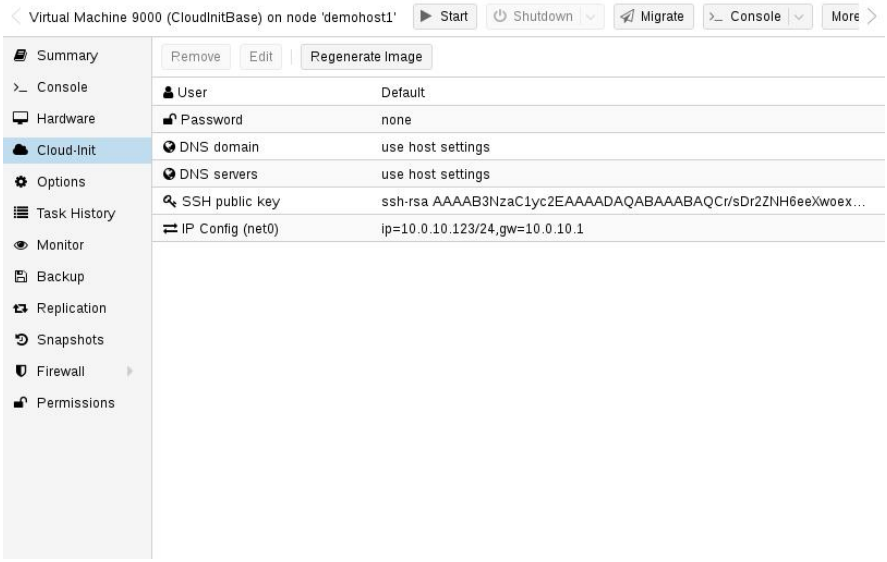
此外还需要配置一个串口控制台，并用于显示虚拟机信息。由于这是 OpenStack 镜像的标准要求，所以有很多 Cloud-Init 镜像都依赖这种配置。

```
qm set 9000 --serial0 socket --vga serial0
```

最后，可以将虚拟机转换为模板。通过该模板，可以用链接克隆快速创建新的虚拟机。这种部署方式比完整克隆（复制）要快得多。

```
qm template 9000
```

10.8.2 部署 Cloud-Init 模板



利用模板可以轻松克隆并部署虚拟机
qm clone 9000 123 –name ubuntu2
然后设置登录认证 SSH 公钥，并配置 IP 地址：
qm set 123 --sshkey ~/.ssh/id_rsa.pub
qm set 123 --ipconfig0 ip=10.0.10.123/24,gw=10.0.10.1
可以通过一个命令行配置 Cloud-Init 的全部参数项。上面的例子是为了避免命令行过长而做了拆分。此外，需要注意确保 IP 配置符合你的网络环境要求。

10.8.3 自定义 Cloud-Init 配置

Cloud-Init 允许用户使用自定义配置文件。具体可以通过 cicustom 选项实现，具体如下：
qm set 9000 --cicustom "user=<volume>,network=<volume>,meta=<volume>"

用户的配置文件必须在共享存储上，且必须是虚拟机所在节点能够访问的，否则虚拟机将不能启动。示例如下：
qm set 9000 --cicustom "user=local:snippets/userconfig.yaml"

一共有三类配置。第一类是上面例子中的 user 配置参数。第二类是 network 配置，第三类是 meta 配置。三类参数都可以同时设定，或根据需要任意组合匹配。如果未使用自定义配置，系统将自动产生一个配置并启用该配置。
自动生成的配置可作为用户自定义配置的基础模板。
qm cloudinit dump 9000 user
同样的命令也可以用于 network 和 meta 配置。

10.8.3 Cloud-Init 参数

- cicustom: [meta=<volume>] [,network=<volume>] [,user=<volume>]

使用指定文件代替自动生成文件。

meta=<volume>

将包含所有元数据的指定文件通过 cloud-init 传递给虚拟机。该文件提供指定 configdrive2 和 nocluod 信息。

network=<volume>

将包含所有网络配置数据的指定文件通过 cloud-init 传递给虚拟机。

user=<volume>

将包含所有用户配置数据的指定文件通过 cloud-init 传递给虚拟机。

- cipassword: <string>

用户口令。通常推荐使用 SSH 密钥认证，不要使用口令方式认证。请注意，旧版 Cloud-Init 不支持口令 hash 加密。

- citype: <configdrive2 | nocloud>

指定 Cloud-Init 配置数据格式。默认依赖于操作系统类型（ostype）。Linux 可设置为 nocloud，Windows 可设置为 configdrive2。

- ciuser: <string>

指定用户名，同时不再使用镜像配置的默认用户。

- ipconfig[n]: [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>]

为对应端口设置 IP 地址和网关。

IP 地址采用 CIDR 格式，网关为可选项，也采用 CIDR 格式 IP 形式设置。

在 DHCP 环境中可将 IP 地址设置为字符串 dhcp，此时应将网关留空。在 IPv6 网络中，如需启用无状态自动配置，将 IP 设置为字符串 auto 即可。

如未设置 IPv4 或 IPv6 地址，Cloud-Init 默认将使用 IPv4 的 dhcp。

gw=<GatewayIPv4>

IPv4 的默认网关

◆ 注意

要配合使用选项：ip

gw6=<GatewayIPv6>

IPv6 的默认网关

◆ 注意

要配合使用选项：ip6

ip=<IPv4Format/CIDR> (default = dhcp)

IPv4 地址，采用 CIDR 格式。

ip=<IPv6Format/CIDR> (default = dhcp)

IPv6 地址，采用 CIDR 格式。

- nameserver: <string>

为容器设置 DNS 服务器 IP 地址。如未设置 searchdomian 及 nameserver，将自动采用服务器主机设置创建有关配置。

- searchdomain: <string>

为容器设置 DNS 搜索域。如未设置 searchdomian 及 nameserver，将自动采用服务器主机设置创建有关配置。

- sshkeys: <string>

设置 SSH 公钥（每行设置一个 key，OpenSSH 格式）。

10.9PCI(e)直通

PCI(e)直通可以让虚拟机直接控制物理服务器的 PCI 硬件设备。与使用虚拟化硬件相比，这种方式的优势有低延迟，高性能以及其他功能特性（例如，任务卸载）。
主要缺点是，一旦采用直通方式，对应硬件就不能再被主机或其他虚拟机使用。

10.9.1 通用要求

硬件直通往往需要硬件设备的支持，以下是启用该功能的前置检查项目和准备工作清单。

硬件设备

硬件设备需要支持 IOMMU（I/O Memory Management Unit）中断重映射，这需要 CPU 和主板的支持。
通常，具备 Intel VT-d 功能的 Intel 硬件系统，或具备 AMD-Vi 功能的 AMD 硬件系统均可以满足要求。但这不意味着直通功能可以开箱即用，硬件设备缺陷，驱动软件不完备等因素都可能导致硬件直通无法正常工作。
一般来说，大部分硬件都可以支持该功能，但服务器级硬件一般比消费级硬件能更好支持直通功能。
可以联系硬件设备厂商，以确定你的硬件设备是否在 Linux 下支持直通功能。

配置

确定硬件支持直通功能后，还需要完成相应配置才行。

IOMMU

首先需要在内核命令行启用 IOMMU 功能，见 3.10.4 节。命令行参数如下：

- Intel CPU

Intel_iommu=on

- AMD CPU

amd_iommu=on

内核模块

将以下内容添加到配置文件'/etc/modules'中，确保内核加载相应模块
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
然后还需要更新 initramfs。命令行如下：
#update-initramfs -u -k all
如果使用了 systemd-boot，请务必按照相关内容将新的 initramfs 同步到引导分区并更新所有的 ESP 配置文件。

完成配置

配置完成后，需要重启以启用新配置。可用以下命令行查看新配置是否已启用。
#dmesg | grep -e DMAR -e IOMMU -e AMD-Vi
如果显示 IOMMU，Directed I/O 或 Interrupt Remapping 已启用则表明配置已生效。具体显示内容随硬件类型而有所不同。
此外还需要确保直通硬件在独立的 IOMMU 组中。可用如下命令查看：
#find /sys/kernel/iommu_groups/ -type 1
如硬件与其功能、根端口或 PCI(e)桥在同一 IOMMU 组也可以正常直通，不受影响。

PCI(e)插槽

某些平台处理 PCI(e)插槽的方式较为特别。如果发现硬件所在 IOMMU 组不符合直通要求，可以换个插槽试试看。

不安全的中断

某些平台允许使用不安全的中断。如需启用该功能，可以在/etc/modprobe.d/目录下新增'.conf'配置文件，并在配置文件中增加以下内容：
options vfio_iommu_type1 allow_unsafe_interrupts=1
务必注意，启用该功能可能导致系统运行不稳定。

GPU 直通注意事项

首先，Web GUI 的 NoVNC 和 SPICE 控制台都不能显示直通 GPU 的显存内容。
如果要通过直通 GPU 或 vGPU 显示图形输出，必须将物理显示器连接到显卡，或者通过虚拟机内的远程桌面软件（如 VNC 或 RDP）才可以。
如果只是把 GPU 当做硬件加速器使用，比如运行 OpenCL 或 CUDA 程序，则不受以上所说情况影响。

10.9.2 主机设备直通

大部分 PCI(e)直通就是把整个 PCI(e)卡直通，例如 GPU 或网卡直通。

主机配置

硬件设备一旦设为直通，主机将不能再使用。具体有两种配置方法：

- 将设备 IDs 作为参数传递给 vfio-pci 模块

在/etc/modprobe.d/目录新增'.conf'配置文件，文件内容示例如下
options vfio-pci ids=1234:5678,4321:8765
其中 1234:5678 和 4321:8765 是厂商和设备 IDs，具体可以执行以下命令查看获取
#lcpci -nn

- 对主机屏蔽驱动以确保可供直通使用

在/etc/modprobe.d/目录新增'.conf'配置文件，文件内容示例如下
blacklist DRIVERNAME
两种方法都需要更新 initramfs 并重启系统确保配置生效，具体参见 10.9.1 节。

虚拟机配置

完成设备直通，还需要为虚拟机设置 hostpciX 参数，示例如下：
#qm set VMID -hostpci0 00:02.0
如果设备有多个功能（例如，'00:02.0'和'00:02.1'），可以用'00:02'即可将其一并直通。
另外，根据设备类型和客户机操作系统的不同，可能还需要设置一些附加参数：

- x-vga=on|off

用于将 PCI(e)设备标记为客户机的主 GPU。启用该参数后，虚拟机的 vga 配置将被忽略。

- pcie=on|off

通知 Proxmox VE 启用 PCIe 或 PCI 端口。某些客户机/设备组合需要启用 PCIe，而不是 PCI。而 PCIe 仅在某些 q35 类型的机器上可用。

- rombar=on|off

用于设置 ROM 是否对客户机可见。默认为可见。某些 PCI(e)设备需要设为对客户机不可见。

- Romfile=<path>

用于设置设备 ROM 文件路径，为可选参数。该路径是相对于/usr/share/kvm/的相对路径。

示例

下面的例子通过 PCIe 直通主 GPU 设备：
#qm set VMID -hostpci0 02:00,pcie=on,x-vga=on

其他注意事项

为获得更好的兼容性，在直通 GPU 设备时，最好选择 q35 芯片组，并选择使用 OVMF（针对虚拟机的 EFI）代替 SeaBIOS，选择 PCIe 代替 PCI。注意，在使用 OVMF 时，同时需要准

备好 EFI ROM，否则还得使用 SeaBIOS。

10.9.3 SR-IOV

另一种 PCI(e)直通方法是利用设备自带的硬件虚拟化功能。当然，这需要硬件设备本身具备相应功能。
SR-IOV（**S**ingle-**R**oot **I**nput/**O**utput **V**irtualization）技术可以支持硬件同时提供多个 VF（**V**irtual **F**unction）供系统使用。每个 VF 都可以用于不同虚拟机，不仅可以提供硬件的全部功能，而且比软件虚拟化设备性能更好，延迟更低。
目前，最常见的支持 SR-IOV 的设备是网卡（**N**etwork **I**nterface **C**ard）。能将单一物理端口虚拟化为多个 VF，同时允许虚拟机调用校验卸载等等硬件特性，从而降低主机 CPU 负载。

主机配置

有两种方法可以启用硬件设备虚拟化功能 VF。

- 设置启用驱动程序中的相应参数

例如 Intel 驱动中的
max_vfs=4
该参数可以配置在/etc/modprobe.d/目录下的.conf 配置文件中。（修改配置后不要忘记更新 initramfs 文件）
参数的具体信息和设置方法可以查看驱动程序文档。

- 通过 sysfs 设置启用

如果设备硬件和驱动程序支持，可以在线调整 VF 数量。例如，可以通过以下命令在设备 0000:01:00.0 上启用 4 个 VF：
#echo 4 > /sys/bus/pci/devices/0000:01:00.0/sriov_numvfs
如果需要将该配置永久生效，可以安装‘sysfsutils’软件包，并在/etc/sysfs.conf 中配置相关参数，或在/etc/sysfs.d/目录下专门创建.conf 配置文件也可以。

虚拟机配置

创建 VF 后，可以运行 lspci 命令查看相应的 PCI(e)设备信息，并根据相应设备 ID 进行直通配置，具体步骤可参考 10.9.2 节普通 PCI(e)设备直通配置。

其他注意事项

配置 SR-IOV 直通，硬件平台支持是尤为重要的。有可能首先要在 BIOS/EFI 中设置启用相关功能，或使用特定 PCI(e)端口。如有疑问，还要咨询平台厂商或查看有关手册才可以。

10.9.4 中介设备（vGPU，GVT-g）

中介设备（mediated device）也是一种实现硬件功能和性能复用的硬件虚拟化技术，多见于 GPU 虚拟化配置中，如 Intel GVT-g 和 Nvidia vGPU。
利用该技术，一个物理硬件设备可以创建多个虚拟设备，效果类似于 SR-IOV。主要区别在

于，中介设备不产生新的 PCI(e)设备，并且只适用于虚拟机。

主机配置

首先，硬件卡需要支持中介设备技术。可以从厂商获取驱动以及相关配置文档。
Intel 的 GVG-g 驱动已经集成在 Linux 内核，并可以在第 5、6、7 代 Intel Core CPU 直接使用，在 E3 v4、E3 v5 和 E3 v6 版本的 Xeon CPU 上也可以直接使用。
在 Intel 显卡上启用该技术，首先要确保已经加载 kvmgt 内核模块（例如将其写进配置文件 /etc/modules），并按 3.10.4 节内核命令行相关内容，添加如下参数：
l915.enable_gvt=1
然后还要按 10.9.1 节内容更新 initramfs，并重启服务器主机。

虚拟机配置

配置直通中介设备，只需要设置虚拟机的 hostpciX 参数的 mdev 属性即可。
具体可以通过 sysfs 查看所支持的硬件设备。如下例，列出 0000:00:02.0 下所有的设备类型：
#ls /sys/bus/pci/devices/0000:00:02.0/mdev_supported_types
每个条目都是一个目录，其中需要关注的重要文件有：

- available_instances

用于记录当前可用的实例数量，每在虚拟机中使用一个 mdev，该计数都会减 1。

- description

包含该类设备的功能简短描述

- create

是一个功能点，用于创建该类设备。如果 hostpciX 的 mdev 属性被配置启用，Proxmox VE 会自动调用该功能点。
下面是针对 Intel GVT-g vGPU（Intel Skylake 6700k）的配置示例：
#qm set VMID -hostpci0 00:02.0,mdev=i915-GVTg_V5_4
执行以上命令后，Proxmox VE 会在虚拟机启动时自动创建该设备，并在虚拟机停止时自动删除清理。

10.10 回调脚本

可以使用 hookscript 属性设置虚拟机回调脚本。
qm set 100 -hookscript local:snippets/hookscript.pl
该脚本会在虚拟机生命周期的多个阶段被调用。如需查看具体例子和相关文档，可以在 /usr/share/pve-docs/examples/guest-example-hookscript.pl 查看范例脚本。

10.11 虚拟机管理命令 qm

命令 qm 是 Proxmox VE 提供的用于管理 Qemu/KVM 虚拟机的命令行工具。通过该命令，可以创建或销毁虚拟机，也可以控制虚拟机运行状态（启动/停止/挂起/恢复）。此外，还可以利用 qm 设置虚拟机配置文件中的参数，创建或删除虚拟磁盘。

10.11.1 命令行示例

用 local 存储中的 iso 文件，在 local-lvm 存储中创建一个虚拟机，配置 4GB 的 IDE 虚拟硬盘。
qm create 300 -ide0 local-lvm:4 -net0 e1000 -cdrom local:iso/proxmox-mailgateway_2.1.iso
启动新建虚拟机。
qm start 300
发出关机命令，并等待直到虚拟机关机。
qm shutdown 300 && qm wait 300
发出关机命令，并等待 40 秒。
qm shutdown 300 && qm wait 300 -timeout 40

10.12 虚拟机配置文件

虚拟机配置文件保存在 Proxmox 集群文件系统中，并可以通过路径 /etc/pve/qemu-server/<VMID>.conf 访问。和/etc/pve 下的其他文件一样，虚拟机配置文件会自动同步复制到集群的其他节点。

☒ 注意

小于 100 的 VMID 被 Proxmox VE 保留内部使用，并且在集群内的 VMID 不能重复。

虚拟机配置文件示例

```
cores: 1
sockets: 1
memory: 512
name: webmail
ostype: l26
bootdisk: virtio0
net0: e1000=EE:D2:28:5F:B6:3E,bridge=vmbr0
virtio0: local:vm-100-disk-1,size=32G
```

虚拟机配置文件就是普通文本文件，可以直接使用常见文本编辑器（vi，nano 等）编辑。这也是日常对虚拟机配置文件进行细微调整的一般做法。但是务必注意，必须彻底关闭虚拟机，然后再启动虚拟机，修改后的配置才能生效。
因此，更好的做法是使用 qm 命令或 WebGUI 来创建或修改虚拟机配置文件。Proxmox VE 能够直接将大部分变更直接应用到运行中的虚拟机，并即时生效。该特性称为“热插拔“，并无需重启虚拟机。

10.12.1 配置文件格式

虚拟机配置文件使用英文冒号字符":"为分隔符的键/值格式。格式如下：
this is a comment
OPTION: value
空行会被自动忽略，以字符"#"开头的行按注释处理，也会被自动忽略。

10.12.2 虚拟机快照

创建虚拟机快照后，qm 会在配置文件中创建一个小节，专门保存创建虚拟机快照时的虚拟机配置。例如，创建名为"testsnapshot"的虚拟机快照后，虚拟机配置文件内容可能会像下面这样：

创建快照后的虚拟机配置文件示例

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

其中 parent 和 snaptime 是和虚拟机快照相关的配置属性。属性 parent 用于保存快照之间的父/子关系，属性 snaptime 是创建快照的时间戳（Unix epoch）。

10.12.3 虚拟机配置项目

- acpi：<boolean> (default = 1)

  启用/禁用 ACPI。
- agent: [enabled=]<1|0> [,fstrim_cloned_disks=<1|0>]

  启用/禁用 Qemu GuestAgent 及其属性。
  enabled=<boolean> (default = 0)
  启用/禁用 Qemu GuestAgent。
  fstrim_cloned_disks=<boolean> (default = 0)
  在克隆/迁移虚拟磁盘后运行 fstrim。
- arch: <aarch64 | x86_64>

  虚拟 CPU 架构。默认为 host。
- args：<string>

传递给 kvm 的任意参数，例如：

args: -no-reboot -no-hpet

☒ 注意

配置项目 args 仅供专家使用。

- autostart : <boolean> (default = 0)

  虚拟机崩溃后自动启动（目前该属性会被自动忽略）
- balloon : <integer> (0 -N)

  为虚拟机配置的目标内存容量，单位为 MB。设为 0 表示禁用 ballon 驱动程序。
- bios : <ovmf | seabios> (default = seabios)

  设置 BIOS 类型。
- boot : [acdn]{1,4} (default = cdn)

  虚拟机启动顺序，软驱（a），硬盘（c），光驱（d），或网络（n）。
- bootdisk : (ide|sata|scsi|virtio)\d+

  指定硬盘为系统启动盘。
- cdrom : <volume>

  相当于-ide2 的别名。
- cicustom: [meta=<volume>] [,network=<volume>] [,user=<volume>]

  使用指定文件代替自动生成文件。

  meta=<volume>

  将包含所有元数据的指定文件通过 cloud-init 传递给虚拟机。该文件提供指定 configdrive2 和 nocluod 信息。

  network=<volume>

  将包含所有网络配置数据的指定文件通过 cloud-init 传递给虚拟机。

  user=<volume>

  将包含所有用户配置数据的指定文件通过 cloud-init 传递给虚拟机。
- cipassword: <string>

  Cloud-Init：用户口令。通常推荐使用 SSH 密钥认证，不要使用口令方式认证。请注意，旧版 Cloud-Init 不支持口令 hash 加密。
- citype: <configdrive2 | nocloud>

  Cloud-Init：指定 Cloud-Init 配置数据格式。默认依赖于操作系统类型（ostype）。Linux 可设置为 nocloud，Windows 可设置为 configdrive2。
- ciuser: <string>

  Cloud-Init：指定用户名，同时不再使用镜像配置的默认用户。
- cores : <integer> (1 -N) (default = 1)

  每个虚拟 CPU 的核心数。

- cpu: [cputype=]<enum> [,flags=<+FLAG[;-FLAG...]>] [,hidden=<1|0>][,hv-vendor-id=<vendor-id>]

  模拟 CPU 类型。

  cputype=<486 | Broadwell | Broadwell-IBRS | Broadwell-noTSX | Broadwell-noTSX-IBRS | Conroe | EPYC | EPYC-IBPB | Haswell | Haswell-IBRS | Haswell-noTSX | Haswell-noTSX-IBRS | IvyBridge | IvyBridge-IBRS | Nehalem | Nehalem-IBRS | Opteron_G1 | Opteron_G2 | Opteron_G3 | Opteron_G4 | Opteron_G5 | Penryn | SandyBridge | SandyBridge-IBRS | Skylake-Client | Skylake-Client-IBRS | Skylake-Server | Skylake-Server-IBRS | Westmere | Westmere-IBRS | athlon | core2duo | coreduo | host | kvm32 | kvm64 | max | pentium | pentium2 | pentium3 | phenom | qemu32 | qemu64> (default = kvm64)

  以上为可选的模拟 CPU 类型值。

  flags=<+FLAG[;-FLAG...]>

  CPU 标识列表，分隔符为分号";"，启用标识使用+FLAG，禁用标识使用-FLAG。目前支持的标识有：pcid, spec-ctrl, ibpb, ssbd, virt-ssbd, amd-ssbd, amd-no-ssb, pdpe1gb,md-clear。

  hidden = <boolean> (default = 0)

  设为 1 表示不标识为 KVM 虚拟机。

  hv-vendor-id=<vendor-id>

  Hyper-V 厂商 ID。Windows 客户机的部分驱动或程序可能需要指定 ID。

- cpulimit：<number> (0 -128) (default = 0)

  CPU 配额上限值。

☒ 注意

如果一台计算机有 2 个 CPU，那么该计算机一共有 2 份额的 CPU 时间片可以分配。设为 0 表示不限制 CPU 配额。

- cpuunits：<integer> (2 -262144) (default = 1024)

  虚拟机的 CPU 时间片分配权重值。该参数供内核的公平调度器使用。设定的权重值越大，虚拟机得到的 CPU 时间片越多。最终分配得到的时间片由该虚拟机权重和所有其他虚拟机权重总和之比决定。

- description：<string>

  虚拟机描述信息。仅供 WebGUI 使用。在虚拟机配置文件中以注释形式保存。

- efidisk0：[file=]<volume> [,format=<enum>] [,size=<DiskSize>]

  配置 EFI 类型虚拟硬盘。

  file = <volume>

  EFI 虚拟硬盘所基于的存储服务卷名称。

  format = <cloop | cow | qcow | qcow2 | qed | raw | vmdk>

EFI 虚拟硬盘所采用的存储格式。

size = <DiskSize>

EFI 虚拟硬盘容量。仅供显示使用，并不能影响实际容量大小。

- freeze：<boolean>

  虚拟机启动时自动冻结 CPU（使用监视器命令 c 可继续启动过程）。

- hookscript: <string>

  设置回调脚本。

- hostpci[n]: [host=]<HOSTPCIID[;HOSTPCIID2...]> [,mdev=<string>][,pcie=<1|0>] [,rombar=<1|0>] [,romfile=<string>] [,x-vga=<1|0>]

  将物理主机 PCI 设备映射给虚拟机。

☒ 注意

该属性允许虚拟机直接访问物理主机硬件。启用后将不能再进行虚拟机迁移操作，因此使用时务必小心。

☒ 警告

该特性仍处于试验阶段，有用户报告该属性会导致故障和问题。

host = <HOSTPCIID[;HOSTPCIID2...]>

将 PCI 设备直通虚拟机使用。可指定一个或一组设备的 PCI ID。HOSTPCIID 格式为“总线号:设备号.功能号“（16 进制数字表示），具体可使用 lspci 命令查看。

mdev=<string>

表示中介设备类型。虚拟机启动时会自动创建设备，停止时自动删除清理。

pcie = <boolean> (default = 0)

标明是否是 PCI-express 类型总线（用于 q35 类型计算机）。

rombar = <boolean> (default = 1)

标明是否将设备 ROM 映射至虚拟机内存空间。

romfile=<string>

Pci 设备的 rom 文件名（文件需要保存在/usr/share/kvm/下）。

x-vga = <boolean> (default = 0)

标明是否启用 vfio-vga 设备支持。

- hotplug：<string> (default = network,disk,usb)

  设置启用的热插拔设备类型。启用热插拔的设备类型之间用英文逗号字符分隔，可选参数值包括 network，disk，cpu，memory 和 usb。设为 0 表示禁用热插拔，设为 1 表示启用默认值 network,disk,usb.

- hugepages：<1024 | 2 | any>

  启用/禁用巨型页。

- ide[n]: [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>][,bps=<bps>] [,bps_max_length=<seconds>] [,bps_rd=<bps>][,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]

[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]
[,detect_zeroes=<1|0>][,discard=<ignore|on>] [,format=<enum>]
[,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>]
[,iops_max_length=<seconds>] [,iops_rd=<iops>] [,iops_rd_max=<iops>]
[,iops_rd_max_length=<seconds>][,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>][,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>]
[,mbps_wr_max=<mbps>][,media=<cdrom|disk>] [,model=<model>]
[,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,secs=<integer>]
[,serial=<serial>][,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>]
[,ssd=<1|0>][,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>]

配置 IDE 类型虚拟硬盘或光驱（n 的值为 0-3）。

aio = <native | threads>

指定 AIO 类型。

backup = <boolean>

设置虚拟硬盘在进行虚拟机备份时是否被纳入备份范围。

bps = <bps>

最大读写操作速度，单位为字节/秒。

bps_max_length = <seconds>

突发读写操作最大时间长度，单位为秒。

bps_rd = <bps>

最大读操作速度，单位为字节/秒。

bps_rd_max_length = <seconds>

突发读操作最大时间长度，单位为秒。

bps_wr = <bps>

最大写操作速度，单位为字节/秒。

bps_wr_max_length = <seconds>

突发写操作最大时间长度，单位为秒。

cache = <directsync | none | unsafe | writeback | writethrough>

虚拟硬盘缓存工作模式。

cyls = <integer>

强制指定虚拟硬盘物理几何参数中的 cylinder 值。

detect_zeroes = <boolean>

设置是否检测并优化零写入操作。

discard = <ignore | on>

设置是否向下层存储服务传递 discard/trim 操作请求。

file = <volume>

IDE 虚拟硬盘所基于的存储服务卷名称。

format = <cloop | cow | qcow | qcow2 | qed | raw | vmdk>

IDE 虚拟硬盘所采用的存储格式。

heads = <integer>

强制指定虚拟硬盘物理几何参数中的 head 值。

iops = <iops>

最大读写 I/O 速度，单位为个/秒。

iops_max = <iops>

最大无限制读写 I/O 速度，单位为个/秒。
iops_max_length = <seconds>
突发读写操作最大时间长度，单位为秒。
iops_rd = <iops>
最大读 I/O 速度，单位为个/秒。
iops_rd_max = <iops>
最大无限制读 I/O 速度，单位为个/秒。
iops_rd_max_length = <seconds>
突发读操作最大时间长度，单位为秒。
iops_wr = <iops>
最大写 I/O 速度，单位为个/秒。
iops_wr_max = <iops>
最大无限制写 I/O 速度，单位为个/秒。
iops_wr_max_ length = <seconds>
突发写操作最大时间长度，单位为秒。
mbps = <mbps>
最大读写操作速度，单位为 MB/秒。
mbps_max = <mbps>
最大无限制读写操作速度，单位为 MB/秒。
mbps_rd = <mbps>
最大读操作速度，单位为 MB/秒。
mbps_rd_max = <mbps>
最大无限制读操作速度，单位为 MB/秒。
mbps_wr = <mbps>
最大写操作速度，单位为 MB/秒。
mbps_wr_max = <mbps>
最大无限制写操作速度，单位为 MB/秒。
media = <cdrom | disk> (default = disk)
虚拟硬盘驱动器介质类型。
model = <model>
虚拟硬盘的模型名，基于 url 编码格式，最大 40 字节。
replicate=<boolean> (default = 1)
磁盘是否被调度复制。
rerror = <ignore | report | stop>
读错误处理方式。
secs = <integer>
强制指定虚拟硬盘物理几何参数中的 sector 值。
serial = <serial>
虚拟硬盘的序列号，基于 url 编码格式，最大 20 字节。
shared=<boolean> (default = 0)
将本地管理卷标记为所有节点均可用。

☒ 警告

该选项并不自动共享卷，只是假定该卷已经被共享。

size = <DiskSize>

虚拟硬盘容量。仅供显示使用，并不能影响实际容量大小。

snapshot = <boolean>

Qemu 快照功能控制选项。设置后，对磁盘的改写会被当成临时的，并在虚拟机重启后全部丢弃。

ssd=<boolean>

设置虚拟磁盘连接到虚拟机的方式，SSD 或硬盘。

trans = <auto | lba | none>

设置虚拟硬盘几何参数地址 bios 解释模式。

werror = <enospc | ignore | report | stop>

写错误处理方式。

wwn = <wwn>

驱动器的唯一名称，使用 16 字节 hex 字符串表示，前缀为 0x。

- ipconfig[n]: [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>]

Cloud-Init：为对应端口设置 IP 地址和网关。

IP 地址采用 CIDR 格式，网关为可选项，也采用 CIDR 格式 IP 形式设置。

在 DHCP 环境中可将 IP 地址设置为字符串 dhcp，此时应将网关留空。在 IPv6 网络中，如需启用无状态自动配置，将 IP 设置为字符串 auto 即可。

如未设置 IPv4 或 IPv6 地址，Cloud-Init 默认将使用 IPv4 的 dhcp。

gw=<GatewayIPv4>

IPv4 的默认网关

◆ 注意

要配合使用选项：ip

gw6=<GatewayIPv6>

IPv6 的默认网关

◆ 注意

要配合使用选项：ip6

ip=<IPv4Format/CIDR> (default = dhcp)

IPv4 地址，采用 CIDR 格式。

ip=<IPv6Format/CIDR> (default = dhcp)

IPv6 地址，采用 CIDR 格式。

- ivshmem: size=<integer> [,name=<string>]

内部虚拟机共享内存。可实现虚拟机之间、主机虚拟机之间的直接通信。

name=<string>

设备文件名称。会自动添加前缀 pve-shm-。默认为虚拟机 VMID，并将在虚拟机停止后自动删除。

size=<integer> (1 - N)

文件大小，单位 MB。

- keyboard：<da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | fr-ca | fr-ch | hu | is | it | ja | lt | mk | nl | no | pl | pt | pt-br | sl | sv | tr>

  键盘布局设置，用于 vnc 服务器。默认采用/etc/pve/datacenter.conf 中的设置值。
- kvm：<boolean> (default = 1)

  启用/禁用 KVM 硬件虚拟化。
- localtime：<boolean>

  设置虚拟机时间是否采用服务器本地时间。如虚拟机操作系统类型为 Microsoft OS，则默认启用该项。
- lock: <backup | clone | create | migrate | rollback | snapshot | snapshot-delete | suspended | suspending>

  锁定/解锁虚拟机。
- machine :(pc|pc(-i440fx)?-\d+\.\d+(\.pxe)?|q35|pc-q35-\d+\.\d+(\.pxe)?|virt(?:-\d+\.\d+)?)

  设置 Qemu 虚拟机类型。
- memory：<integer> (16 -N) (default = 512)

  设置虚拟机内存容量，单位为 MB。启用 balloon 驱动时，该值为最大可用内存值。
- migrate_downtime：<number> (0 -N) (default = 0.1)

  设置虚拟机在线迁移时最大停机时间（单位为秒）。
- migrate_speed：<integer> (0 -N) (default = 0)

  设置虚拟机迁移时最大数据传输速度（单位为 MB/s）。设为 0 表示不限速。
- name：<string>

  设置虚拟机名称。仅用于 WebGUI 界面。
- nameserver: <string>

  Cloud-Init：为容器设置 DNS 服务器 IP 地址。如未设置 searchdomian 及 nameserver，将自动采用服务器主机设置创建有关配置。
- net[n]：[model=]<enum> [,bridge=<bridge>] [,firewall=<1|0>] [,link_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX:XX>] [,queues=<integer>] [,rate=<number>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,<model>=<macaddr>]

  设置虚拟网络设备。
  bridge = <bridge>
  虚拟网络设备桥接的虚拟交换机。Proxmox VE 默认创建虚拟交换机名为 vmbr0。
  如未指定虚拟交换机，Proxmox VE 将创建 KVM 网络设备（NAT），并提供 DHCP 和 DNS 服务。具体地址如下：
  10.0.2.2 Gateway
  10.0.2.3 DNS Server
  10.0.2.4 SMB Server
  其中 DHCP 服务器将从 10.0.2.15 开始分配 IP 地址
  firewall = <boolean>
  设置虚拟网络设备是否被防火墙保护。

link_down = <boolean>
设置虚拟网络设备是否可以被断开连接（类似于拔掉网线）。
macaddr = <XX:XX:XX:XX:XX:XX>
MAC 地址。
model = <e1000 | e1000-82540em | e1000-82544gc | e1000-82545em |
i82551 | i82557b | i82559er | ne2k_isa | ne2k_pci | pcnet |
rtl8139 | virtio | vmxnet3>
虚拟网卡类型。其中 virtio 性能最好。如虚拟机不支持 virtio，最好使用 e1000。
queues = <integer> (0 -16)
设置虚拟网卡的包队列数量。
rate = <number> (0 -N)
虚拟网卡最大传输速度，单位为 Mb/s，值为浮点数。
tag = <integer> (1 -4094)
虚拟网卡在数据包上自动标记的 VLAN 号。
trunks = <vlanid[;vlanid...]>
虚拟网卡所连接的 VLAN 号。

- numa：<boolean> (default = 0)

  启用/禁用 NUMA。

- numa[n]：cpus=<id[-id];...> [,hostnodes=<id[-id];...>] [,memory=<number>] [,policy=<preferred|bind|interleave>]

  NUMA 拓扑结构。
  cpus = <id[-id];...>
  当前 NUMA 节点上的 CPU 列表。
  hostnodes = <id[-id];...>
  所采用的主机 NUMA 节点。
  memory = <number>
  NUMA 节点所提供的内存容量。
  policy = <bind | interleave | preferred>
  NUMA 分配策略。

- onboot：<boolean> (default = 0)

  设置虚拟机是否在物理服务器启动时自动启动。

- ostype：<l24 | l26 | other | solaris | w2k | w2k3 | w2k8 | win10 |win7 | win8 | wvista | wxp>

  虚拟机操作系统类型。用于启用针对操作系统的优化和功能特性。可选值如下：

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 8/2012/2012r2
win10	Microsoft Windows 10/2016

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+

  将物理主机并口设备映射给虚拟机(n 的值为 0-2)。

☒ 注意

该属性允许虚拟机直接访问物理主机硬件。启用后将不能再进行虚拟机迁移操作，因此使用时务必小心。

☒ 警告

该特性仍处于试验阶段，有用户报告该属性会导致故障和问题。

- protection：<boolean> (default = 0)

  设置虚拟机保护标识。启用后将禁止删除虚拟机或虚拟机硬盘。
- reboot：<boolean> (default = 1)

  允许虚拟机重启。设为 0 后，虚拟机重启时将自动关闭。
- sata[n]: [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>][,bps=<bps>] [,bps_max_length=<seconds>] [,bps_rd=<bps>][,bps_rd_max_length=<seconds>] [,bps_wr=<bps>] [,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>][,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>] [,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>][,iops_wr=<iops>] [,iops_wr_max=<iops>] [,iops_wr_max_length=<seconds>] [,mbps=<mbps>] [,mbps_max=<mbps>] [,mbps_rd=<mbps>][,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>][,media=<cdrom|disk>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,secs=<integer>] [,serial=<serial>][,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>]

  配置 SATA 类型虚拟硬盘或光驱（n 的值为 0-5）。

  aio = <native | threads>
  指定 AIO 类型。
  backup = <boolean>
  设置虚拟硬盘在进行虚拟机备份时是否被纳入备份范围。
  bps = <bps>
  最大读写操作速度，单位为字节/秒。
  bps_max_length = <seconds>
  突发读写操作最大时间长度，单位为秒。
  bps_rd = <bps>
  最大读操作速度，单位为字节/秒。

bps_rd_max_length = <seconds>
突发读操作最大时间长度，单位为秒。
bps_wr = <bps>
最大写操作速度，单位为字节/秒。
bps_wr_max_length = <seconds>
突发写操作最大时间长度，单位为秒。
cache = <directsync | none | unsafe | writeback | writethrough>
虚拟硬盘缓存工作模式。
cyls = <integer>
强制指定虚拟硬盘物理几何参数中的 cylinder 值。
detect_zeroes = <boolean>
设置是否检测并优化零写入操作。
discard = <ignore | on>
设置是否向下层存储服务传递 discard/trim 操作请求。
file = <volume>
IDE 虚拟硬盘所基于的存储服务卷名称。
format = <cloop | cow | qcow | qcow2 | qed | raw | vmdk>
IDE 虚拟硬盘所采用的存储格式。
heads = <integer>
强制指定虚拟硬盘物理几何参数中的 head 值。
iops = <iops>
最大读写 I/O 速度，单位为个/秒。
iops_max = <iops>
最大无限制读写 I/O 速度，单位为个/秒。
iops_max_length = <seconds>
突发读写操作最大时间长度，单位为秒。
iops_rd = <iops>
最大读 I/O 速度，单位为个/秒。
iops_rd_max = <iops>
最大无限制读 I/O 速度，单位为个/秒。
iops_rd_max_length = <seconds>
突发读操作最大时间长度，单位为秒。
iops_wr = <iops>
最大写 I/O 速度，单位为个/秒。
iops_wr_max = <iops>
最大无限制写 I/O 速度，单位为个/秒。
iops_wr_max_length = <seconds>
突发写操作最大时间长度，单位为秒。
mbps = <mbps>
最大读写操作速度，单位为 MB/秒。
mbps_max = <mbps>
最大无限制读写操作速度，单位为 MB/秒。
mbps_rd = <mbps>
最大读操作速度，单位为 MB/秒。

mbps_rd_max = <mbps>
最大无限制读操作速度，单位为 MB/秒。
mbps_wr = <mbps>
最大写操作速度，单位为 MB/秒。
mbps_wr_max = <mbps>
最大无限制写操作速度，单位为 MB/秒。
media = <cdrom | disk> (default = disk)
虚拟硬盘驱动器介质类型。
replicate=<boolean> (default = 1)
磁盘是否被调度复制。
rerror = <ignore | report | stop>
读错误处理方式。
secs = <integer>
强制指定虚拟硬盘物理几何参数中的 sector 值。
serial = <serial>
虚拟硬盘的序列号，基于 url 编码格式，最大 20 字节。
shared=<boolean> (default = 0)
将本地管理卷标记为所有节点均可用。

☒ 警告

该选项并不自动共享卷，只是假定该卷已经被共享。

size = <DiskSize>
虚拟硬盘容量。仅供显示使用，并不能影响实际容量大小。
snapshot = <boolean>
Qemu 快照功能控制选项。设置后，对磁盘的改写会被当成临时的，并在虚拟机重启后全部丢弃。
ssd=<boolean>
设置虚拟磁盘连接到虚拟机的方式，SSD 或硬盘。
trans = <auto | lba | none>
设置虚拟硬盘几何参数地址 bios 解释模式。
werror = <enospc | ignore | report | stop>
写错误处理方式。
wwn = <wwn>
驱动器的唯一名称，使用 16 字节 hex 字符串表示，前缀为 0x。

- scsi[n]: [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>][,bps=<bps>] [,bps_max_length=<seconds>] [,bps_rd=<bps>][,bps_rd_max_length=<seconds>] [,bps_wr=<bps>] [,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>][,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>] [,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>][,iops_wr=<iops>] [,iops_wr_max=<iops>]

[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]
[,mbps_max=<mbps>][,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>],mbps_wr_max=<mbps>] [,media=<cdrom|disk>]
[,queues=<integer>][,replicate=<1|0>] [,rerror=<ignore|report|stop>]
[,scsiblock=<1|0>] [,secs=<integer>] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>] [,trans=<none|lba|auto>]
[,werror=<enum>] [,wwn=<wwn>]

配置 SCSI 类型虚拟硬盘或光驱（n 的值为 0-13）。

aio = <native | threads>

指定 AIO 类型。

backup = <boolean>

设置虚拟硬盘在进行虚拟机备份时是否被纳入备份范围。

bps = <bps>

最大读写操作速度，单位为字节/秒。

bps_max_length = <seconds>

突发读写操作最大时间长度，单位为秒。

bps_rd = <bps>

最大读操作速度，单位为字节/秒。

bps_rd_max_length = <seconds>

突发读操作最大时间长度，单位为秒。

bps_wr = <bps>

最大写操作速度，单位为字节/秒。

bps_wr_max_length = <seconds>

突发写操作最大时间长度，单位为秒。

cache = <directsync | none | unsafe | writeback | writethrough>

虚拟硬盘缓存工作模式。

cyls = <integer>

强制指定虚拟硬盘物理几何参数中的 cylinder 值。

detect_zeroes = <boolean>

设置是否检测并优化零写入操作。

discard = <ignore | on>

设置是否向下层存储服务传递 discard/trim 操作请求。

file = <volume>

IDE 虚拟硬盘所基于的存储服务卷名称。

format = <cloop | cow | qcow | qcow2 | qed | raw | vmdk>

IDE 虚拟硬盘所采用的存储格式。

heads = <integer>

强制指定虚拟硬盘物理几何参数中的 head 值。

iops = <iops>

最大读写 I/O 速度，单位为个/秒。

iops_max = <iops>

最大无限制读写 I/O 速度，单位为个/秒。

iops_max_length = <seconds>

突发读写操作最大时间长度，单位为秒。

iops_rd = <iops>

最大读 I/O 速度，单位为个/秒。
iops_rd_max = <iops>
最大无限制读 I/O 速度，单位为个/秒。
iops_rd_max_length = <seconds>
突发读操作最大时间长度，单位为秒。
iops_wr = <iops>
最大写 I/O 速度，单位为个/秒。
iops_wr_max = <iops>
最大无限制写 I/O 速度，单位为个/秒。
iops_wr_max_length = <seconds>
突发写操作最大时间长度，单位为秒。
iothread = <boolean>
设置是否启用 iothreads。
mbps = <mbps>
最大读写操作速度，单位为 MB/秒。
mbps_max = <mbps>
最大无限制读写操作速度，单位为 MB/秒。
mbps_rd = <mbps>
最大读操作速度，单位为 MB/秒。
mbps_rd_max = <mbps>
最大无限制读操作速度，单位为 MB/秒。
mbps_wr = <mbps>
最大写操作速度，单位为 MB/秒。
mbps_wr_max = <mbps>
最大无限制写操作速度，单位为 MB/秒。
media = <cdrom | disk> (default = disk)
虚拟硬盘驱动器介质类型。
queues = <integer> (2 -N)
设置队列数量。
replicate=<boolean> (default = 1)
磁盘是否被调度复制。
rerror = <ignore | report | stop>
读错误处理方式。
scsiblock=<boolean> (default = 0)
是否将 scsi-block 用于主机块设备直通

☒ 警告

在主机内存较低且内存碎片化严重时可能导致 I/O 错误。

secs = <integer>
强制指定虚拟硬盘物理几何参数中的 sector 值。
serial = <serial>
虚拟硬盘的序列号，基于 url 编码格式，最大 20 字节。

shared=<boolean> (default = 0)
将本地管理卷标记为所有节点均可用。

☒ 警告

该选项并不自动共享卷，只是假定该卷已经被共享。

size = <DiskSize>
虚拟硬盘容量。仅供显示使用，并不能影响实际容量大小。
snapshot = <boolean>
Qemu 快照功能控制选项。设置后，对磁盘的改写会被当成临时的，并在虚拟机重启后全部丢弃。
ssd=<boolean>
设置虚拟磁盘连接到虚拟机的方式，SSD 或硬盘。
trans = <auto | lba | none>
设置虚拟硬盘几何参数地址 bios 解释模式。
werror = <enospc | ignore | report | stop>
写错误处理方式。
wwn = <wwn>
驱动器的唯一名称，使用 16 字节 hex 字符串表示，前缀为 0x。

- scsihw : <lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci | virtio-scsi-single>

(default = lsi)
设置 SCSI 控制器类型。

- searchdomain: <string>

Cloud-Init：为容器设置 DNS 搜索域。如未设置 searchdomian 及 nameserver，将自动采用服务器主机设置创建有关配置。

- serial[n] : (/dev/.+|socket)

在虚拟机内创建串口设备（n 的值为 0-3），并直通到物理服务器的串口设备（如 /dev/ttyS0）或主机上创建的 unix socket（可使用 qm terminal 打开终端连接）。

☒ 注意

该属性允许虚拟机直接访问物理主机硬件。启用后将不能再进行虚拟机迁移操作，因此使用时务必小心。

☒ 警告

该特性仍处于试验阶段，有用户报告该属性会导致故障和问题。

- shares : <integer> (0 -50000) (default = 1000)

用于 auto-ballooning 的共享内存容量。设置的值越大，虚拟机得到的内存越多。具体分配到的内存由当前虚拟机的权重和所有其他虚拟机权重之和的比例决定。设置为 0 表示禁用 auto-ballooning。Pvestatd 会自动执行 auto-ballooning。

- smbios1： [base64=<1|0>] [,family=<Base64 encoded string>][,manufacturer=<Base64 encoded string>] [,product=<Base64 encoded string>] [,serial=<Base64 encoded string>] [,sku=<Base64 encoded string>] [,uuid=<UUID>] [,version=<Base64 encoded string>]

  设置 SMBIOS 的类型 1 字段。
  base64=<boolean>
  用于设置 SMBIOS 是否采用 base64 编码的参数值。
  family = <Base64 encoded string>
  设置 SMBIOS1 家族名称。
  manufacturer = <Base64 encoded string>
  设置 SMBIOS1 厂商名。
  product = <Base64 encoded string>
  设置 SMBIOS1 产品 ID。
  serial = <Base64 encoded string>
  设置 SMBIOS1 序列号。
  sku = <Base64 encoded string>
  设置 SMBIOS1 的 SKU 字符串。
  uuid = <UUID>
  设置 SMBIOS1 的 UUID。
  version = <Base64 encoded string>
  设置 SMBIOS1 版本。
- smp : <integer> (1 -N) (default = 1)

  设置 CPU 数量。请使用选项-sockets 代替。
- sockets : <integer> (1 -N) (default = 1)

  设置 CPU 的 sockets 数量。
- sshkeys: <string>

  Cloud-Init：设置 SSH 公钥（每行设置一个 key，OpenSSH 格式）。
- startdate : (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (default = now)

  设置系统时钟的初始时间。日期格式示例为：now 或 2006-06-17T16:01:21 或 2006-06-17。
- startup : `[[order=]\d+] [,up=\d+] [,down=\d+] `

  开机或关机操作。参数 order 是一个非负整数值，用于指定虚拟机开机启动顺序。关机顺序和开机顺序相反。此外还可以设置开机或关机的延迟时间，也就是当前虚拟机启动或关机后下一个虚拟机开机或关机的等待时间。
- tablet : <boolean> (default = 1)

  启用/禁用 USB 指针设备。该设备一般用来允许在 VNC 中使用鼠标的绝对坐标。否则鼠标将和 VNC 终端内的位置不同步。如果你在一台主机上运行了很多通过 VNC 终端访问的虚拟机，可以考虑禁用该参数以节省不必要的上下文切换。该项在 spice 终端下默认是禁用的（-vga=gxl）。
- tdf : <boolean> (default = 0)

  启用/禁用时间漂移修正。

- template：<boolean> (default = 0)

  启用/禁用模板。
- unused[n]：<string>

  用于标识未使用的存储卷。该参数仅供内部使用，不要手工编辑该参数。
- usb[n]：[host=]<HOSTUSBDEVICE|spice> [,usb3=<1|0>]

  设置 USB 设备（n 的值为 0 到 4）。
  host = <HOSTUSBDEVICE|spice>
  主机 USB 设备或端口或值 spice。HOSTUSBDEVICE 的配置语法为：
  'bus-port(.port)*'（十进制数）或
  'vendor_id:product_id'（十六进制数）或
  'spice'
  可使用命令 lsusb –t 列出当前的 usb 设备。

☒ 注意

该属性允许虚拟机直接访问物理主机硬件。启用后将不能再进行虚拟机迁移操作，因此使用时务必小心。

  值 spice 用于设置 usb 设备到 spice 的重定向。
  usb3 = <boolean> (default = 0)
  标识是否为 USB3 设备或端口（该参数在 spice 重定向下不稳定，将被忽略）。
- vcpus：<integer> (1 -N) (default = 0)

  可热插拔的虚拟 cpu 数量。
- vga: [[type=]<enum>] [,memory=<integer>]

  设置 VGA 硬件。如果使用高分辨率显示器（>=1280x1024x16），应该设置更高显存。QEMU2.9 之后，除部分 Windows 操作系统（XP 及更旧版本）使用 cirrus 以外，其他操作系统默认 VGA 显示器类型为 std。设置为 qxl 将启用 SPICE 显示服务器。对于 Win 系列操作系统，可以设置多个独立显示器，对于 Linux 系统可以自行添加显示器。也可以设置不使用图形显示卡，而使用串口设备作为终端。
  memory=<integer> (4 - 512)
  设置 VGA 显存大小。对串口终端无效。
  type=<cirrus | none | qxl | qxl2 | qxl3 | qxl4 | serial0 | serial1 | serial2 | serial3 | std | virtio | vmware> (default = std)
  设置 VGA 类型。
- virtio[n]: [file=]<volume> [,aio=<native|threads>] [,backup=<1|0>][,bps=<bps>] [,bps_max_length=<seconds>] [,bps_rd=<bps>][,bps_rd_max_length=<seconds>] [,bps_wr=<bps>] [,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect_zeroes=<1|0>][,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>]

[,iops_max_length=<seconds>] [,iops_rd=<iops>] [,iops_rd_max=<iops>]
[,iops_rd_max_length=<seconds>][,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]
[,mbps_max=<mbps>][,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>][,mbps_wr_max=<mbps>][,media=<cdrom|disk>]
[,replicate=<1|0>][,rerror=<ignore|report|stop>] [,secs=<integer>]
[,serial=<serial>][,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>]
[,trans=<none|lba|auto>] [,werror=<enum>]

配置 VIRTIO 类型虚拟硬盘（n 的值为 0-15）。

aio = <native | threads>

指定 AIO 类型。

backup = <boolean>

设置虚拟硬盘在进行虚拟机备份时是否被纳入备份范围。

bps = <bps>

最大读写操作速度，单位为字节/秒。

bps_max_length = <seconds>

突发读写操作最大时间长度，单位为秒。

bps_rd = <bps>

最大读操作速度，单位为字节/秒。

bps_rd_max_length = <seconds>

突发读操作最大时间长度，单位为秒。

bps_wr = <bps>

最大写操作速度，单位为字节/秒。

bps_wr_max_length = <seconds>

突发写操作最大时间长度，单位为秒。

cache = <directsync | none | unsafe | writeback | writethrough>

虚拟硬盘缓存工作模式。

cyls = <integer>

强制指定虚拟硬盘物理几何参数中的 cylinder 值。

detect_zeroes = <boolean>

设置是否检测并优化零写入操作。

discard = <ignore | on>

设置是否向下层存储服务传递 discard/trim 操作请求。

file = <volume>

IDE 虚拟硬盘所基于的存储服务卷名称。

format = <cloop | cow | qcow | qcow2 | qed | raw | vmdk>

IDE 虚拟硬盘所采用的存储格式。

heads = <integer>

强制指定虚拟硬盘物理几何参数中的 head 值。

iops = <iops>

最大读写 I/O 速度，单位为个/秒。

iops_max = <iops>

最大无限制读写 I/O 速度，单位为个/秒。

iops_max_length = <seconds>

突发读写操作最大时间长度，单位为秒。

iops_rd = <iops>
最大读 I/O 速度，单位为个/秒。
iops_rd_max = <iops>
最大无限制读 I/O 速度，单位为个/秒。
iops_rd_max_length = <seconds>
突发读操作最大时间长度，单位为秒。
iops_wr = <iops>
最大写 I/O 速度，单位为个/秒。
iops_wr_max = <iops>
最大无限制写 I/O 速度，单位为个/秒。
iops_wr_max_length = <seconds>
突发写操作最大时间长度，单位为秒。
iothread = <boolean>
设置是否启用 iothreads。
mbps = <mbps>
最大读写操作速度，单位为 MB/秒。
mbps_max = <mbps>
最大无限制读写操作速度，单位为 MB/秒。
mbps_rd = <mbps>
最大读操作速度，单位为 MB/秒。
mbps_rd_max = <mbps>
最大无限制读操作速度，单位为 MB/秒。
mbps_wr = <mbps>
最大写操作速度，单位为 MB/秒。
mbps_wr_max = <mbps>
最大无限制写操作速度，单位为 MB/秒。
media = <cdrom | disk> (default = disk)
虚拟硬盘驱动器介质类型。
replicate=<boolean> (default = 1)
磁盘是否被调度复制。
rerror = <ignore | report | stop>
读错误处理方式。
secs = <integer>
强制指定虚拟硬盘物理几何参数中的 sector 值。
serial = <serial>
虚拟硬盘的序列号，基于 url 编码格式，最大 20 字节。
shared=<boolean> (default = 0)
将本地管理卷标记为所有节点均可用。

☒ 警告

该选项并不自动共享卷，只是假定该卷已经被共享。

size = <DiskSize>
虚拟硬盘容量。仅供显示使用，并不能影响实际容量大小。

snapshot = <boolean>

Qemu 快照功能控制选项。设置后，对磁盘的改写会被当成临时的，并在虚拟机重启后全部丢弃。

trans = <auto | lba | none>

设置虚拟硬盘几何参数地址 bios 解释模式。

werror = <enospc | ignore | report | stop>

写错误处理方式。

- vmgenid: <UUID> (default = 1 (autogenerated))

  虚拟机生成 ID（vmgenid）设备是一个可被客户机操作系统识别的 128 位整数标识符。当虚拟机配置发生改变时（例如执行快照或从模板恢复导致的变化），可凭此通知客户机操作系统。客户机操作系统接到通知后，就能做出合理响应，如将分布式数据库副本标记为脏，重新初始化随机数生成器等等。注意，手工修改配置文件并不能自动重新创建虚拟机生成 ID，必须通过调用 API/CLI 提供的创建或更新接口才有效。

- vmstatestorage: <string>

  虚拟机状态卷/文件的默认存储。

- watchdog：[[model=]<i6300esb|ib700>] [,action=<enum>]

  创建虚拟看门狗设备。一旦启用（由虚拟机），虚拟机必须定期重置看门狗，否则虚拟机将自动重启（或执行指定的操作）。

  action = <debug | none | pause | poweroff | reset | shutdown>

  看门狗超时后所要进行的操作。

  model = <i6300esb | ib700> (default = i6300esb)

  虚拟看门狗类型。

10.12 锁

在线迁移，创建快照和备份（vzdump），Proxmox VE 都会对虚拟机加锁，以防止对虚拟机不恰当的并发操作。有时，你需要手工移除锁（例如，意外断电）。命令如下：

qm unlock <vmid>

☒ 警告

执行该操作前，务必确保设置锁的操作已经停止运行。

第 11 章 Proxmox 容器管理工具

与基于全虚拟化的虚拟机技术相比，容器是一种轻量级的虚拟化技术。容器并不对操作系统进行虚拟化，而是直接调用其所处主机的操作系统。这意味着所有的容器都共享同一个主机操作系统内核，并直接访问主机的各类资源。
由于容器无需虚拟化操作系统内核，因此能有效节约 CPU 和内存资源，从而赋予容器技术相当的优势。一般情况下，容器运行时的资源消耗接近于 0，完全可以忽略不计。但是，容器技术也有不可忽视的弱点：

- 容器内只能运行基于 Linux 的操作系统，比如你无法在容器内运行 FreeBSD 或 MS Windows 系统。
- 出于安全性的考虑，对主机资源的访问需要被有效控制。通常通过 AppArmor，SecComp 过滤器或 Linux 内核的其他组件来实现。这意味着在容器内无法调用一些 Linux 内核调用。

Proxmox VE 使用 LXC 作为底层容器技术。我们把 LXC 当作底层库来调用，并利用了其众多功能特性。直接使用 LXC 相关工具的难度太高，我们将有关工具进行了包装，以“Proxmox 容器管理工具”的形式提供给用户使用，该工具的名字是 pct。
容器管理工具 pct 和 Proxmox VE 紧密集成在一起，不仅能够感知 Proxmox VE 集群环境，而且能够象虚拟机那样直接利用 Proxmox VE 的网络资源和存储资源。你甚至可以在容器上配置使用 Proxmox VE 防火墙和 HA 高可用性。
我们的主要目标是提供一个和虚拟机一样的容器运行环境，同时能避免不必要的代价。我们称之为“系统容器”。

➢ 注意

如果你想运行微容器（如 docker，rkt 等），最好是在虚拟机里面运行它。

11.1 技术概览

- LXC（https://linuxcontainers.org）
- 集成在 Proxmxox VE 图形界面中（GUI）
- 简单易用的命令行工具 pct

- 支持通过 Proxmox VE REST API 调用
- 通过 lxcfs 提供容器化的/proc 文件系统
- 基于 AppArmor/Seccomp 的安全性增强
- 基于 CRIU 的在线迁移（计划中）
- 基于主流 Linux 内核
- 基于镜像的部署（模板）
- 可直接使用 Proxmox VE 存储服务
- 基于主机的容器配置（网络、DNS、存储等）

11.2 安全性分析

由于容器直接使用主机 Linux 内核，所以恶意用户可利用的攻击面非常宽泛。如果你计划向不可信的用户提供容器服务，必须认真考虑该问题。一般来说，基于全虚拟化的虚拟机能够达到更好的隔离效果。
好消息是，LXC 利用了 Linux 内核的众多安全特性，例如 AppArmor、CGroups 以及 PID 和用户 namespaces，这大大改善了容器的使用安全性。

11.3 用户操作系统配置

我们通常会尝试检测容器中的操作系统类型，然后修改容器中的一些文件，以确保容器正常工作。以下是我们在容器启动时的例行操作清单：

设置/etc/hostname
　　设置容器名称
修改/etc/hosts
　　允许查找容器主机名
网络配置
　　向容器传递完整的网络配置信息
配置 DNS
　　向容器传递 DNS 服务器配置信息
调整 init 系统初始化服务
　　例如，修改 getty 进程数量
设置 root 口令

创建新容器时，修改 root 口令

重新生成 ssh_host_keys

以确保每个容器的 key 都不重复

随机化 crontab

以确保各容器的 cron 调度任务不会同时启动

Proxmox VE 会用如下注释行将修改内容标识出来

--- BEGIN PVE ---

<data>

--- END PVE ---

以上标识符会插入相关文件的合适位置。如果配置文件中已经有标识符，Proxmox VE 会更新相关配置，并不再修改原标识符位置。

可以在配置文件相同路径下创建一个.pve-ignore 文件，避免 Proxmox VE 修改该配置文件。例如，只要/etc/.pve-ignore.hosts 文件存在，Proxmox VE 就不会修改/etc/hosts 文件配置内容。用户用如下命令创建空文件即可：

touch /etc/.pve-ignore.hosts

由于大部分配置修改都和操作系统类型相关，因此配置内容随 Linux 发行版和版本号改变而不同。你可以将 ostype 设置为 unmanaged 彻底禁止 Proxmox VE 修改配置。

操作系统类型检测通过测试容器内特定文件内容而实现：

Ubuntu

检查/etc/lsb-release (DISTRIB_ID=Ubuntu)

Debian

检测/etc/debian_version

Fedora

检测/etc/fedora-release

RedHat or CentOS

检测/etc/redhat-release

ArchLinux

检测/etc/arch-release

Alpine

检测/etc/alpine-release

Gentoo

检测/etc/gentoo-release

11.4 容器镜像

容器镜像，也称为“模板”或“应用程序”，实际上一个一个 tar 文件，其中包含了运行容器所需要的所有文件。你可以将它理解为容器的一个备份。和当前大多数容器管理工具一样，pct 使用容器镜像创建容器，例如：

pct create 999 local:vztmpl/debian-8.0-standard_8.0-1_amd64.tar.gz

Proxmox VE 自带了一组最基本的容器镜像，涵盖了大部分常见操作系统，你也可以运行命

令行工具 pveam（Proxmox VE Appliance Manager 的缩写）下载这些镜像。你还可以用该工具（或通过 WebGUI 界面）下载 TurnKey Linux 的容器镜像。
我们的镜像源有一组可用镜像，Proxmox VE 默认设置有一个 cron 任务每天定时下载该镜像列表。你也可以手工触发升级：
pveam update
然后可以查看可用镜像列表：
pveam available
该列表包含了很多镜像，你可以指定查看感兴趣的小节，例如可以指定查看 system 类的镜像：
列出可用镜像：
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

你需要将镜像下载到 Proxmox VE 的存储中之后才可以使用它创建容器。最直接的方法就是下载到 local 存储服务中。在 Proxmox VE 集群环境中，建议将镜像保存在一个共享存储服务中，以便所有节点访问。
pveam download local debian-8.0-standard_8.0-1_amd64.tar.gz
现在就可以使用该镜像创建容器了。也可以用如下命令查看 local 上已下载的所有镜像：
pveam list local
local:vztmpl/debian-8.0-standard_8.0-1_amd64.tar.gz 190.20MB

以上命令列出了保存镜像的 Proxmox VE 存储卷的完整标识符。其中包含有存储服务名称，其他 Proxmox VE 命令可以直接使用该存储服务名。例如可以用如下命令删除镜像：
pveam remove local:vztmpl/debian-8.0-standard_8.0-1_amd64.tar.gz

11.5 容器存储

传统容器的存储模型十分简陋，通常只支持一个挂载点，即根文件系统。此外，支持的文件系统类型也非常有限，例如 ext4 和 nfs，这大大限制了容器的使用。如果要挂载额外的存储，用户必须自行编写脚本文件，既复杂又容易出错。因此，我们在 Proxmox VE 中努力避免这些问题。
我们提供的基于 LXC 的容器模型在存储方面非常灵活。首先，你可以使用多个挂载点，从而可以根据应用类型的不同选择合适的存储。例如，你可以为根文件系统选用速度相对较慢（因此也较便宜）的存储，同时为数据库应用挂载第二个高速高性能分布式存储。进一步信

息可查看挂载点一节。
第二个重大改进是，你可以将 Proxmox VE 支持的任意类型的存储服务用于容器。也就是说，你可以将容器保存在本地 lvmthin 或 zfs 上，共享 iSCSI 存储或 ceph 分布式存储服务上。进一步，还可以利用存储服务的高级特性，比如快照和克隆。vzdump 也可以利用快照特性实现一致的容器备份。
最后，但也很重要的是，你可以在容器中直接使用服务器本地设备，或通过绑定挂载使用服务器本地目录。这样就可以在容器中以零性能损失访问本地存储服务。而绑定挂载也提供了一种在多个容器间共享数据的简便方式。

11.5.1 FUSE 挂载

☒ 警告

鉴于当前 Linux 内核的冻结子系统的问题，而以挂起或快照模式备份时需要将容器冻结，所以强烈反对在容器中使用 FUSE 挂载。

如果实在不能用其他挂载机制或存储技术替代 FUSE 挂载，万不得已时，仍然可以在 Proxmox 服务器创建 FUSE 挂载，并通过绑定挂载提供给容器使用。

11.5.2 容器内设置存储配额

在容器内设置存储配额能够有效限制每个用户可以使用的硬盘空间大小。目前，该功能仅在基于 ext4 文件系统的容器上可以使用，并且不能用于非特权容器。
激活 quota 选项后，挂载点会增加以下挂载参数项：srjquota=aquota.user,grpjquota=aquota.group,jqfmt=vfsv0
这些参数让你能够像在其他系统上一样使用存储配额功能。你可以用如下命令初始化/aquota.user 和/aquota.group 文件：

```
quotacheck -cmug /
quotaon /
```

然后可以通过 edquota 命令编辑配额。具体配置可以参考容器所采用的 Linux 发行版镜像自带的技术文档。

➢ 注意

你需要对每个挂载点执行以上命令，并用实际挂载点路径替代根路径 /。

11.5.3 容器内设置访问控制列表

容器内可以配置使用标准 Posix Access Control Lists。通过 ACLs 能够详尽地控制文件属主，其细致程度远胜传统的 user/group/others 模型。

11.5.4 备份容器挂载点

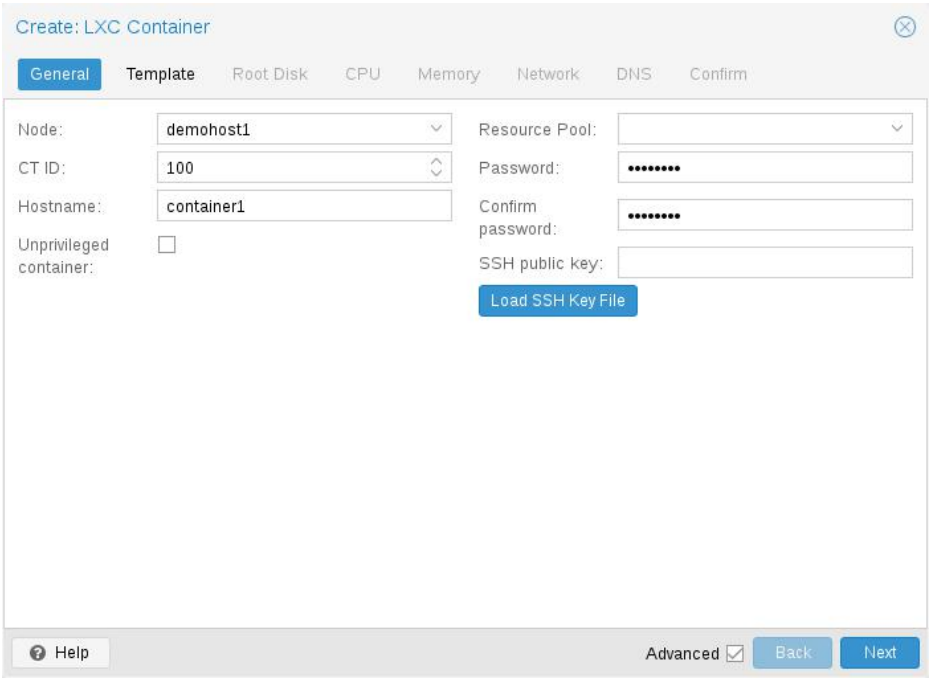
除了根磁盘以外，容器的其他挂载点默认不备份。可以设置挂载点的 Backup 参数，改变默认备份行为。

11.5.5 复制容器挂载点

默认情况下，在根磁盘被调度复制时，容器的其他挂载点也会被复制。如过其他挂载点不需要被复制，可以勾选挂载点的 Skip replication 选项。
在 Proxmox VE 5.0 中，调度复制只能用于 zfspool 类存储，所以当容器设置了调度复制，同时又挂在了其他类存储时，需要在相应挂载点设置 Skip replication。

11.6 容器设置项

11.6.1 通用设置项



容器通用设置项如下：

- Node：容器所处的物理服务器
- CT ID：Proxmox VE 用于标识容器的唯一序号。
- Hostname：容器的主机名。

- Resource Pool：逻辑上划分的一组容器和虚拟机。
- Password：容器的 root 口令。
- SSH Public Key：用于 SSH 连接登录 root 用户的公钥。
- Unprivileged container：该项目用于在容器创建阶段设置创建特权容器或非特权容器。

特权容器

增强容器安全性的常用手段有关闭不必要的功能，使用强制访问控制（AppArmor），SecComp 过滤器和命名空间 namespaces。LXC 工作组认为特权容器技术是不安全的，并且不打算为新发现的容器逃逸漏洞申报 CVE 编号和发布快速修复补丁。所以，特权容器仅限于在内部可信环境中使用，或在确保容器中没有不可信 root 权限进程时使用。

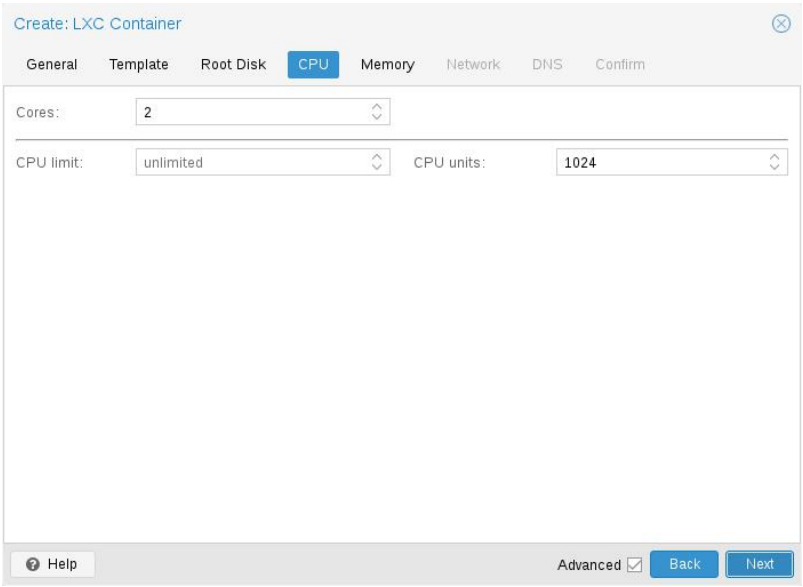
非特权容器

非特权容器采用了名为用户命名空间 user namespaces 的内核新特性。容器内 UID 为 0 的 root 用户被影射至外部的一个非特权用户。也就是说，在非特权容器中常见的安全问题（容器逃逸，资源滥用等）最终只能影响到外部的一个随机的非特权用户，并最终归结为一个一般的内核安全缺陷，而非 LXC 容器安全性问题。LXC 工作组认为非特权容器的设计是安全的。

➢ 注意

如果容器使用 systemd 作为系统初始化服务，请确保容器内的 systemd 版本高于或等于 220。

11.6.2 CPU



你可以通过 cores 参数项设置容器内可见的 CPU 数量。该参数项基于 Linux 的 cpuset cgroup（控制 group）实现。在 pvestatd 服务内有一个任务专门用于在 CPU 之间平衡容器工作负载。你可以用如下命令查看 CPU 分配情况：

```
# pct cpusets
---------------------
102: 6 7
105: 2 3 4 5
108: 0 1
---------------------
```

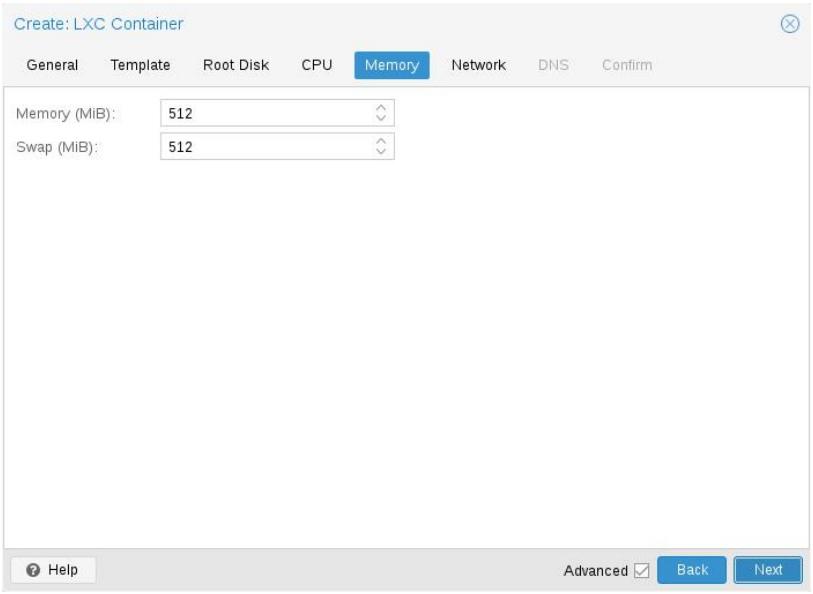
容器能够直接调用主机内核，所以容器内的所有任务进程都由主机的 CPU 调度器直接管理。Proxmox VE 默认使用 Linux CFS（完全公平调度器），并提供以下参数项可以进一步控制 CPU 分配。

cpulimit： 你可以设置该参数项进一步控制分配的 CPU 时间片。需要注意，该参数项类型为浮点数，因此你可以完美地实现这样的配置效果，即给容器分配 2 个 CPU 核心，同时限制容器总的 CPU 占用为 0.5 个 CPU 核心。具体如下：

```
cores: 2
cpulimit: 0.5
```

cpuunits： 该参数项是传递给内核调度器的一个相对权重值。参数值越大，容器得到的 CPU 时间越多。具体得到的 CPU 时间片由当前容器权重占全部容器权重总和的比重决定。该参数默认值为 1024，可以调大该参数以提高容器的优先权。

11.6.3 内存

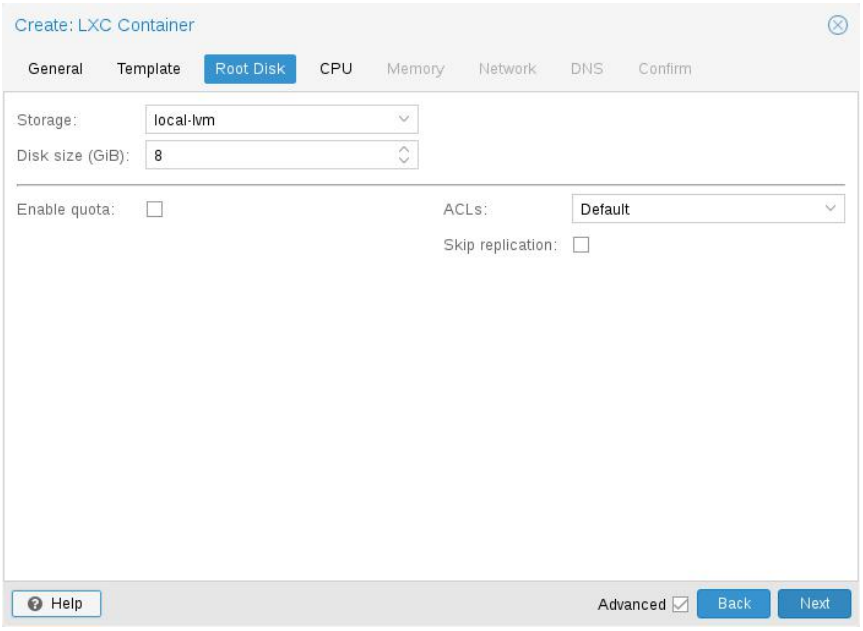


容器内存由 cgroup 内存控制器管理。

memory： 容器的内存总占用量上限。对应于 cgroup 的 memory.limit_in_bytes 参数项。

swap： 用于设置允许容器使用主机 swap 空间的大小。对应于 cgroup 的 memory.memsw.limit_in_bytes 参数项，cgroup 的参数实际上是内存和交换分区容量之和（memory+swap）。

11.6.4 挂载点



容器的根挂载点通过 rootfs 属性配置。除此之外，你还可以再配置 10 个挂载点，分别对应

于参数 mp0 到 mp9。具体设置项目如下：

- rootfs: [volume=]<volume>[,mountoptions=<opt[;opt...]>] [,acl=<1|0>] [,quota=<1|0>] [,ro=<1|0>][,shared=<1|0>] [,size=<DiskSize>]

  配置容器根文件系统存储卷。全部配置参数见后续内容。

- mp[n]: [volume=]<volume> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>][,mountoptions=<opt[;opt...]>][,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>][,size=<DiskSize>]

  配置容器附加挂载点存储卷。

acl=<boolean>
启用/禁用 acl。
backup=<boolean>
用于配置在备份容器时是否将挂载点纳入备份范围。（仅限于附加挂载点）
[,mountoptions=<opt[;opt...]>]
rootfs/mps 挂载点的附加参数
mp=<Path>
存储卷在容器内部的挂载点路径。

➢ 注意

出于安全性考虑，禁止含有文件链接。

quota=<boolean>
在容器内启用用户空间配额（对基于 zfs 子卷的存储卷无效）。
replicate=<boolean> (default = 1)
卷是否被可以被调度任务复制。
ro=<boolean>
用于标识只读挂载点。
shared=<boolean> (default = 0)
用于标识当前存储卷挂载点对所有节点可见。

☒ 警告

设置该参数不等于自动共享挂载点，而仅仅表示当前挂载点被假定已经共享。

size=<DiskSize>
存储卷容量（参数值只读）。
volume=<volume>
存储卷命令，即挂载到容器的设备或文件系统路径。

目前主要有 3 类不同的挂载：基于存储服务的挂载，绑定挂载，设备挂载。

容器 rootfs 典型配置示例

rootfs: thin1:base-100-disk-1,size=8G

基于存储服务的挂载

基于存储服务的挂载由 Proxmox VE 的存储子系统管理，一共有 3 种不同方式：

- 硬盘镜像：也就是内建了 ext4 文件系统的硬盘镜像。
- ZFS 存储卷：技术上类似于绑定挂载，但通过 Proxmox VE 存储子系统管理，并且支持容量扩充和快照功能。
- 目录：可以设置 size=0 禁止创建硬盘镜像，直接创建目录存储。

> ➢ 注意
>
> 可以 STORAGE_ID:SIZE_IN_GB 的形式在指定存储上创建指定大小的卷。例如，执行 pct set 100 –mp0 thin1:10,mp=/path/in/container 将在存储 thin1 上创建 10GB 大小的卷，并将卷 ID 替换为分配卷 ID。

绑定挂载

绑定挂载可以将 Proxmox VE 主机上的任意目录挂载到容器使用。可行的使用方法有：

- 在容器中访问主机目录
- 在容器中访问主机挂载的 USB 设备
- 在容器中访问主机挂载的 NFS 存储

绑定挂载并不由 Proxmox VE 存储子系统管理，因此你不能创建快照或在容器内启用配额管理。在非特权容器内，你可能会因为用户映射关系和不能配置 ACL 而遇到权限问题。

> ➢ 注意
>
> vzdump 将不会备份绑定挂载设备上的数据。

> ☒ 警告
>
> 出于安全性考虑，最好为绑定挂载创建专门的源目录路径，例如在/mnt/bindmounts 下创建的目录。永远不要将/，/var 或/etc 等系统目录直接绑定挂载给容器使用，否则将可能带来极大的安全风险。

> ➢ 注意
>
> 绑定挂载的源路径必须没有任何链接文件。

例如，要将主机目录/mnt/bindmounts/shared 挂载到 ID 为 100 的容器中的/shared 下，可在配置文件/etc/pve/lxc/100.conf 中增加一行配置信息 p0:/mnt/bindmounts/shared,mp=/shared。或者运行命令 pct set 100 -mp0 /mnt/bindmounts/shared,mp=/shared 也可以达到同样效果。

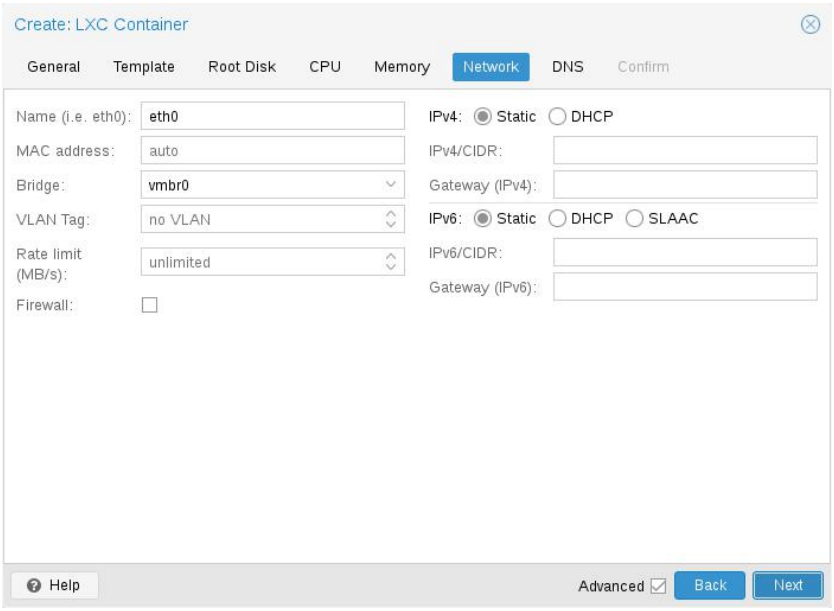
设备挂载

设备挂载可以将 Proxmox VE 上的块存储设备直接挂载到容器中使用。和绑定挂载类似，设备挂载也不由 Proxmox VE 存储子系统管理，但用户仍然可以配置使用 quota 和 acl 等功能。

> ➢ 注意
>
> 设备挂载仅在非常特殊的场景下才值得使用，大部分情况下，基于存储服务的挂载能提供和设备挂载几乎一样的功能和性能，同时还提供更多的功能特性。

> ➢ 注意
>
> vzdump 将不会备份设备挂载上的数据。

11.6.5 网络



单个容器最多支持配置 10 个虚拟网卡设备，其名称分别为 net0 到 net9，并支持以下配置参数项：

- net[n]: name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:XX>][,ip=<(IPv4/CIDR|dhcp|manual)>] [,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>] [,trunks=<vlanid;vlanid...]>] [,type=<veth>]

  为容器配置虚拟网卡设备。

  bridge=<bridge>

  虚拟网卡设备连接的虚拟交换机。

  firewall=<boolean>

设置是否在虚拟网卡上启用防火墙策略。
gw=<GatewayIPv4>
IPv4 通信协议的默认网关。
gw6=<GatewayIPv6>
IPv6 通信协议的默认网关。
hwaddr=<XX:XX:XX:XX:XX:XX>
虚拟网卡的 MAC 地址。
ip=<(IPv4/CIDR|dhcp|manual)>
IPv4 地址，以 CIDR 格式表示。
ip6=<(IPv6/CIDR|auto|dhcp|manual)>
IPv6 地址，以 CIDR 格式表示。
mtu=<integer> (64 -N)
虚拟网卡的最大传输单元。（lxc.network.mtu）
name=<string>
容器内可见的虚拟网卡名称。（lxc.network.name）
rate=<mbps>
虚拟网卡的最大传输速度。
tag=<integer> (1 -4094)
虚拟网卡的 VLAN 标签。
trunks=<vlanid[;vlanid...]>
虚拟网卡允许通过的 VLAN 号。
type=<veth>
虚拟网卡类型。

11.6.6 容器的自启动和自关闭

创建容器后，你也许会希望容器能够随主机自行启动。为此，你可以在 Web 界面的容器 Options 选项卡上选择 Start at boot，或用如下命令设置：
pct set <ctid> -onboot 1

启动和关闭顺序

如果要精细调整容器的启动顺序，可以使用以下参数：

- **Start/Shutdown order**：用于设置启动优先级。例如，设为 1 表示你希望容器第 1 个被启动。（我们采用了和启动顺序相反的关闭顺序，所以 Start order 设置为 1 的容器将最后被关闭）
- **Startup delay**：用于设置当前容器启动和继续启动下一个容器之间的时间间隔。例如，设置为 240 表示你希望当前容器启动 240 秒后再继续启动下一个容器。
- **Shutdown timeout**：用于设置发出关闭命令后 Proxmox VE 等待容器执行关闭操作的时间，单位为秒。该参数默认值为 60，也就是说 Proxmox VE 在发出关闭容器命令后，会等待 60 秒钟，如果容器不能在 60 秒内完成关机离线操作，Proxmox VE 将通知用户容器关闭操作失败。

请注意，未设置 Start/Shutdown order 参数的容器将始终在设置了这些参数的容器之后启动。并且这些参数仅能影响同一 Proxmox VE 主机上的容器启动顺序，其作用范围局限在单一服务器内部，而不是整个集群。

11.6.7 回调脚本

可以使用 hookscript 属性设置容器回调脚本。
pct set 100 -hookscript local:snippets/hookscript.pl
该脚本会在容器生命周期的多个阶段被调用。如需查看具体例子和相关文档，可以在 /usr/share/pve-docs/examples/guest-example-hookscript.pl 查看范例脚本。

11.7 备份和恢复

11.7.1 容器备份

可以使用 vzdump 命令备份容器。详细信息请参考 vzdump 的 man 手册。

11.7.2 容器备份恢复

可以用 pct restore 命令将 vzdump 生成的容器备份恢复出来。默认情况下，pct restore 将尝试尽可能按照备份文件中的配置信息恢复容器。但也可以在恢复命令中手工指定容器配置参数，以覆盖备份文件中的配置备份（详情可查看 pct 命令的 man 手册）。

➢ 注意

可运行命令 pvesm extractconfig 查看 vzdump 备份文件中的配置备份信息。

根据对挂载点处理方式的不同，一共有两种恢复模式：

“简单”恢复模式

如果在恢复命令中既没有指定 rootfs 参数也没有指定任何 mpX 参数，则按以下步骤从备份配置文件恢复挂载点配置信息：
1.从备份文件提取挂载点及相关配置项。
2.对于基于存储服务的挂载点，创建相应存储卷（在 storage 参数指定的存储服务上创建，如未设置则默认在 local 存储服务上创建）。
3.从备份文件中提取备份数据。
4.增加绑定挂载点和设备挂载点，并进一步恢复配置（仅限于 root 用户）。
基于 Web 界面的恢复操作采用的就是简单模式。

➢ 注意

鉴于绑定挂载点和设备挂载点中的数据永远不会被备份，因此最后一步中不会有任何实际数据被恢复，而仅仅是恢复挂载点配置信息。这种处理方法基于一个前提假设，即这两类挂载点中的数据已被其他机制备份（例如，同时绑定挂载到多个容器的 NFS 存储空间），或根本不需要备份。

“高级”恢复模式

如果指定 rootfs 参数（或者，指定任意 mpX 参数组合），恢复命令 pct restore 将自动进入高级恢复模式。高级恢复模式将完全忽略备份文件中保存的 rootfs 和 mpX 配置信息，转而采用命令行中指定的配置信息。
高级模式允许在恢复操作时灵活配置挂载点信息，例如：

- 为每个挂载点分别设置目标存储，存储卷容量及其他配置参数。
- 按照新指定的挂载点调整备份文件数据存储分布情况。
- 恢复到设备挂载点和（或）绑定挂载点（仅限于 root 用户）。

11.8 使用 pct 管理容器

Proxmox VE 使用 pct 命令管理容器。你可以用 pct 命令创建或销毁容器，也可以控制容器的运行（启动，关闭，迁移等）。你还可以用 pct 命令设置相关配置文件中的参数，例如网络配置或内存上限。

11.8.1 命令行示例

使用 Debian 模板创建一个容器（假定你已经通过 Web 界面下载了模板）

pct create 100 /var/lib/vz/template/cache/debian-8.0-standard_8.0-1_amd64.tar.gz
启动 100 号容器
pct start 100
通过 getty 启动登录控制台
pct console 100
进入 LXC 命名空间并使用 root 用户启动一个 shell
pct enter 100
显示容器配置
pct config 100
在容器运行的状态下增加名为 eth0 的虚拟网卡，同时设置桥接虚拟交换机 vmbr0，IP 地址和网关。
pct set 100 -net0 name=eth0,bridge=vmbr0,ip=192.168.15.147/24,gw=192.168.15.1
调整容器内存减少到 512MB
pct set 100 -memory 512

11.8.2 获取调试日志

在 pct start 无法启动某个容器时，运行 lxc-start 命令搜集调试日志输出有可能帮助排查故障原因（用容器 ID 替换如下命令中的 ID）：
lxc-start -n ID -F -l DEBUG -o /tmp/lxc-ID.log
该命令将尝试用前台模式启动容器，可以在另外一个控制台运行 pct shutdown ID 或 pct stop ID 停止容器。
收集到的日志信息保存在/tmp/lxc-ID.log 中。

- **注意**

如果你在最近一次运行 pct start 命令尝试启动容器后修改了容器配置，在执行 lxc-start 命令前，你应该至少再运行一次 pct start 命令，以更新容器 lxc-start 命令可用的配置。

11.9 迁移

在集群环境中，你可以用如下命令迁移容器
pct migrate <vmid> <target>
该命令只对关机离线的容器有效。如果容器使用了本地存储和挂载点，而迁移目标服务器配置了同名的存储服务和资源，迁移命令将自动通过网络把相关数据复制到目标服务器。
如果你想迁移在线状态的容器，唯一的方法是使用重启迁移。具体是在迁移命令中使用-restart 标识和可选参数-timeout。
重启迁移命令将关闭容器，并在指定时间后杀死容器（默认时间为 180 秒），然后按照离线迁移的步骤迁移容器，并在迁移完成后在目标节点重新启动容器。

11.10 容器配置文件

容器配置信息全部保存在/etc/pve/lxc/<CTID>.conf 文件中，其中<CTID>是容器的数字 ID。和/etc/pve 目录下的所有文件一样，容器配置文件也会被自动复制到集群的所有其他节点。

➢ 注意

小于 100 的 CTID 都被 Proxmox VE 内部保留使用。同一集群内不能有重复的 CTID。

容器配置文件示例

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

容器配置文件采用了简单的文本格式，可以用常见的编辑器（vi，nano 等）编辑修改。这也是日常进行少量配置调整的常用方法，但务必注意必须重启容器才能让新的配置生效。因此，更好的方法是使用 pct 命令修改配置，或通过 WebGUI 进行。Proxmox VE 能让大部分配置变更对运行中的容器即时生效。该功能称为“热插拔”，并且无须重启容器。

11.10.1 配置文件格式

容器配置文件采用了简单的冒号分隔的键/值格式。每一行的格式如下：

this is a comment
OPTION: value

空行将被自动忽略，以#字符开头的行将被当作注释处理，也会被自动忽略。
可以在配置文件中直接添加底层 LXC 风格的配置，例如：

lxc.init_cmd: /sbin/my_own_init

或

lxc.init_cmd = /sbin/my_own_init

这些配置将被直接传递给底层 LXC 管理工具。

11.10.2 快照

当你创建一个快照后，pct 将在原配置文件中创建一个独立小节保存快照创建时的配置。例如，创建名为“testsnapshot”的快照后，你的配置文件会类似于下面的例子：

创建快照后的配置文件示例

memory: 512

swap: 512
parent: testsnaphot
...
[testsnaphot]
memory: 512
swap: 512
snaptime: 1457170803
...
其中 parent 和 snaptime 是和快照相关的配置属性。属性 parent 用于保存快照之间的父/子关系。属性 snaptime 用于标识快照创建时间（Unix epoch）。

11.10.3 参数项

- arch: <amd64 | arm64 | armhf | i386> (default = amd64)

  操作系统架构类型。
- cmode: <console | shell | tty> (default = tty)

  控制台模式。默认情况下，控制台命令尝试打开到 tty 设备的连接。设置 cmode 为 console 后，将尝试连接到/dev/console 设备。设置 cmode 为 shell 后，将直接调用容器内的 shell（no login）。
- console: <boolean> (default = 1)

  挂接到容器的控制台设备（/dev/console）。
- cores: <integer> (1 -128)

  分配给容器的 CPU 核心数量。默认容器可以使用全部的核心。
- cpulimit: <number> (0 -128) (default = 0)

  CPU 分配限额。

➢ 注意

如计算机有 2 个 CPU，一共可分配的 CPU 时间为 2。设置为 0 表示无限制。

- cpuunits: <integer> (0 -500000) (default = 1024)

  分配给容器的 CPU 权重。该参数用于内核的公平调度器。参数值越大，容器能获得的 CPU 时间片越多。获得的 CPU 时间片具体由当前容器权重和所有其他容器权重总和的比值决定。

➢ 注意

可将该参数设为 0 以禁用公平调度器。

- description: <string>

容器描述信息。仅供 Web 界面显示使用。

- features: [fuse=<1|0>] [,keyctl=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>]

  设置容器可以使用的高级特性

  fuse=<boolean> (default = 0)

  在容器中启用 fuse 文件系统。注意，同时使用 fuse 和 freezer cgroup 可能导致 I/O 死锁。

  keyctl=<boolean> (default = 0)

  仅适用于非特权容器：允许调用 keyctl()系统调用。主要用于在容器内运行 docker。默认情况下，容器无法看到该系统调用，从而避免 systemd-networkd 因权限不足调用 keyctl()失败而导致的致命错误。因此，是否启用该参数主要取决于你在容器内运行 systemd-networkd 还是 docker。

  mount=<fstype;fstype;...>

  用于设置允许容器挂载指定文件系统。该参数指定了允许 mount 命令挂载的文件系统类型列表。需要注意的是，在容器内挂载其他文件系统会对危害安全性。例如，通过访问 loop 设备，可以在挂载文件系统时绕过设备 cgroup 组的 mknod 权限，挂载 NFS 文件系统有可能完全阻塞主机 I/O，并导致无法重启等等。

  nesting=<boolean> (default = 0)

  设置允许容器嵌套。最好在启用 ID 映射的非特权容器内使用。注意，该特性会将主机的 procfs 和 sysfs 暴露给容器。

- hookscript: <string>

  设置回调脚本。

- hostname: <string>

  容器的主机名。

- lock: <backup | create | disk | fstrim | migrate | mounted | rollback | snapshot | snapshot-delete>

  设置锁定/解锁容器。

- memory: <integer> (16 -N) (default = 512)

  分配给容器的内存容量。

- mp[n]: [volume=]<volume> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>][,mountoptions=<opt[;opt...]>][,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>][,size=<DiskSize>]

  给容器设置附加挂载点。

  acl=<boolean>

  设置启用或禁用 ACL。

  backup=<boolean>

  设置在备份容器时是否将挂载点纳入备份范围（仅对卷挂载点有效）。

  [,mountoptions=<opt[;opt...]>]

  rootfs/mps 挂载点的附加参数

  mp=<Path>

  容器内的挂载点路径。

- 注意

出于安全性考虑，禁止包含任何文件链接。

quota=<boolean>
启用容器内的用户配额功能（对基于 ZFS 子卷的挂载点无效）。
replicate=<boolean> (default = 1)
设置卷是否可以被调度任务复制。
ro=<boolean>
设置挂载点为只读。
shared=<boolean> (default = 0)
设置非卷挂载点为所有节点可共享。

☒ 警告

设置该参数不等于自动共享挂载点，而仅仅表示当前挂载点被假定已经共享。

size=<DiskSize>
挂载点存储卷容量（参数值只读）。
volume=<volume>
挂载到容器的卷、设备或目录。

- nameserver: <string>

  设置容器所使用的 DNS 服务器 IP 地址。如未指定 nameserver 和 searchdomain，将在创建容器时直接使用主机的相关配置。

- net[n]: name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:XX>][,ip=<(IPv4/CIDR|dhcp|manual)>] [,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>] [,trunks=<vlanid;vlanid...]>] [,type=<veth>]

  为容器配置虚拟网卡设备。
  bridge=<bridge>
  虚拟网卡设备连接的虚拟交换机。
  firewall=<boolean>
  设置是否在虚拟网卡上启用防火墙策略。
  gw=<GatewayIPv4>
  IPv4 通信协议的默认网关。
  gw6=<GatewayIPv6>
  IPv6 通信协议的默认网关。
  hwaddr=<XX:XX:XX:XX:XX:XX>
  虚拟网卡的 MAC 地址。
  ip=<(IPv4/CIDR|dhcp|manual)>
  IPv4 地址，以 CIDR 格式表示。
  ip6=<(IPv6/CIDR|auto|dhcp|manual)>
  IPv6 地址，以 CIDR 格式表示。
  mtu=<integer> (64 -N)
  虚拟网卡的最大传输单元。（lxc.network.mtu）

name=<string>
容器内可见的虚拟网卡名称。（lxc.network.name）
rate=<mbps>
虚拟网卡的最大传输速度。
tag=<integer> (1 -4094)
虚拟网卡的 VLAN 标签。
trunks=<vlanid[;vlanid...]>
虚拟网卡允许通过的 VLAN 号。
type=<veth>
虚拟网卡类型。

- onboot: <boolean> (default = 0)

  设置容器是否随主机自动启动。

- ostype: <alpine | archlinux | centos | debian | fedora | gentoo | opensuse | ubuntu | unmanaged>

  设置操作系统类型。供容器内部配置使用，并和/usr/share/lxc/config/<ostype>.common.conf 中的 lxc 启动脚本对应。设置为 unmanaged 表示跳过操作系统相关配置。

- protection: <boolean> (default = 0)

  设置容器的保护标志。设置后将禁止删除/变更容器及容器硬盘配置。

- rootfs: [volume=]<volume> [,acl=<1|0>][,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskSize>]

  为容器配置根文件系统卷。
  acl=<boolean>
  设置启用或禁用 ACL。
  [,mountoptions=<opt[;opt...]>]
  rootfs/mps 挂载点的附加参数
  quota=<boolean>
  在容器内启用用户空间配额（对基于 zfs 子卷的存储卷无效）。
  replicate=<boolean> (default = 1)
  设置卷是否可以被调度任务复制。
  ro=<boolean>
  用于标识只读挂载点。
  shared=<boolean> (default = 0)
  设置非卷挂载点为所有节点可共享。

☒ 警告

设置该参数不等于自动共享挂载点，而仅仅表示当前挂载点被假定已经共享。

size=<DiskSize>
挂载点存储卷容量（参数值只读）。
volume=<volume>

挂载到容器的卷、设备或目录。

- searchdomain: <string>

  设置容器的 DNS 搜索域。如未指定 nameserver 和 searchdomain，将在创建容器时直接使用主机的相关配置。

- startup: `[[order=]\d+] [,up=\d+] [,down=\d+] `

  启动和关闭行为设置。参数 order 为非负整数，用于定义启动顺序。关闭顺序和启动顺序相反。此外还可以设置启动延时秒数，以指定下一个虚拟机启动或关闭之前的时间间隔。

- swap: <integer> (0 -N) (default = 512)

  分配给容器的 SWAP 容量，单位为 MB。

- template: <boolean> (default = 0)

  启用/禁用模板。

- tty: <integer> (0 -6) (default = 2)

  指定容器可用的 tty 数量。

- unprivileged: <boolean> (default = 0)

  设置容器以非特权用户权限运行。（不要手工修改该配置）

- unused[n]: <string>

  标识未使用的存储卷。仅供 Proxmox VE 内部使用，不要手工修改该配置。

11.11 锁

容器迁移、快照创建和备份（vzdump）操作会设置容器锁，以避免不恰当的并发操作。某些情况下，你需要手工移除容器锁（例如，意外断电）。

pct unlock <CTID>

☒ 警告

执行该操作前，务必确保设置锁的操作已经停止运行。

第 12 章 Proxmox VE 防火墙

Proxmox VE 防火墙为你的 IT 基础设施提供了一种简单易用的防护手段。你既可以为集群内的所有主机设置防火墙策略，也可以为单个虚拟机和容器定义策略。防火墙宏，安全组，IP 集和别名等特性将大大简化策略配置管理。
尽管所有的防火墙策略都保存在集群文件系统，但基于 iptables 的防火墙服务在每个节点都是独立运行的，从而为虚拟机提供了完全隔离的防护。这套分布式部署的防火墙较传统防火墙提供了更高的带宽。
Proxmox VE 防火墙完全支持 IPv4 和 IPv6。IPv6 的支持是完全透明的，我们默认自动对两种协议通信同时进行过滤和检测。所以没有必要为 IPv6 专门建立并维护防火墙策略。

12.1 区域

Proxmox VE 防火墙将网络划分为不同区域

Host

流出/流入集群节点的网络通信

VM

流出/流入虚拟机的网络通信

对每个区域，你都可以对流入/流出流量定义防火墙策略。

12.2 配置文件

防火墙相关的配置文件全部保存在 Proxmox VE 集群文件系统中，所以能够自动在所有节点间同步复制，而防火墙管理服务 pve-firewall 将在防火墙策略改变后自动更新底层 iptables 策略。
你可以在 WebGUI 界面完成所有的防火墙配置（例如通过，数据中心→防火墙，或者通过，节点→防火墙），或者也可以直接用你喜欢的编辑器编辑配置文件。
防火墙配置文件按小节把键-值策略对组织起来。以#字符开头的行和空行被当作注释处理。每个小节开头第一行格式都是“[小节名]”。

12.2.1 集群级别的防火墙配置

作用域为整个集群的防火墙配置保存在
/etc/pve/firewall/cluster.fw
该配置文件由以下小节构成：

[OPTIONS]

该小节用于设置整个集群的防火墙配置项。

ebtables: <boolean> (default = 1)

集群范围内启用 ebtables。

enable: <integer> (0 -N)

启用/禁用集群范围的防火墙。

log_ratelimit: [enable=]<1|0> [,burst=<integer>] [,rate=<rate>]

设置日志记录速度阀值。

burst=<integer> (0 - N) (default = 5)

将被记录的初始突发包。

enable=<boolean> (default = 1)

启用或禁用阀值

rate=<rate> (default = 1/second)

突发缓冲区重新填充频度。

policy_in: <ACCEPT | DROP | REJECT>

流入方向的防火墙策略。

policy_out: <ACCEPT | DROP | REJECT>

流出方向的防火墙策略。

[RULES]

该小节用于设置所有节点公共的防火墙策略。

[IPSET <name>]

整个集群范围内有效的 IP 集合定义。

[GROUP <name>]

整个集群范围内有效的组定义。

[ALIASES]

整个集群范围内有效的别名定义。

启用防火墙

防火墙默认是被完全禁用的。你可以按如下方式设置启用参数项：

```
[OPTIONS]
# enable firewall (cluster wide setting, default is disabled)
enable: 1
```

☒ 重要

启用防火墙后，默认所有主机的通信都将被阻断。唯一例外是集群网络内的 WebGUI（端口 8006）和 ssh（端口 22）访问可以继续使用。

如果你希望远程管理 Proxmox VE 服务器，你需要首先配置防火墙策略，允许远程 IP 访问 WebGUI（端口 8006）。根据需要，你还可以开通 ssh（端口 22）或 SPICE（端口 3128）的访问权限。

➢ 注意

请在启用防火墙前先打开到 Proxmox VE 服务器的一个 SSH 连接，这样即使策略配置有误，也还可以通过该连接访问服务器。

为简化配置，你可以创建一个名为“管理地址”的 IPSet，并把所有的远程管理终端 IP 地址添加进去。这样就可以创建策略允许所有的远程地址访问 WebGUI。

12.2.2 主机级别的防火墙配置

主机级别的防火墙配置保存在
/etc/pve/nodes/<nodename>/host.fw
该文件中的配置可以覆盖 cluster.fw 中的配置。你可以提升报警日志级别，设置 netfilter 相关参数。该配置文件由以下小节构成：

[OPTIONS]

该小节用于设置当前主机的防火墙配置项。

enable: <boolean>

启用/禁用主机防火墙策略。

log_level_in: <alert | crit | debug | emerg | err | info | nolog | notice | warning>

流入方向的防火墙日志级别。

log_level_out: <alert | crit | debug | emerg | err | info | nolog | notice | warning>

流出方向的防火墙日志级别。

log_nf_conntrack: <boolean> (default = 0)

启用记录连接跟踪信息。

ndp: <boolean>

启用 NDP。

nf_conntrack_allow_invalid: <boolean> (default = 0)

在跟踪连接时允许记录不合法的包。

nf_conntrack_max: <integer> (32768 -N)

最大的跟踪连接数量。

nf_conntrack_tcp_timeout_established: <integer> (7875 -N)

反向连接建立超时时间。

nosmurfs: <boolean>

启用 SMURFS 过滤器。

smurf_log_level: <alert | crit | debug | emerg | err | info | nolog | notice | warning>

SMURFS 过滤器日志级别。

tcp_flags_log_level: <alert | crit | debug | emerg | err | info | nolog | notice | warning>

非法 TCP 标志过滤器日志级别。

tcpflags: <boolean>

启用非法 TCP 标志组合过滤器。

[RULES]

该小节用于设置当前主机的防火墙策略。

12.2.3 虚拟机和容器级别的防火墙配置

虚拟机和容器级别的防火墙配置保存在
/etc/pve/firewall/<VMID>.fw
其内容由以下数据构成：

[OPTIONS]

该小节用于设置当前虚拟机或容器的防火墙配置项。

dhcp: <boolean>

启用 DHCP。

enable: <boolean>

启用/禁用防火墙策略。

ipfilter: <boolean>

启用默认 IP 地址过滤器。相当于为每个网卡接口增加一个空白的 ipfilter-net<id>地址集合。该 IP 地址集合隐式包含了一些默认控制，例如限制 IPv6 链路本地地址为网卡 MAC 生成的地址。对于容器，配置的 IP 地址将被隐式添加进去。

log_level_in: <alert | crit | debug | emerg | err | info | nolog | notice | warning>

流入方向的防火墙日志级别。

log_level_out: <alert | crit | debug | emerg | err | info | nolog | notice | warning>

流出方向的防火墙日志级别。

macfilter: <boolean>

启用/禁用 MAC 地址过滤器。

ndp: <boolean>

启用 NDP。

policy_in: <ACCEPT | DROP | REJECT>

流入方向的防火墙策略。

policy_out: <ACCEPT | DROP | REJECT>

流出方向的防火墙策略。

radv: <boolean>

允许发出路由通知。

[RULES]

该小节用于设置当前虚拟机或容器的防火墙策略。

[IPSET <name>]

IP 集合定义。

[ALIASES]

IP 地址别名定义。

启用虚拟机或容器上的防火墙

每个虚拟网卡设备都有一个防火墙启用标识。你可以控制每个网卡的防火墙启用状态。在设置启用虚拟机防火墙后，你必须设置网卡上的防火墙启用标识才可以真正启用防火墙。

12.3 防火墙策略

防火墙策略定义了网络通信方向（IN 或 OUT）和处理动作（ACCEPT，DENY，REJECT）。你也可以定义一个宏来预定义的策略和配置项，还可以在策略前插入字符“|”来禁用策略。

防火墙策略语法

[RULES]
DIRECTION ACTION [OPTIONS]
|DIRECTION ACTION [OPTIONS] # disabled rule
DIRECTION MACRO(ACTION) [OPTIONS] # use predefined macro

如下参数可用于完善策略匹配规则。

--dest <string>

设置数据包目的地址。可以设置为一个 IP 地址，一个 IP 集合（IP 集合名称）或 IP 别名。也可以设置为一个 IP 地址范围如 20.34.101.207-201.3.9.99，或一组 IP 地址和网络地址列表（使用逗号分隔开）。注意不要在列表中同时混合配置 IPv4 地址和 IPv6 地址。

--dport <string>

设置 TCP/UDP 目的端口。可像/etc/services 一样设置为服务名称或端口号（0-65535），也可按照“\d+:\d+”格式设置为端口范围，如 80:85，也可以设置为由逗号分隔开的端口和端口范围列表。

--iface <string>

设置网卡名称。可以设置为网络配置中的虚拟机和容器网卡名称（net\d+）。主机级别的防火墙策略可使用任意字符串。

--log <alert | crit | debug | emerg | err | info | nolog | notice | warning>

防火墙策略的日志级别。

--proto <string>

设置 IP 协议。你可以设置为协议名称（tcp/udp）或/etc/protocols 中定义的协议编号。

--source <string>

设置数据包源地址。可以设置为一个 IP 地址，一个 IP 集合（IP 集合名称）或 IP 别名。也可以设置为一个 IP 地址范围如 20.34.101.207-201.3.9.99，或一组 IP 地址和网络地址列表（使用逗号分隔开）。注意不要在列表中同时混合配置 IPv4 地址和 IPv6 地址。

--sport <string>

设置 TCP/UDP 目的端口。可像/etc/services 一样设置为服务名称或端口号（0-65535），也可按照“\d+:\d+”格式设置为端口范围，如 80:85，也可以设置为由逗号分隔开的端口和端口范围列表。

以下是一些防火墙策略示例

[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

12.4 安全组

安全组是一个防火墙策略的集合。安全组属于集群级别的防火墙对象，可用于所有的虚拟机防火墙策略。例如，你可以定义一个名为“webserver”的安全组，以开放 http 和 https 服务端口。
/etc/pve/firewall/cluster.fw
[group webserver]
IN ACCEPT -p tcp -dport 80
IN ACCEPT -p tcp -dport 443

之后，就可以将该安全组添加到虚拟机防火墙策略中
/etc/pve/firewall/<VMID>.fw
[RULES]
GROUP webserver

12.5 IP 地址别名

IP 地址别名能够让你为 IP 地址定义一个名称。之后可以通过名称来引用 IP 地址：

- 在 IP 集合内部
- 在防火墙的 source 和 dest 属性中

12.5.1 标准 IP 地址别名 local_network

该别名是系统自动定义的。可以使用如下命令查看分配的地址别名：
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/

防火墙将利用该别名自动生成策略，开放 Proxmox VE 集群对网络的访问权限（corosync，API，SSH）。
用户可以修改 cluster.fw 中定义的别名。如果你在公共网络上有一台独立的 Proxmox VE 主机，最好明确指定本地 IP 地址的别名
/etc/pve/firewall/cluster.fw
[ALIASES]
local_network 1.2.3.4 # use the single ip address

12.6 IP 地址集合

IP 地址集合可用来定义一组网络和主机。你可以在防火墙策略的 source 和 dest 属性中用"+名称"的格式引用 IP 地址集合。
如下策略将允许来自名为 management 的 IP 地址集合的 HTTP 访问
IN HTTP(ACCEPT) -source +management

12.6.1 标准 IP 地址集合 management

标准IP地址集合management仅限主机级别防火墙使用(不支持在虚拟机级别防火墙使用)。系统对该 IP 地址集合开放日常管理所需的网络访问权限（PVE GUI，VNC，SPICE，SSH）。本地集群网络地址将被自动添加到该 IP 地址集合（别名 cluster_network），以便于集群内的主机相互通讯（multicast，ssh 等）。
/etc/pve/firewall/cluster.fw
[IPSET management]
192.168.2.10
192.168.2.10/24

12.6.2 标准 IP 地址集合 blacklist

标准 IP 地址集合 blacklist 中的地址对任何主机或虚拟机发起的访问请求都将被丢弃。
/etc/pve/firewall/cluster.fw
[IPSET blacklist]
77.240.159.182
213.87.123.0/24

12.6.3 标准 IP 地址集合 ipfilter-net*

该类过滤器专门为虚拟机的虚拟网卡定义，主要用于防止 IP 地址欺骗。为虚拟网卡定义该 IP 地址集合后，从网卡发出的任何与 ipfilter 集合中 IP 地址不符的数据包都将被丢弃。

对于配置指定 IP 地址的容器，如果定义了该 IP 地址集合（或通过在虚拟机防火墙 options 选项卡勾选通用 IP Filter 激活），容器 IP 地址会被自动加入该 IP 地址集合。
/etc/pve/firewall/<VMID>.fw
[IPSET ipfilter-net0] # only allow specified IPs on net0
192.168.2.10

12.7 服务及管理命令

防火墙在每个节点都运行了两个服务进程：

- pvefw-logger: NFLOG 服务进程（替换 ulogd）。
- pve-firewall: 更新 iptables 策略。

还提供了一个管理命令 pve-firewall，可用于启停防火墙服务：
pve-firewall start
pve-firewall stop
或查看防火墙服务状态：
pve-firewall status
如上命令将读取并编译所有的防火墙策略，如果发现配置错误，将会自动发出告警。
如果你需要查看生成的 iptables 策略，可以运行如下命令：
iptables-save

12.8 默认防火墙策略

防火墙默认会对以下网络通信开启控制策略：

12.8.1 进/出数据中心的丢弃/拒绝策略

即使防火墙的出入控制策略被设为 DROP 或 REJECT, 集群内 Proxmox VE 主机仍将允许以下网络访问。

- 通过 loopback 端口的流量。
- 已建立的网络连接。
- 基于 IGMP 协议的通信流量。
- 开放管理终端到 8006 端口的 TCP 访问权限，以便访问 Web 管理控制台。
- 开放管理终端到 5900-5999 端口的 TCP 访问权限，以便使用 VNC 终端。

- 开放管理终端到 3128 端口的 TCP 访问权限，以便使用 SPICE 代理。
- 开放管理终端到 22 端口的 TCP 访问权限，以便 ssh 访问。
- 为 corosync 集群服务开放 5404 和 5405 端口的 UDP 访问权限。
- 为集群服务开放多播访问权限。
- 类型 3（目的不可达）、4（拥堵控制）、11（超时）的 ICMP 流量。

以下网络包将被丢弃，并且即使开启日志也不会被日志记录。

- 处于非法连接状态的 TCP 流量。
- 和 corosync 无关的广播、多播和任意目的数据包，即目的端口不是 5404 和 5405 的数据包。
- 目的端口是 43 的 TCP 数据包。
- 目的端口是 135 和 445 的 UDP 数据包。
- 目的端口是 137-139 的 UDP 数据包。
- 源端口是 137 且目的端口是 1024-65535 的 UDP 数据包。
- 目的端口是 1900 的 UDP 数据包。
- 目的端口是 135，139，445 的 TCP 数据包。
- 源端口是 53 的 UDP 数据包。

其余网络通信将被丢弃或拒绝，并被日志记录。具体处理结果由 Firewall→Options 中的选项决定，具体包括 NDP，SMURFS 和 TCP 标志位过滤等。
可以查看以下命令的输出
iptables-save
了解防火墙策略的活动情况。该命令的输出也会被合并到系统报告，并在 Web GUI 的节点描述选项卡展示，或通过 pvereport 命令查看。

12.8.2 进/出客户机的丢弃/拒绝策略

默认情况下，除 DHCP、NDP、路由告知、明确通过 MAC 和 IP 地址过滤策略排除的数据包以外，有关客户机的网络数据包都会被丢弃或拒绝。数据中心的丢弃和拒绝策略会被虚拟机自动继承，但和主机有关的例外通过策略则不会被集成。
可以使用 12.8.1 节中的 iptables-save 命令查看所有访问控制策略。

12.9 防火墙日志记录

默认情况下，所有的防火墙策略都不产生日志记录。如要启用日志记录，需要在 Firewall→Options 中设置出/入网络数据包的日志级别 loglevel。主机、客户机的日志级别可以分别设置。设置后，Proxmox VE 的防火墙日志将被启用，并可以在 Firewall→Log 中查看。需要注意，只有被标准策略丢弃或拒绝的数据包会产生日志记录（详见 12.8 节默认防火墙策略）。防火墙日志级别 loglevel 并不影响产生日志数量。只用于改变日志记录的 LOGID 前缀，以便后续处理。
具体的 loglevel 取值如下表：

loglevel	LOGID
nolog	—
emerg	0
alert	1
crit	2
err	3
warning	4
notice	5
info	6
debug	7

典型的防火墙日志记录如下：
VMID LOGID CHAIN TIMESTAMP POLICY: PACKET_DETAILS
如为主机防火墙日志记录，VMID 会被设置为 0.

12.9.1 用户自定义防火墙策略日志记录

如需让用户自定义防火墙策略生成日志记录，可以为每条策略分别设置日志级别。通过 Firewall→Options 可以为每条策略设置非常精细的日志级别。
当然可以通过 WebUI 在创建或修改每条策略时设置或改变 loglevel，也可以通过 pvesh 调用相应 API 进行设置。
此外，还可以通过修改防火墙日志文件调整日志级别，只要在相应策略后添加-log <loglevel>即可。
示例如下，以下两条命令效果是一样的，都不产生日志：
IN REJECT -p icmp -log nolog
IN REJECT -p icmp
但以下策略将产生 debug 级别日志。
IN REJECT -p icmp -log debug

12.10 提示和窍门

12.10.1 如何开放 FTP

FTP 是一个古老的协议，使用固定端口 21 和其他一些动态端口。所以，你需要配置一条开放端口 21 的策略，并加载 ip_contrack_ftp 内核模块。加载命令如下：
modprobe ip_conntrack_ftp
进一步还需要在/etc/modules 中添加 ip_contrack_ftp（以便系统重启后自动加载）。

12.10.2 集成 Suricata IPS

你也可以集成使用 Suricata IPS（入侵防御系统）。
只有通过防火墙策略校验的数据包才会发送给 IPS。
被防火墙拒绝/丢弃的数据包不会发送给 IPS。
首先需要在 Proxmox VE 主机安装 suricata：
apt-get install suricata
modprobe nfnetlink_queue
不要忘记在/etc/modules 中添加 nfnetlink_queue，以便系统下次重启后自动加载。
然后可以在指定虚拟机的防火墙上激活 IPS：
/etc/pve/firewall/<VMID>.fw
[OPTIONS]
ips: 1
ips_queues: 0
ips_queues 配置项将为虚拟机绑定一个指定的 cpu 队列。
可用队列定义在如下配置文件中
# /etc/default/suricata
NFQUEUE=0

12.11 IPv6 注意事项

防火墙中有一些专用于 IPv6 的配置项。首先，IPv6 不再使用 ARP 协议，取而代之的是 NDP（Neighbor Discovery Protocol），而 NDP 工作在 IP 层，需要配置 IP 地址后才可以使用。为此，系统用虚拟网卡 MAC 地址生成了一个 IPv6 链路本地地址。在主机级别防火墙和虚拟机级别的防火墙上，NDP 配置项默认都是启用的，以便邻居发现（NDP）数据包的收发。除了邻居发现以外，NDP 也被用于完成其他任务，比如自动配置和路由通知。
虚拟机默认可以发送路由查询消息（以获取路由）和接收路由通知数据包。这允许虚拟机使用无状态的自动配置。但是，虚拟机默认不能向外发送宣称自己是路由器的路由通知数据包，除非设置“允许路由通知”（radv:1）配置项。

为便于 NDP 使用链路本地地址通信，防火墙提供了一个“IP 过滤器”（ipfilter:1）配置项。启用该配置的效果类似于在虚拟机网卡上启用 IP 地址集合 ipfilter-net*，然后把链路本地地址添加进去一样（详情可查看标准 IP 地址集合 ipfilter-net*一节）。

12.12 Proxmox VE 端口列表

- Web 界面：8006
- VNC 控制台：5900-5999
- SPICE proxy：3128
- sshd（用于集群管理）：22
- rpcbind：111
- corosync 多播（集群通信使用）：5404, 5405 UDP

第 13 章 用户管理

Proxmox VE 支持多种身份认证方式，例如 Linux PAM，内部 Proxmox VE 认证服务，微软活动目录。
基于角色的用户和权限管理覆盖了所有对象（虚拟机，存储，节点等），能够实现细粒度的访问控制。

13.1 用户

Proxmox VE 的用户信息保存在/etc/pve/user.cfg 中。但该文件中不保存口令信息，用户通过本章后续介绍的认证域进行认证。因此，Proxmox VE 需要联合用户名和认证域信息<userid>@<realm>才可以定义完整的用户身份。
配置文件中每条用户记录同时包含了以下信息

- 名
- 姓
- 电子邮件
- 组
- 可选的过期时间
- 用户信息注释
- 用户启用/禁用标志
- 双因子认证密钥

13.1.1 系统管理员

系统 root 用户可以通过 Linux PAM 域登录系统，并拥有最高管理权限。该用户不能被删除，但其属性可以被修改，系统邮件会发送到为该用户分配的电子邮件地址。

13.1.2 组

用户可以同时加入多个组。组可以有效简化访问权限控制工作。以组为单位赋予访问权限比

直接向单个用户赋权要方便的多，最终得到的访问控制列表也要短的多，便于处理。

13.2 认证域

Proxmox VE 用户实际上其他外部认证域用户的一个副本。认证域信息都保存在/etc/pve/domains.cfg。以下是可用的认证域：

Linux PAM 标准认证

该认证域要求在集群的所有节点上创建相同的 Linux 系统用户（例如，使用 adduser 命令创建用户），并使用 Linux 口令认证用户身份。

useradd heinz
passwd heinz
groupadd watchman
usermod -a -G watchman heinz

Proxmox VE 认证服务器

用户口令保存在 Unix 风格的口令文件（/etc/pve/priv/shadow.cfg）中。口令使用 SHA-256 哈希算法加密。该方式是小规模（或中等规模）环境下最便于使用的认证方式。用户在 Proxmox VE 内部即可完成身份认证，无须任何外部支持，所有用户身份都由 Proxmox VE 管理，并可在 WebGUI 界面直接修改口令。

LDAP

也可以使用 LDAP 服务器（例如 openldap）进行用户身份认证。LDAP 支持部署备用服务器，并允许通过加密的 SSL 连接传递认证信息。

LDAP 将在基本域名（base_dn）下搜索用户属性名（user_attr）指定的用户名。

例如，如果一个用户的 ldif 身份信息记录如下：

user1 of People at ldap-test.com
dn: uid=user1,ou=People,dc=ldap-test,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: user1
cn: Test User 1
sn: Testers
description: This is the first test user.

则基本域名为“ou=People,dc=ldap-test,dc=com”，用户属性为“uid”。

如果 Proxmox VE 在请求验证用户身份前需要先认证（bind）ldap 服务器的真实性，可以通过/etc/pve/domains.cfg 中的 bind_dn 属性配置认证域名称。认证口令则保存在/etc/pve/priv/ldap/<realmname>.pw（例如/etc/pve/priv/ldap/my-ldap.pw），其中仅有一行明文口令信息。

微软活动目录

该方式需要指定认证域和认证服务器。类似 ldap，可以配置备用服务器，可选端口和 SSL 加密。

13.3 双因子认证

有两种方式可以实现双因子认证：
首先是可以通过认证域方式实现，也就是 TOTP 或 YubiKey OTP 实现。使用这种方式，新建用户时需要将其持有的 Key 信息添加到系统，不然就没办法登录。使用 TOTP 的用户，如果被允许先登录，可以在登录后修改 TOTP 信息。
此外，如果认证域未强制要求提供双因子认证，用户也可以通过 TOTP 选择自行启用双因子认证。如果服务器配置了 AppID，且未强制开启其他双因子认证方式，用户也可以选择使用 U2F 认证。

13.3.1 强制双因子认证域

只需在添加或修改认证域时从 TFA 下拉框中选择可用的双因子认证方法即可。当一种认证域启用双因子认证后，只有能提供双因子认证信息的用户才可以登录。
目前有两种可用的双因子认证方法：

时间令牌（TOTP）

该方式采用标准 HMAC-SHA1 算法，用当前时间和用户密钥计算得到的哈希值认证用户身份。而时间步进长度和口令长度都是可以配置的。
一个用户可以设置多个密钥（用空格分隔开），可用 Base32（RFC3548）或 16 进制形式记录。
Proxmox VE 提供了密钥生成工具（oathkeygen），可以产生 Base32 格式的随机密钥，与多种 OTP 工具直接配合使用，例如 oathtool 命令行工具，Google 认证器或 Android 应用 FreeOTP。

YubiKey 令牌

使用 YubiKey 进行双因子认证，必须预先配置 Yubico API ID，API 密钥和可用的服务器 URL，同时用户必须拥有 YubiKey 硬件。如果要从 YubiKey 提取 Key ID，需要将 YubiKey 插入计算机 USB 接口并激活，然后将输入口令的前 12 个字符拷贝到用户的 Key ID 字段。
关于 YubiCloud 使用方法和如何建立自己的认证服务器，可查看 YubiKey OTP 文档。

13.3.2 用户自定义 TOTP 认证

在未强制使用 YubiKey OTP 的情况下，用户可以在登录时在用户列表通过 TFA 按钮选择使用 TOTP 双因子认证。

打开 TFA 窗口后，可以得到 TOTP 认证配置对话框。其中 Secret 栏是 key 信息，可以通过 Randomize 按钮自动生成新的 key。Issue Name 为可选项，用于设置 key 所属的 TOTP app 信息。大部分 TOTP 应用会显示 Issue name 和 OTP 值。用户名也将包含在 TOTP app 的二维码中。

生成 key 后，会自动产生一个二维码，可用于大部分 OTP app，如 FreeOTP。用户登录时需要提供用户口令（root 用户除外），并在 Verification Code 栏输入当前 OTP 值，最后点击 Apply 按钮。

13.3.3 服务器端 U2F 配置

如需使用 U2F 认证，服务器需要配置拥有合法 https 证书的域。还需要配置初始的 AppID。

➢ 注意

修改 AppID 会导致已有 U2F 注册失效。

具体可以通过/etc/pve/datacenter.cfg 配置，示例如下：

u2f: appid=https://mypve.example.com:8006

对单一节点，AppID 可以是浏览器中的 Web UI 地址，包括 https://头以及端口信息。某些浏览器匹配 AppID 的规则会比其他浏览器更严格。

对多节点集群，最好使用独立的 https 服务器提供 appid.json 文件，这种方式能兼容更多浏览器。如果所有节点都使用同一顶级域下的子域，可以使用 TLD 作为 AppID，但要注意只有部分浏览器兼容这种做法。

➢ 注意

损坏的 AppID 通常会导致错误，但也会遇到不发生错误的情形，特别在使用 Chromium 和顶级域 AppID 访问节点时。有鉴于此，建议对多种浏览器测试有关配置，特别是修改 AppID 可能导致已有 U2F 注册失效的情况。

13.3.4 激活用户 U2F 认证

如需使用 U2F 认证，首先要打开 TFA 窗口的 U2F 选项卡，输入当前口令（root 用户除外），点击注册按钮。如果服务器配置正确，浏览器也接受服务器提供的 AppID，系统会弹出消息，提示用户点击 U2F 设备按钮（如使用的是 YubiKey，设备按钮等将会以每秒两次的频率闪烁）。Firefox 用户可能需要使用 U2F 令牌前通过 about:config 启用 security.webauth.u2f。

13.4 权限管理

用户进行任何操作前（例如查看、修改、删除虚拟机配置），都必须被赋予合适的权限。Proxmox VE 采用了基于角色和对象路径的权限管理系统。权限管理表中的一个条目记录了用户或组在访问某个对象或路径时所拥有的角色。也就是说，每条访问策略都可以用（路径，用户，角色）或（路径，组，角色）三元组来表示，其中角色包含了允许进行的操作，路径标明了操作的对象。

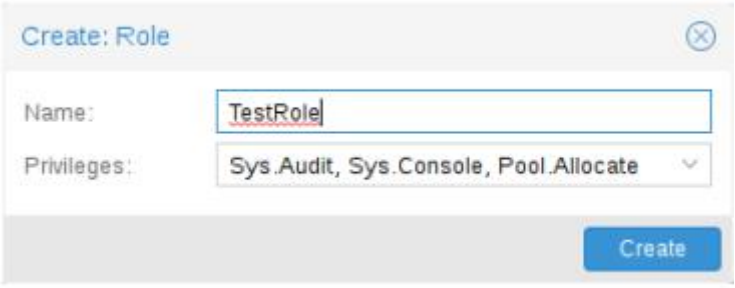
13.4.1 角色

角色实际上是一个权限列表。Proxmox VE 预定义了多个角色，能够满足大部分的管理需要。

- Administrator：拥有所有权限
- NoAccess：没有任何权限（用于禁止访问）
- PVEAdmin：有权进行大部分操作，但无权修改系统设置（Sys.PowerMgmt，Sys.Modify，Realm.Allocate）。
- PVEAuditor：只有只读权限。
- PVEDatastoreAdmin：创建和分配备份空间和模板。
- PVEDatastoreUser：分配备份空间，查看存储服务。
- PVEPoolAdmin：分配资源池。
- PVESysAdmin：分配用户访问权限，审计，访问系统控制台和系统日志。

- PVETemplateUser：查看和克隆模板。
- PVEUserAdmin：用户管理。
- PVEVMAdmin：管理虚拟机。
- PVEVMUser：查看，备份，配置 CDROM，访问虚拟机控制台，虚拟机电源管理。

在 WebGUI 可以查看系统预定义的所有角色。
目前，新增角色可以通过 GUI 或命令行进行。



使用 GUI 方式时，依次打开数据中心的 Permissions→User 选项卡，然后点击创建按钮，之后可以设置角色名并在权限下拉框中选择所需权限。
使用命令行方式添加角色时，可以使用 pveum 命令行工具，如下：

```
pveum roleadd PVE_Power-only -privs "VM.PowerMgmt VM.Console"
pveum roleadd Sys_Power-only -privs "Sys.PowerMgmt Sys.Console"
```

13.4.2 权限

权限是指进行某种操作的权力。为简化管理，一组权限可以被编组构成一个角色，角色可以用于制定权限管理表的条目。注意，权限不能被直接赋予用户和对象路径，而必须借助角色才可以。
目前有如下权限：
节点/系统相关的权限

- Permissions.Modify：修改访问权限
- Sys.PowerMgmt：管理节点电源（启动，停止，重启，关机）
- Sys.Console：访问节点控制台
- Sys.Syslog：查看 syslog
- Sys.Audit：查看节点状态/配置
- Sys.Modify：创建/删除/修改节点网络配置参数
- Group.Allocate：创建/删除/修改组

- Pool.Allocate：创建/删除/修改资源池
- Realm.Allocate：创建/删除/修改认证域
- Realm.AllocateUser：将用户分配到认证域
- User.Modify：创建/删除/修改用户访问权限和详细信息

虚拟机相关的权限

- VM.Allocate：创建/删除虚拟机
- VM.Migrate：迁移虚拟机到其他节点
- VM.PowerMgmt：电源管理（启动，停止，重启，关机）
- VM.Console：访问虚拟机控制台
- VM.Monitor：访问虚拟机监视器（kvm）
- VM.Backup：备份/恢复虚拟机
- VM.Audit：查看虚拟机配置
- VM.Clone：克隆/复制虚拟机
- VM.Config.Disk：添加/修改/删除虚拟硬盘
- VM.Config.CDROM：弹出/更换 CDROM
- VM.Config.CPU：修改 CPU 配置
- VM.Config.Memory：修改内存配置
- VM.Config.Network：添加/修改/删除虚拟网卡
- VM.Config.HWType：修改模拟硬件类型
- VM.Config.Options：修改虚拟机的其他配置
- VM.Snapshot：创建/删除虚拟机快照

存储相关的权限

- Datastore.Allocate：创建/删除/修改存储服务，删除存储卷
- Datastore.AllocateSpace：在存储服务上分配空间
- Datastore.AllocateTemplate：分配/上传模板和 iso 镜像

- Datastore.Audit：查看/浏览存储服务

13.4.3 对象和路径

访问权限是针对对象而分配的，例如虚拟机，存储服务或资源池。我们采用了类似文件系统路径的方式来标识这些对象。所有的路径构成树状结构，用户可以选择将高层对象获得的权限（短路径）扩展到下层的对象。
路径是可模板化的。当 API 调用申请访问一个模板化路径时，路径中可以包含 API 调用的参数。其中 API 参数需要用花括号括起来。某些参数会被隐式地从 API 调用的 URI 中获取。例如在调用/nodes/mynode/status 时，路径/nodes/{node}实际上申请了/nodes/mynode 的访问权限。而在对路径/access/acl 的 PUT 请求中，{path}实际上引用了该方法的 path 参数。
一些示例如下：

- /nodes/{node}：访问 Proxmox VE 服务器
- /vms：所有的虚拟机
- /vms/{vmid}：访问指定虚拟机
- /storage/{storeid}：访问指定存储服务
- /pool/{poolname}：访问指定存储池中虚拟机
- /access/groups：组管理操作
- /access/realms/{realmid}：管理指定认证域

权限继承

如前所述，对象路径全体构成了类似文件系统的树状结构，而权限能够自上而下地继承下去（继承标识默认是启用的）。我们采用了以下的继承策略：

- 针对单一用户的权限将覆盖针对组的权限。
- 针对组赋权后，组内所有用户自动获得赋限。
- 明确的赋权会覆盖从高层继承来的赋权。

13.4.4 资源池

资源池主要用来将虚拟机和存储服务组织起来，并形成一个组。当对资源池赋予访问权限后（/pool/{poolid}），其中所有成员都会继承该权限，从而大大简化访问控制配置工作。

13.4.5 我究竟需要哪些权限？

在 http://pve.proxmox.com/pve-docs/api-viewer/记录了每一个方法所需的 API 调用权限。所需的权限以列表形式表示，可以看作一个由访问权限检查函数构成的逻辑树。

["and", <subtests>...] and ["or", <subtests>...]

当前列表中的所有（and）或任意一个（or）权限需要被满足。

["perm", <path>, [<privileges>...], <options>...]

该路径是一个模板参数（查看对象和路径一节）。访问目标路径时，列表中所有的（或任意一个，如果使用了 any 选项）权限需要被满足。如果指定了 require-param 选项，则需要满足指定的参数权限，除非 API 调用时标明该参数为可选的。

["userid-group", [<privileges>...], <options>...]

调用方需要拥有/access/groups 所列出的任意权限。此外，根据是否设置 groups_param 参数，还需要额外进行两个权限检查：

- 设置了 groups_param：API 调用使用了不可选的组参数，调用方必须对参数引用的所有组拥有该组所列出的任意权限。
- 未设置 groups_param：通过 userid 参数传递的用户必须存在，并且是组的成员，而调用方拥有所列出的任意权限（通过/access/groups/<group>路径）。

["userid-param", "self"]

向 API 传递的 userid 参数值必须和申请进行操作的用户一致。(通常和 or 联合使用，以允许用户在没有权限的情况下在自身执行操作。)

["userid-param", "Realm.AllocateUser"]

用户需要对/access/realm/<realm>,拥有 Realm.AllocateUser 访问权。其中<realm>是用户通过 userid 参数传递的认证域。注意，用户不一定需要真的存在，因为用户 ID 是以<username>@<realm>形式传递的。

["perm-modify", <path>]

其中 path 是一个模板化参数（参见对象和路径一节）。用户需要拥有 Permissions.Modify 权限，或根据以下不同路径拥有相应权限：

- /storage/...：需要额外拥有权限’ Datastore.Allocate`
- /vms/...：需要额外拥有权限’ VM.Allocate`
- /pool/...：需要额外拥有权限’ Pool.Allocate`

如果路径为空，需要对/access 拥有 Permission.Modify 权限。

13.5 命令行工具

大部分用户使用 WebGUI 就能够完成用户管理任务了。但 Proxmox VE 还提供了一个全功能

的命令行工具 pveum（“Proxmox VE User Manager”的缩写）。由于 Proxmox VE 的命令行工具都通过封装 API 实现的，因此你也可以通过调用 REST API 来使用这些功能。
如下是一些使用示例。如需要显示帮助信息，可运行：
pveum
或（针对特定命令显示更详细的信息）
pveum help useradd
创建新用户：
pveum useradd testuser@pve -comment "Just a test"
设置或修改口令（不是所有认证域都支持该命令）：
pveum passwd testuser@pve
禁用用户：
pveum usermod testuser@pve -enable 0
创建新用户组：
pveum groupadd testgroup
创建新角色：
pveum roleadd PVE_Power-only -privs "VM.PowerMgmt VM.Console"

13.6 实际应用示例

13.6.1 管理员组

一个很实用的特性是创建一组具有全部管理权限的管理员用户（不使用 root 用户）。
定义管理员组：
pveum groupadd admin -comment "System Administrators"
赋予权限：
pveum aclmod / -group admin -role Administrator
向管理员组添加管理员用户：
pveum usermod testuser@pve -group admin

13.6.2 审计员

赋予用户或用户组 PVEAuditor 角色就可以赋予相应用户对系统的只读权限。
例 1：允许用户 joe@pve 查看系统所有对象
pveum aclmod / -user joe@pve -role PVEAuditor
例 2：允许用户 joe@pve 查看所有虚拟机
pveum aclmod /vms -user joe@pve -role PVEAuditor

13.6.3 分配用户管理权限

如果需要将用户管理权限赋予 joe@pve，可以运行如下命令：
pveum aclmod /access -user joe@pve -role PVEUserAdmin

之后，joe@pve 用户就可以添加和删除用户，修改其他用户的口令和属性。这是一个权限非常大的角色。你应该将该权限限制在指定的认证域和用户组。以下是限制 joe@pve 仅能修改 pve 认证域中 customers 用户组用户的示例：

pveum aclmod /access/realm/pve -user joe@pve -role PVEUserAdmin

pveum aclmod /access/groups/customers -user joe@pve -role PVEUserAdmin

- **注意**

执行以上命令后，joe@pve 用户能够添加用户，但添加的用户只能属于 pve 认证域中的 customers 用户组。

13.6.4 资源池

一个企业往往设立有多个部门，将资源和管理权限分配给各个部门是很常见的做法。资源池是一组虚拟机和存储服务的集合，你可以在 WebGUI 创建资源池，然后向资源池添加资源（虚拟机，存储服务）。

你可以向资源池赋予访问权限，这些权限会被其成员自动继承获取。

假定你有一个软件开发部，首先创建用户组

pveum groupadd developers -comment "Our software developers"

然后为该组创建一个新用户

pveum useradd developer1@pve -group developers –password

- **注意**

参数-password 将会提示你设立用户口令。

假定你已经通过 WebGUI 创建资源池“dev-pool”，现在我们可以向该资源池赋予访问权限：

pveum aclmod /pool/dev-pool/ -group developers -role PVEAdmin

现在我们的软件开发部门就可以管理该资源池中的资源了。

第 14 章 高可用性

现代社会严重依赖于计算机网络提供的信息，移动终端的普及进一步加重了这种依赖，人们无时无地都需要能够访问网络。如果你在提供这类服务，那么确保服务始终在线就变得非常重要。

我们可以通过计算服务在线时间（A）和总时间段（B）的比值来定义服务可用性。通常都表示为在一年内的在线时间比率。

表 14.1：可用性，一年内的当机时间

可用性 %	**一年内的停机时间**
99	3.65 **天**
99.9	8.76 **小时**
99.99	52.56 **分钟**
99.999	5.26 **分钟**
99.9999	31.5 **秒**
99.99999	3.15 **秒**

提高可用性的方法有很多。最具逼格的是重写软件，以便软件能够同时在多个主机并发运行。这要求软件本身具备错误检测和故障转移能力。对于只包含静态页面的 Web 网站来说，这种方法还能凑合用用。但更多情况下，这种方式都非常复杂，经常因为你无法修改软件而完全没有可行性。以下是一些不修改软件的提高可用性的办法：

- 使用可靠的服务器硬件

➢ 注意

由于质量的不同，相同功能的计算机硬件往往具有不同的可用性指标。大部分厂商将可靠性较高的硬件作为“服务器级”产品出售，当然其价格也更高。

- 消除单点故障（冗余硬件）
  - 使用不间断电源（UPS）
  - 为主板配备多路电源

- 使用 ECC 内存
- 使用多路网卡
- 使用 RAID 技术管理本地存储
- 使用分布式多副本存储技术保存虚拟机镜像

- 减少停机时间
  - 可快速访问的管理界面（24/7）
  - 可用的空闲节点（Proxmox VE 集群中的其他节点）
  - 自动化故障检测（ha-manager 提供）
  - 自动化故障转移（ha-manager 提供）

由于彻底消除了对硬件的依赖，Proxmox VE 这样的虚拟化技术能够轻松实现服务的高可用性。在配置了冗余存储和网络资源的情况下，遭遇个别服务器节点故障时，可以很容易在集群中其他服务器节点恢复服务运行。
Proxmox VE 进一步提供了 ha-manager 组件，能够自动完成包括故障检测和故障转移在内的一切高可用管理任务。
Proxmox VE 的 ha-manager 组件就像一个“全自动”的管理员。你只需将资源（虚拟机，容器等）配置交给它管理，ha-manager 就会连续监测服务运行状态，并在发生故障时将服务转移到其他节点运行。当然，ha-manager 也可以处理日常的管理操作请求，例如开机、停止、重新部署和迁移虚拟机。
但高可用性不是免费的午餐。实现高可用性需要投入更多资源，预备空闲节点等都会增加成本，因此你应该认真计算评估高可用性的收益和所需成本。

➢ 注意

将可用性从 99%提高到 99.9%还是比较容易的。但从 99.9999%提高到 99.99999%则难的多也贵的多。ha-manager 的故障检测和故障转移时间大概为 2 分钟，因此能实现的可用性最多不超过 99.999%。

14.1 部署条件

在开始部署 HA 之前，需要满足以下条件：

- 集群最少有 3 个节点（以得到稳定的 quorum）
- 为虚拟机和容器配置共享存储
- 硬件冗余（各个层面）

- 使用可靠的“服务器”硬件
- 硬件看门狗 - 如不具备 ,也可以退而求其次使用 Linux 内核的软件看门狗(softdog)
- 可选的硬件隔离设备

14.2 资源

我们将 ha-manager 管理的对象称为资源。一个资源（也称为“服务”）由一个唯一的服务 ID 标识（SID）。服务 ID 由资源类型和类型内的 ID 两部分组成，例如 vm:100，是指一个 vm 类型（虚拟机）的资源，而资源 ID 为 100。
目前主要有两类资源，虚拟机和容器。一个资源对应一个虚拟机或容器，资源的所有相关软件需要安装到这个虚拟机或容器中，而不是像 rgmanager 那样把多个资源捆绑成一个大资源。通常来说，HA 管理的资源不应再依赖其他资源。

14.3 管理任务

本节将简单介绍常见管理任务。首先是在资源上激活 HA，也就是把资源添加到 HA 的资源配置中，可以通过 WebGUI 进行，也可以使用命令行工具完成该操作，如下：
```
# ha-manager add vm:100
```
之后，HA 组件将启动该资源并全力确保它连续运行。当然，你也可以配置该资源的“指定”工作状态，例如可以要求 HA 组件停止该资源的运行：
```
# ha-manager set vm:100 --state stopped
```
然后再启动运行
```
# ha-manager set vm:100 --state started
```
你也可以使用常用的虚拟机和容器管理工具来改变资源运行状态，而常用工具会自动调用 HA 组件完成操作指令。因此
```
# qm start 100
```
将资源状态设置为 started。命令 qm stop 的原理类似，只是将资源状态设置为 stopped。

➢ 注意

HA 组件以异步方式工作，并需要和集群其他成员进行通讯，因此从发出指令到观察到操作完成需要一些时间。

可以用如下命令查看 HA 的资源配置情况：
```
# ha-manager config
vm:100
        state stopped
```
可以用如下命令查看 HA 管理器和资源状态：
```
# ha-manager status
quorum OK
```

master node1 (active, Wed Nov 23 11:07:23 2016)
lrm elsa (active, Wed Nov 23 11:07:19 2016)
service vm:100 (node1, started)
可以用如下命令将资源迁移到其他节点：
ha-manager migrate vm:100 node2
上面的命令采用在线迁移方式，虚拟机在迁移过程中将保持运行。在线迁移需要通过网络将虚拟机内存数据传输到目标节点，因此在某些情况下关闭虚拟机然后在目标节点重新启动可能更快，具体可使用 relocate 命令进行：
ha-manager relocate vm:100 node2
最后，可用如下命令将资源从 HA 的资源配置中删除：
ha-manager remove vm:100

➢ **注意**

该操作并不需要启停虚拟机。

所有的 HA 管理操作都可通过 WebGUI 进行，一般情况下无须使用命令行。

14.4 工作原理

本节将详细描述 HA 管理器的内部工作原理，包括所有服务进程及其协同工作过程。HA 在每个节点上都有两个服务进程：

pve-ha-lrm

该服务称为本地资源管理器（LRM），其主要任务是控制本地节点的资源运行状态，首先从当前管理器状态文件读取资源的指定工作状态，然后执行相应的操作命令。

pve-ha-crm

该服务称为集群资源管理器（CRM），其主要任务是负责集群节点之间的协同决策工作，具体包括向 LRM 发送命令，处理命令执行结果，在出现故障时将资源转移到其他节点运行，此外还负责故障节点隔离。

➢ **注意**

HA 服务利用了集群文件系统提供的锁机制。通过锁机制，确保了每次只有一个 LRM 被激活并处于工作状态。由于 LRM 只在获取锁之后才能执行 HA 任务，我们可以在获取锁之后将故障节点标记为隔离，然后可以在其他节点安全地恢复原来在故障节点运行的 HA 资源，而无须担心故障节点的干扰。整个过程都在拥有 HA 管理器主锁的 CRM 监督下进行。

14.4.1 资源状态

CRM 使用一个枚举变量来记录当前资源的状态。不仅 WebGUI 界面有显示当前资源状态，并且你可以运行 ha-manager 命令行工具获取该状态。
ha-manager status
quorum OK
master elsa (active, Mon Nov 21 07:23:29 2016)
lrm elsa (active, Mon Nov 21 07:23:22 2016)

service ct:100 (elsa, stopped)
service ct:102 (elsa, started)
service vm:501 (elsa, started)
以下是可能的状态

stopped

资源已停止（LRM 确认）。如果 LRM 检测到应处于停止状态的资源仍然在运行，它将再次停止该资源。

request_stop

资源应被停止。该状态下，CRM 将等待 LRM 确认资源已停止。

stopping

正在挂起的停止请求。表示 CRM 仍未接到该停止请求。

started

资源处于运行状态，并且 LRM 应该在发现资源未运行时立刻启动该资源。如果资源因故停止运行，LRM 会在检测到后立刻重启它（查看 14.8 节启动失败策略）。

starting

正在挂起的启动请求。表示 CRM 未得到 LRM 对该资源正在运行的确认。

fence

等待节点完成隔离（将节点从集群投票范围内隔离出去）。一旦完成隔离，资源将在其他节点恢复（查看 14.7 节隔离）。

freeze

表示禁止访问资源。该状态用于节点重启过程，或 LRM 重启过程（查看 14.10 节软件包升级）。

ignored

将虚拟机暂时脱离 HA 管理。可用于临时人工管控虚拟机，同时保留 HA 配置不变。

migrate

将资源迁移（在线）到其他节点。

error

因 LRM 错误，资源被禁用。该状态往往意味着需要手工干预（查看 14.9 节错误恢复）。

queued

表示资源刚被添加到 HA，而 CRM 尚未确认已看到该资源。

disabled

资源被停止运行，并被标记为 disabled。

14.4.2 本地资源管理器

本地资源管理器（pve-ha-lrm）以系统服务形式启动。启动后，该服务将等待集群进入多数票状态，以确保集群锁机制正常工作。

该服务有 3 种状态：

wait for agent lock

表示 LRM 在等待获取的独占锁。如果未配置任何 HA 资源，该状态就相当于空闲状态。

active

表示配置了 HA 资源，并且 LRM 获得了独占锁。

lost agent lock

表示 LRM 失去了独占锁，一般意味着有错误发生，并且节点失去了多数票。

LRM 进入 active 状态后，将读取配置文件/etc/pve/ha/manager_status，并根据它所管理的资源判断应该执行的管理命令。每条命令都由一个独立工作进程执行，因此可以并发执行多条命令，但默认最多同时并发执行 4 条命令。可以修改数据中心配置项 max_worker 来调整默认并发数。当命令执行完后，工作进程将被回收，执行结果也会被 CRM 记录保存。

➢ 注意

默认的并发数 4 不一定适用于所有环境。例如，同时执行 4 个在线迁移操作可能会导致对网络的竞争使用，特别在物理网络速度较慢或配置了大内存资源时。在任何情况下必须确保避免发生竞争的情况，必要时可以降低 max_worker 的值。相反，如果你的硬件配置极端牛逼，也可以考虑增加 max_worker 的值。

CRM 发出的每条命令都由一个 UID 标识，当工作进程完成命令执行后，执行结果将被写入 LRM 状态文件/etc/pve/nodes/<nodename>/lrm_status，而 CRM 可能会收集该结果并用它自己的状态机进一步处理该结果。

通常，CRM 和 LRM 对每一个资源的操作都是同步进行的。也就是说，CRM 发出一个唯一 UID 标识的命令，LRM 则执行一次该命令并将执行结果写回文件，而执行结果用同一个 UID 标识。这确保了 LRM 不会执行过期的命令。但 stop 命令和 error 命令是两个例外，这两个命令不依赖于处理结果，并总是在 stopped 或 error 状态执行。

➢ 注意

HA 组件会记录每个操作的日志。这有助于理解集群中发生的事以及发生的原因。这对于了解两个服务进程 LRM 和 CRM 干了什么尤为重要。你可以用命令 journalctl -u pve-ha-lrm 查看资源所在节点的本地资源管理器日志，并用同样命令查看当前主节点的 pve-ha-crm 服务日志。

14.4.3 集群资源管理器

集群资源管理器（pve-ha-crm）在每个节点启动后，将进入等待状态直到获取管理器锁。管理器锁每次只能由一个节点获取，而成功获取该锁的节点将被提升为 CRM 主节点。

该服务有 3 种状态：

wait for agent lock

表示 CRM 在等待获取的独占锁。如果未配置任何 HA 资源，该状态就相当于空闲状态。

active

表示配置了 HA 资源，并且 CRM 获得了独占锁。

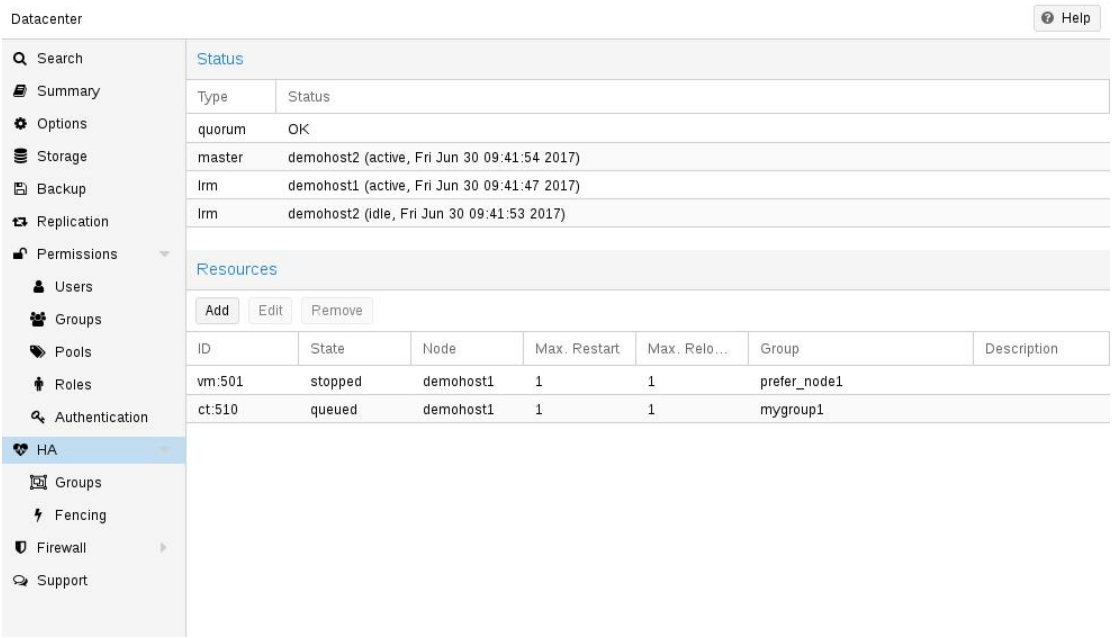
lost agent lock

表示 CRM 失去了独占锁，一般意味着有错误发生，并且节点失去了多数票。

CRM 的主要任务是管理那些纳入 HA 管理的资源，并尽力确保资源处于指定的状态。例如，对于一个指定状态为 started 的资源，一旦被发现未运行就会立刻被启动，如果资源意外崩

溃，也会被自动重启。而 CRM 将负责告诉 LRM 具体进行哪些操作。
一个节点失去集群多数票后，会进入 unknown 状态。此时，如果 CRM 能够安全释放故障节点的锁，相关资源将会被转移到其他节点重新启动。
当集群节点判定自己不再拥有集群多数票后，LRM 将等待新的多数票形成。只要没能形成多数票，节点就无法重置看门狗，最终看门狗超时后会触发节点重启，看门狗默认超时时间为 60 秒。

14.5 HA 仿真器



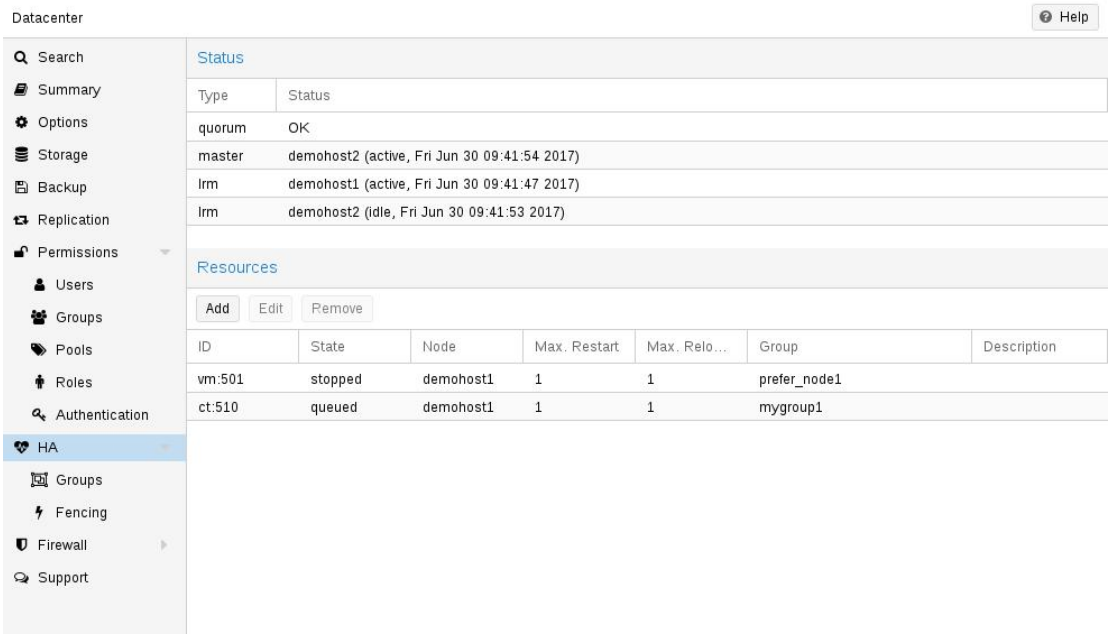
通过 HA 仿真器，用户可以了解并测试 Proxmox VE HA 解决方案的所有功能。
默认情况下，可以通过 HA 仿真器测试带有 6 个虚拟机的 3 节点集群环境。用户还可以根据需要增加删除虚拟机或容器。
用户并不需要真的安装配置一个真实的集群环境。HA 仿真器是开箱即用的。
只需要运行 apt 命令如下：
apt install pve-ha-simulator
用户甚至可以直接在 Debian 系统上运行该命令安装 HA 仿真器，而无需安装其他任何 Proxmox VE 软件包。只需下载软件包，然后复制到目标系统即可安装。如果使用 apt 命令在本地系统安装，依赖软件包将被自动安装。
如需通过远程启动 HA 仿真器，必须预先在本地操作系统配置好 X11 重定向。
如果你使用的是 Linux 操作系统，可以运行以下命令：
ssh root@<IPofPVE> -Y
如果使用的 windows 操作系统，可以尝试使用 mobaxterm 软件。
不管是在 Proxmox VE 服务器还是在 Debian 系统安装了 HA 仿真器后，都可以按以下步骤启动。
首先要为 HA 仿真器创建一个工作目录，以便保存当前状态并写入默认配置：
mkdir working
然后将创建的目录路径作为参数传递给 pve-ha-simulator 命令：

pve-ha-simulator working/
接下来就可以尝试启动、停止、迁移仿真 HA 服务，或检查节点故障时的现象。

14.6 配置步骤

HA 组件被紧密集成到了 Proxmox VE API 中。因此，你既能够通过 ha-manager 命令行配置 HA，也可以通过 WebGUI 配置 HA，两种方式都很简便，更进一步，还可以用自动化工具直接调用 API 配置 HA。
HA 配置文件全部保存在/etc/pve/ha/目录中，因此可以被自动复制到集群所有节点，所有节点都共享使用相同的 HA 配置。

14.6.1 资源



资源配置文件/etc/pve/ha/resources.cfg 保存了 ha-manager 管理的所有资源列表。资源列表中的资源配置格式如下：

<type>: <name>
    <property> <value>
    ...

每条资源配置信息都以冒号分隔的资源类型和资源名称开始，这也是 ha-manager 命令用于标识 HA 资源的 ID（例如 vm:100 或 ct:101），而后续配置行包含了附加属性：

comment: <string>
    描述信息。
group: <string>
    HA 组标识符。

max_relocate: <integer> (0 -N) (default = 1)

资源启动失败后尝试重新部署最大次数。

max_restart: <integer> (0 -N) (default = 1)

资源启动失败后尝试重新启动最大次数。

state: <disabled | enabled | ignored | started | stopped> (default =started)

资源的指定状态。CRM 将根据该状态值管理相关资源。请注意 enabled 是 started 的别名。

started

CRM 将尝试启动资源，并在成功启动后将状态设置为 started。如果遭遇节点故障或启动失败，CRM 将尝试恢复资源。如果所有尝试均告失败，状态将被设为 error。

stopped

CRM 将努力确保资源处于停止状态。但在遭遇节点时，CRM 还是会尝试将资源重新部署到其他节点。

disabled

CRM 将努力确保资源处于停止状态。但在遭遇节点时，CRM 不会将资源重新部署到其他节点。设置该状态的主要目的是将资源从 error 状态恢复出来，因为这是 error 状态的资源唯一可以被设置的状态。

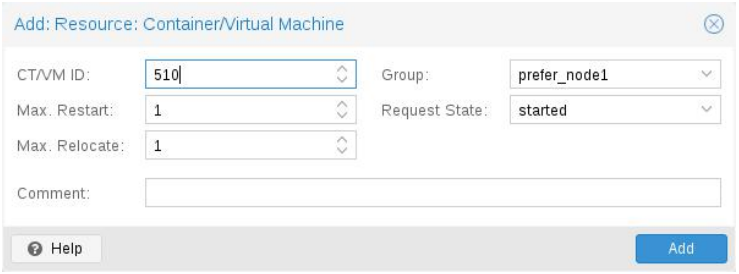
ignored

该状态表示资源不再接受 HA 管理，CRM 和 LRM 也不再管理相关资源。所有 Proxmox 将努 VE API 将绕过 HA 组件，直接对相关资源进行操作。对该资源执行的 CRM 命令将直接返回，而不做任何操作。同时，在节点发生故障时，资源也不会被自动故障转移。

以下是一个实际生产中的例子，其中包含了一个虚拟机和一个容器。可以看到，配置文件的语法其实非常简单，所以你可以用文本编辑器直接读取或修改这些配置文件：

配置示例（/etc/pve/ha/resources.cfg）

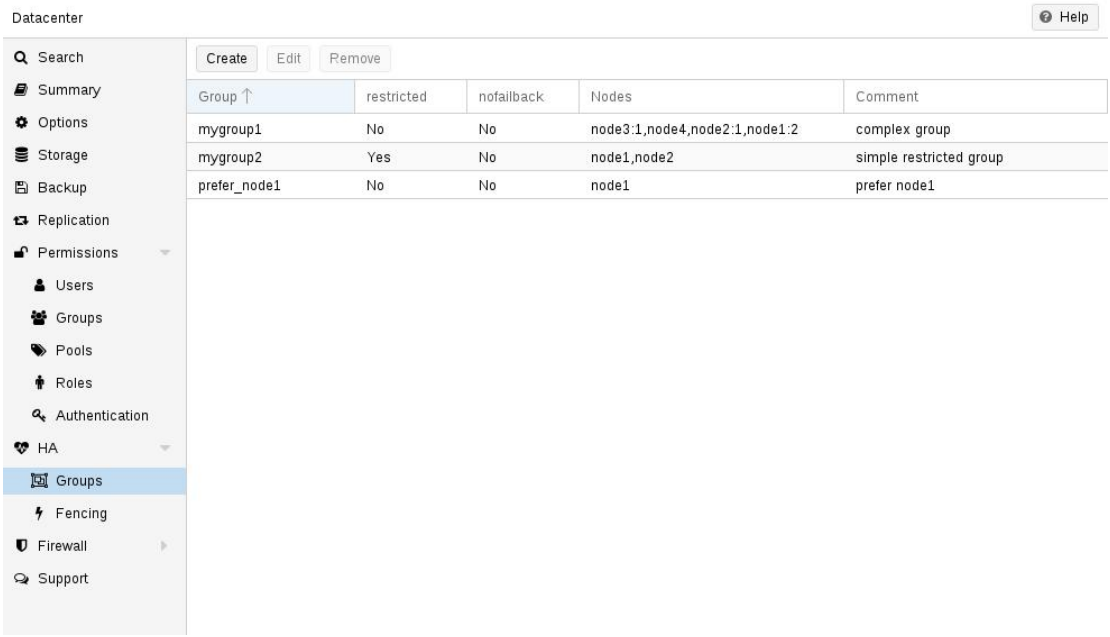
```
vm: 501
        state started
        max_relocate 2
ct: 102
        # Note: use default settings for everything
```



以上配置示例是由命令行工具 ha-manager 生成的：

```
# ha-manager add vm:501 --state started --max_relocate 2
# ha-manager add ct:102
```

14.6.2 组



HA 的组配置文件/etc/pve/ha/groups.cfg 用于定义集群节点服务器组。一个资源可以被指定只能在一个组内的节点上运行。组配置示例如下：

group: <group>
    nodes <node_list>
    <property> <value>
    ...

comment: <string>

描述信息。

nodes: <node>[:<pri>]{,<node>[:<pri>]}*

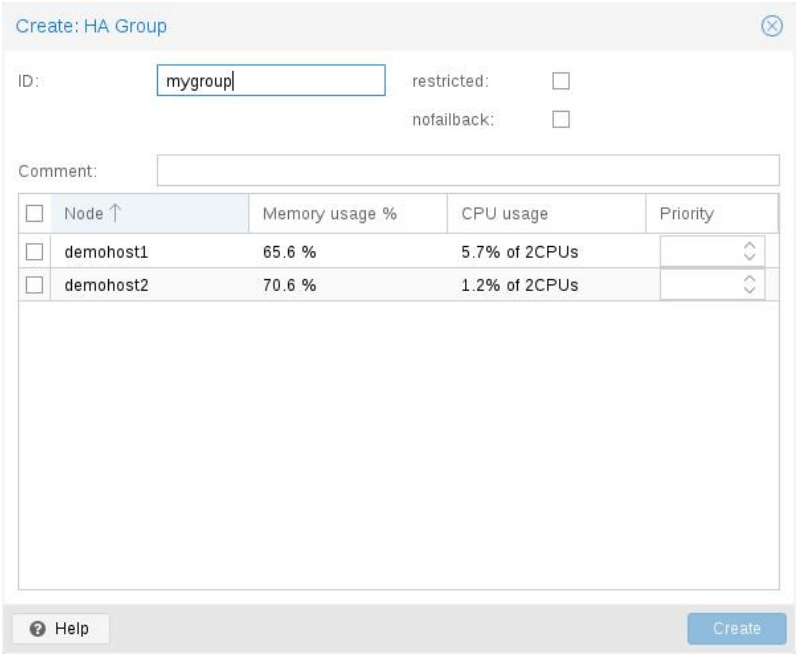
节点组成员列表，其中每个节点都可以被赋予一个优先级。绑定在一个组上的资源会优先选择在最高优先级的节点上运行。如果有多个节点都被赋予最高优先级，资源将会被平均分配到这些节点上运行。优先级的值只有相对大小意义。

nofailback: <boolean> (default = 0)

CRM 会尝试在最高优先级的节点运行资源。当有更高优先级的节点上线后，CRM 将把资源迁移到更高优先级节点。设置 nofailback 后，CRM 将继续保持资源在原节点运行。

restricted: <boolean> (default = 0)

绑定到 restricted 组的资源将只能够在该组的节点运行。如果该组的节点全部关机，则相关资源将停止运行。而对于非 restricted 组而言，如果该组的节点全部关机，相关资源可以转移到集群内的任何节点运行，一旦该组节点重新上线，相关资源会立刻迁移回到该组节点上运行。可以通过设置只有一个成员的非 restricted 组实现更好表现。



指定资源在固定节点运行是很常见的做法，但通常也会允许资源在其他节点运行。为此，你可以设置一个只有一个节点的非 restricted 组：

ha-manager groupadd prefer_node1 --nodes node1

对于节点较多的集群而言，可以考虑制定更加详尽的故障转移策略。例如，你可以指定一组资源固定在 node1 节点运行。一旦 node1 节点不可用，你可以将相关资源平均分配到 node2 和 node3 节点运行。如果 node2 和 node3 也遭遇故障，则可以进一步转移到 node4 运行。为达到该效果，你可以设置节点列表如下：

ha-manager groupadd mygroup1 -nodes "node1:2,node2:1,node3:1,node4"

另一个例子是，如果某个资源需要用到只有特定节点，比如 node1 和 node2，才具有的硬件或其他资源。我们就需要确保 HA 管理器不在其他节点运行该资源。为此，我们需要创建一个由指定节点构成的 restricted 组：

ha-manager groupadd mygroup2 -nodes "node1,node2" -restricted

以上命令创建的配置文件如下：

配置文件示例（/etc/pve/ha/groups.cfg）

```
group: prefer_node1
        nodes node1
group: mygroup1
        nodes node2:1,node4,node1:2,node3:1
group: mygroup2
        nodes node2,node1
        restricted 1
```

选项 nofailback 主要用于在管理操作中避免意外的资源迁移。例如，如果你需要将一个资源迁移到一个优先级较低的节点运行，就需要设置 nofailback 选项来告诉 HA 管理器不要立刻把资源迁移回原来的高优先级节点。

另一种可能场景是，在节点因故障被隔离后，相关资源会自动迁移到其他节点运行，而管理员在把故障节点重新恢复加入集群后，可能会希望先查明故障原因并检测该节点是否能稳定运行。这时可以设置 nofailback 选项组织 HA 管理器立刻把相关资源迁移故障节点运行。

14.7 隔离

在节点发生故障后，隔离能够确保故障节点彻底离线。这样做主要是为了避免在其他节点恢复资源运行时重复运行同一个资源。这是非常重要的，如果不能确保隔离故障节点，就不可能在其他节点安全恢复资源运行。
如果节点没有被隔离，该节点就可能处于一种不可知的状态，并仍然能够访问集群的共享资源。而这是非常危险的！想象一下这种情形，如果隔离切断了故障节点的所有网络连接，但没有切断对存储的访问，现在尽管故障节点不能再访问网络，但其上的虚拟机仍在运行，并能够向共享存储写入数据。
如果我们现在在其他节点再次启动该虚拟机，我们就可能引发危险的竞争条件，因为现在两个节点上的两个虚拟机在同时向同一个镜像写入数据。这样的情况下，很可能会损坏虚拟机的所有数据，并导致整个虚拟机不可用。当然，我们再启动同一个虚拟机的操作很可能会因为存储禁止多次挂载的保护措施而失败。

14.7.1 Proxmox VE 的隔离措施

隔离节点的方法有很多种，例如隔离设备可以切断节点电源或禁止和外部通信。但这些方法往往过于昂贵，并可能导致其他的问题，例如在隔离设备失效时就无法恢复任何服务。
因此我们采用了一种较简便的隔离方法，而没有采用任何外部隔离设备硬件。具体是采用看门狗计时器来实现。
可能的隔离措施

- 外部电源开关
- 通过在交换机禁止外部网络通信来隔离节点
- 基于看门狗的自隔离

自从微控制器诞生以来，看门狗就广泛用于重要系统和具有高可靠性要求的系统中。看门狗通常都是一块独立的简单集成电路，用于检测计算机故障并帮助从故障中恢复。
在正常情况下，ha-manager 会定期重置看门狗计时器，以防止超时。如果发生硬件故障或程序错误，计算机未能重置看门狗，计时器就会超时并触发主机重启（reboot）。
最新的服务器主板一般集成了硬件看门狗，但需要配置后才能使用。如果服务器没有配置硬件看门狗，可以退而求其次使用 Linux 内核的 softdog。软件看门狗不仅可靠，但并不独立于服务器硬件，因此可靠性较硬件看门狗低一些。

14.7.2 硬件看门狗配置

出于安全考虑，所有的硬件看门狗模块默认都是被禁止的。如果不能正确初始化，硬件看门狗就和一枝上了膛的枪一样危险。你可以在/etc/default/pve-ha-namager 中指定硬件看门狗驱动模块来启用硬件看门狗，示例如下：

# select watchdog module (default is softdog)
WATCHDOG_MODULE=iTCO_wdt
该配置将被 watchdog-mux 服务读取，并在开机时加载指定的模块。

14.7.3 恢复被隔离的服务

当节点发生故障并被成功隔离后，CRM 服务将尝试把资源从故障节点转移到其他节点运行。资源迁移目标节点的选择，由 group 资源参数配置，当前可用节点列表，各节点当前的运行负载情况共同决定。
CRM 服务首先在用户设定的节点列表（从 group 配置）和当前可用节点列表之间进行交叉比对选出可用节点列表，然后从中选择具有最高优先级的节点，最后再从中选出负载最低的节点作为目标节点。这可以资源迁移导致节点超载的可能性降到最低。

☒ 重要

发生节点故障后，CRM 会将相关资源分配给其他节点继续运行，从而使得这些节点承担更多资源的运行，有可能导致负载过高。特别在小规模集群中有可能发生这种情况。因此，请认真设计你的集群，以确保能处理这种最坏的情况。

14.8 启动失败策略

当一个服务在某节点上启动失败一次或若干次后，将按照启动失败策略进行处置。启动失败策略包括设置在同一节点的重启次数，以及转移到其他节点继续启动之前的重启次数。该策略的目标是避免共享资源临时不可用导致的启动失败。例如，由于网络问题，共享存储在某个节点上暂时不可用，但在其他节点仍然可以正常访问，转移到其他节点运行的策略将允许该资源继续运行。
对每一个服务，都有两个服务启动恢复策略参数可以配置：
max_restart
当前节点上重启失败服务的最大尝试次数。默认为 1。
max_relocate
在把服务转移到其他节点继续运行之前尝试重启失败服务的最大次数。只有在当前节点尝试重启次数超过 max_relocate 后，才会把服务转移到其他节点。默认为 1。

➢ 注意

当服务启动成功后，转移计数器会被重置为 0。也就是说，如果未能排除故障，服务继续重启，只有重启策略反复生效。

14.9 错误恢复

如果经过各种尝试都不能恢复，服务将进入 error 状态。该状态下 HA 组件将不再操作该服务。改变 error 状态的唯一方法就是手工禁用服务：

```
# ha-manager set vm:100 --state disabled
```

该操作也可以通过 WebGUI 界面进行。

从 error 状态恢复的步骤如下：

- 确保资源处于安全并一致的状态（例如：在服务不能停止时强行杀死进程）
- 禁用资源以移除 error 标识
- 修复导致错误的故障
- 排除故障后，重新启动资源。

14.10 软件包升级

升级 ha-manager 时，你应该一个节点一个节点的进行。出于多种原因，永远不要同时升级所有节点。首先，尽管我们会彻底测试 Proxmox VE，但不能确保消除一切 bug，特别在你个性化的安装环境中。逐个节点进行升级，并在升级后检查每个节点的运行情况有助于在发生意外时恢复集群。同时升级所有节点可能导致集群崩溃，并非最佳实践。

此外，Proxmox VE 的 HA 组件在集群节点和本地资源管理器之间采用了请求确认协议来传递命令。在重启时，LRM 将向 CRM 发出请求，冻结其所有服务。这将防止 LRM 重启时避免相关资源被集群访问。这样 LRM 就可以在重启时安全地关闭看门狗。LRM 重启通常发生在软件升级时，当前的主 CRM 需要确认 LRM 的请求，如果不这样做，升级过程持续的时间可能过长，并可能触发看门狗重启服务器。

14.11 节点维护

在维护节点时，例如更换硬件或安装新内核时，可以将节点关机或重启。

14.11.1 关机

关机（断电）通常在需要停止节点一段时间时使用。此时，LRM 将停止其管理的所有服务。也就是说，其他节点将接手继续运行这些服务。

➢ 注意

最新的服务器往往配置了大容量内存。所以我们先停止所有资源运行，然后在其他节点启动，以避免大量内存数据的在线迁移。如果你希望使用在线迁移，你需要在关闭节点前手工执行。

14.11.2 重启

重启节点可使用 reboot 命令。这通常在安装新内核后执行。请注意重启和“关机”的区别，重启后节点会很快恢复运行。
重启前，LRM 告诉 CRM 它希望重启，并等待 CRM 将所有资源置于 freeze 状态（也就是在软件包升级时所处于的状态，见 14.10 节）。这样相关资源就不会迁移到其他节点。想法，重启后 CRM 将在当前节点重启相关资源。

14.11.3 手工迁移资源

最后但不是唯一，你可以在关机或重启前手工把资源迁移到其他节点运行。该方式的好处是你将全程掌控资源运行状态，并且可以决定使用在线迁移或离线迁移。

➢ 注意

请不要杀死 pve-ha-crm，pve-ha-lrm 或 watchdog-mux 等服务。由于它们是基于看门狗的管理服务，这样做可能会导致服务器重启。

第 15 章 备份与恢复

备份在所有 IT 环境中都是一个非常重要的需求，Proxmox VE 内置了一个完整的备份解决方案，能够对在任意存储服务上的任意类型虚拟机进行备份。此外，系统管理员还可以通过 mode 选项在备份数据一致性和虚拟机停机时间长度之间进行取舍。

Proxmox VE 目前只支持全备份 – 包括虚拟机/容器的配置以及全部数据。备份命令可以通过 WebGUI 或 vzdump 命令行工具发出。

备份存储

在进行备份前，首先要定义一个备份用存储服务。关于添加存储服务的步骤，可以参考存储服务相关章节。鉴于备份采用文件形式保存，备份用存储必须是文件级存储服务。大部分情况下，NFS 服务器是备份用存储的良好选择。备份虚拟机后，你可以进一步将相关文件保存在磁带上以用于离线归档。

调度备份

也可以调度方式执行备份操作，以便在指定的日期和时间自动备份指定节点上的虚拟机。调度备份的配置可在 WebGUI 中的数据中心配置界面进行，配置的调度任务会自动保存到 /etc/cron.d/vzdump 文件中。

15.1 备份模式

根据备份对象的种类，有多种数据一致性模式（mode）可以选择：

虚拟机备份

stop 模式

该模式能提供最强的数据一致性保障，代价是备份过程中虚拟机要停机。该模式的执行流程依次是，停止虚拟机运行，后台执行 Qemu 进程备份虚拟机数据。备份完成后，Qemu 进程将虚拟机恢复到备份开始前的运行状态。通过 live backup 特性可以保证数据一致性。

suspend 模式

提供该模式的唯一原因是兼容性。该模式首先会挂起虚拟机，然后执行 snapshot 模式。鉴于该模式会挂起虚拟机，导致虚拟机长时间停止运行，而同时并没有改进备份数据一致性，因此建议直接使用 snapshot 模式。

snapshot 模式

采用该模式虚拟机停机时间最短，代价是备份数据有可能不一致。该模式实际上采用的是 Proxmox VE 在线备份，也就是在虚拟机运行状态下复制数据。如果启用了 guest agent（agent:1），该模式将调用 guest-fsfreeze-freeze 和 guest-fsfreeze-thaw 以改进数据一致性。

可点击此处查看 Proxmox VE 对 QemuServer 在线备份的技术概览资料。

➢ 注意

Proxmox VE 在线备份技术对任意类型存储服务上的虚拟机可以进行类似 snapshot 形式的备份。但并不需要底层存储服务支持 snapshot 功能。另外请注意，备份操作是由后台 Qemu 进程完成的，尽管虚拟机可能已停止运行，但在 Qemu 读取虚拟机磁盘的过程中，虚拟机状态仍会显示为运行。但此时只有虚拟机磁盘有读取动作，虚拟机本身并没有运行。

容器备份

stop 模式

备份过程中停止容器运行。该模式可能导致较长的停机时间。

suspend 模式

该模式利用 rsync 将容器数据复制到一个临时位置（参看选项--tmpdir）。之后将挂起容器，并再次调用 rsync 同步复制之前复制过程中改变的文件。完成后将恢复容器运行。该模式下的停机时间较短，但需要额外的空间来保存容器备份。

当容器位于服务器本地磁盘，而备份目标位置在外部 NFS/CIFS 服务器上时，你应该设置—tmpdir 将临时位置指定在本地磁盘上，这样能大大提高性能。此外，在将配置了 ACLs 的本地磁盘容器备份到外部 NFS 服务器上时，必须设置 tmpdir 为本地磁盘目录。

snapshot 模式

采用该模式需要底层存储服务的 snapshot 功能支持。首先，容器会被挂起以确保备份数据一致性。然后将为容器所在存储卷创建一个临时快照，该快照会被打包到一个 tar 文件。备份完成后，临时快照会被删除。

➢ 注意

Snapshot 模式要求被备份存储卷所在存储服务支持 snapshot。可以设置挂载点选项 backup=no 将指定存储卷排除在备份范围之外（同时排除对相关存储支持 snapshot 功能的要求）。

➢ 注意

默认配置下，只有根磁盘挂载点会被备份，其他附加挂载点不会被备份。可以设置挂载点的 Backup 参数将附加挂载点纳入备份范围。Device 和 bind 并未纳入 Proxmox VE 存储库的管理，所以也不会被备份。

15.2 备份文件命名

新版 vzdump 将利用虚拟机类型和备份时间编码备份文件名称，示例如下：

vzdump-lxc-105-2009_10_09-11_04_43.tar

这样就可以在同一目录下保存同一虚拟机的多个备份文件。可以设置参数 maxfiles 指定同一

虚拟机最大备份文件数量。

15.3 恢复

可以在 Web GUI 或通过如下命令恢复备份文档。

pct restore

容器恢复命令

qmrestore

虚拟机恢复命令

详情可查看相应的手册。

15.3.1 恢复限速

恢复大型备份文件是非常耗费资源的，特别是从读取备份存储和写入目标存储的操作，会给存储带来很大压力，并挤占其他虚拟机对存储的访问请求，影响其他虚拟机的正常运行。为避免该问题，可以对备份任务设置限速。Proxmox VE 为备份恢复提供了两种限速：

- **读限速：用于限制从备份存储读取的最大速度。**
- **写限速：用于限制向指定存储写入的最大速度。**

读限速间接影响写限速，因为备份恢复过程中，写入数据量不可能超出读取数据量。因此较低的读限速将覆盖较高的写限速。只有在目标存储设置了‘Data.Allocate’权限时，较高的读限速才会覆盖写限速。

在恢复命令行中可以使用‘—bwlimit <integer>’参数来设置特定恢复任务的限速。限度单位为 Kibit/s，也就是说，设置为‘10240’时相当于读限速 10MiB/s，剩余带宽可供其他虚拟机使用，从而确保正常运行。

➢ **注意**

可以设置 bwlimit 为’0’，禁用限速。这可以帮助你尽快恢复重要虚拟机。（存储需要设置‘Data.Allocate’权限）

大多数情况下，存储可用读写带宽是保持不变的。所以可以为每个存储设置默认限速。参考命令如下：

```
# pvesm set STORAGEID --bwlimit KIBs
```

15.4 配置文件

全局配置信息保存在/etc/vzdump.conf。该文件采用了冒号分隔的键/值配置格式。例子如下：

```
OPTION: value
```

空行会被自动忽略，以#开头的行将按注释处理，也会被自动忽略。该文件中的配置值被用

作默认配置，如在命令行中指定了新值，则默认值将被覆盖。
目前支持的选项如下：

bwlimit : <integer> (0 -N) (default = 0)

I/O 带宽上限（单位 KB/秒）。

compress : <0 | 1 | gzip | lzo> (default = 0)

备份文件压缩设置。

dumpdir : <string>

指定备份文件保存位置。

exclude-path : <string>

排除指定的文件/目录（shell 全局）。

ionice : <integer> (0 -8) (default = 7)

设置 CFQ ionice 优先级。

lockwait : <integer> (0 -N) (default = 180)

等待全局锁的最长时间（单位为分钟）。

mailnotification : <always | failure> (default = always)

设置电子邮件通知发送时机。

mailto : <string>

电子邮件通知发送地址列表，分隔符为逗号。

maxfiles : <integer> (1 -N) (default = 1)

保存的单一虚拟机备份文件最大数量。

mode : <snapshot | stop | suspend> (default = snapshot)

备份模式。

pigz : <integer> (default = 0)

设置 N>0 时，用 pigz 代替 gzip 进行压缩。设置 N=1 将使用服务器一半数量的核心，设置 N>1 将使用 N 个核心。

pool: <string>

备份资源池中的所有虚拟机。

remove : <boolean> (default = 1)

当备份文件数量超过 maxfiles 时，自动删除最老的备份文件。

script : <string>

启用指定的勾子脚本。

stdexcludes : <boolean> (default = 1)

排除临时文件和日志数据。

stopwait : <integer> (0 -N) (default = 10)

等待虚拟机停止运行的最长时间（单位为分钟）。

storage : <string>

指定备份文件保存位置。

tmpdir : <string>

指定临时文件保存位置。

vzdump.conf 配置示例

```
tmpdir: /mnt/fast_local_disk
storage: my_backup_storage
mode: snapshot
```

bwlimit: 10000

15.5 勾子脚本

你可以设置参数--script 指定勾子脚本。根据设置的参数不同，该脚本将在备份过程的不同阶段被调用。你可以在文档目录中找到使用范例（vzdump-hook-script.pl）。

15.6 排除文件

➢ **注意**

该选项仅对容器备份有效。

默认 vzdump 会忽略以下文件（可设置参数--stdexcludes 0 禁用）
/tmp/? *
/var/tmp/? *
/var/run/? * pid
你也可以手工指定（增加）排除目录，例如：
vzdump 777 --exclude-path /tmp/ --exclude-path '/var/foo *'
（仅忽略 tmp 目录）
配置文件也会被保存到备份文件中（在./etc/vzdump），并在恢复时同步恢复。

15.7 例子

备份 777 号虚拟机 。不带快照，仅将虚拟机私有数据和配置文件保存到默认备份目录（通常为/var/lib/vz/dump/）。
vzdump 777
使用挂起/恢复模式创建临时快照，并使用 rsync 备份（最小化停机时间）。
vzdump 777 --mode suspend
备份所有虚拟机并向 root 和 admin 发送邮件通知。
vzdump --all --mode suspend --mailto root --mailto admin
使用 snapshot 模式（不停机）和非默认备份目录。
vzdump 777 --dumpdir /mnt/backup --mode snapshot
备份多个（指定的）虚拟机。
vzdump 101 102 103 --mailto root
备份除 101 和 102 号以外的所有虚拟机。
vzdump --mode suspend --exclude 101,102
恢复备份文件到新创建的 600 号容器。
pct restore 600 /mnt/backup/vzdump-lxc-777.tar
恢复备份文件到新建的 601 号 QemuServer 虚拟机。

qmrestore /mnt/backup/vzdump-qemu-888.vma 601
利用管道将 101 号容器克隆到新建 300 号容器，并设置根文件系统容量为 4GB。
vzdump 101 --stdout | pct restore --rootfs 4 300 -

第 16 章 重要服务

16.1 pvedaemon – Proxmox VE API 守护进程

该守护进程在 127.0.0.1:85 上提供了 Proxmox VE API 的调用接口。该进程以 root 权限运行，能够执行所有特权操作。

➢ 注意

该守护进程仅监听本地地址，外部无法直接访问。守护进程 pveproxy 负责向外部提供 API 调用接口。

16.2 pveproxy – Proxmox VE API 代理进程

该进程通过 HTTPS 在 TCP 8006 端口向外部提供 Proxmox VE API 调用接口。该进程以 www-data 权限运行，因此权限非常有限。更高权限的操作将由本地的 pvedaemon 进程执行。
指向其他节点的操作请求将自动发送到对应节点，也就是说你可以从 Proxmox VE 的一个节点管理整个集群。

16.2.1 基于主机的访问控制

可以为 pveproxy 配置类似于“apache2”的访问控制列表。相关访问控制列表保存在 /etc/default/pveproxy 中。例如：
ALLOW_FROM="10.0.0.1-10.0.0.5,192.168.0.0/22"
DENY_FROM="all"
POLICY="allow"
IP 地址可以用类似 Net::IP 的语法指定，而 all 是 0/0 的别名。
默认策略是 allow。

匹配情况	POLICY=deny	POLICY=allow
仅有 Allow 匹配上	允许访问	允许访问
仅有 Deny 匹配上	拒绝访问	拒绝访问
均未匹配上	拒绝访问	允许访问
同时匹配到 Allow 和 Deny	拒绝访问	允许访问

16.2.2 SSL 加密套件

可以在配置文件/etc/default/pveproxy 中指定密码列表。例如：
CIPHERS="ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256"
以上是默认配置。可以查看 openssl 软件包中的 man 页面 ciphers(1)获取更多可用选项。
此外，可以设置客户端使用/etc/default/pveproxy 中的指定密码（默认使用列表中第一个可以同时为 client 和 pveproxy 接受的）
HONOR_CIPHER_ORDER=0

16.2.3 Diffie-Hellman 参数

可以在配置文件/etc/default/pveproxy 中指定 Diffie-Hellman 参数。只需将参数 DHPARAMS 设置为包含 DH 参数的 PEM 文件路径即可。例如：
DHPARAMS="/path/to/dhparams.pem"
如未设置该参数，将使用内置的 skip2048 参数。

➢ **注意**

DH 参数仅在协商使用基于 DH 密钥交换算法的加密套件时有效。

16.2.4 其他 HTTPS 证书

Proxmox VE 可以改用外部证书，或 ACME 证书。
Pveproxy 默认使用证书/etc/pve/local/pve-ssl.pem 和/etc/pve/local/pve-ssl.key，如果以上

文件不存在，将改用/etc/pve/local/pveproxy-ssl.pem 和/etc/pve/local/pveproxy-ssl.key。详细信息可以查看第 3 章 Proxmox VE 服务器管理。

16.2.5 压缩

在客户端支持的情况下，默认 pveproxy 使用 gzip 对 HTTP 流量进行压缩。可以在/etc/default/pveproxy 中禁用该功能
COMPRESSION=0

16.3 pvestatd – Proxmox VE 监控守护进程

该守护进程定时获取虚拟机、存储和容器的状态数据。结果将自动发送到集群中的所有节点。

16.4 spiceproxy – SPICE 代理进程

SPICE（Simple Protocol for Independent Computing Environments）是一个开源远程计算解决方案，能够为远程桌面和设备（例如键盘、鼠标、音频）的提供客户端访问接口。主要使用场景是访问远程虚拟机和容器。
该守护进程监听 TCP 3128 端口，并通过 HTTP 代理将 SPICE 客户端的连接请求转发给相应的 Proxmox VE 虚拟机。该进程以 www-data 权限运行，权限非常有限。

16.4.1 基于主机的访问控制

可以为 spice 配置类似于“apache2”的访问控制列表。相关访问控制列表保存在/etc/default/pveproxy 中。详情可查看 pveproxy 文档。

第 17 章 命令行工具

17.1 pvesubscription – 订阅管理工具

Promxox VE 订阅管理工具。

17.2 pveperf – Proxmox 性能测试脚本

用于收集 CPU/硬盘性能数据的工具。硬盘可用对应的文件系统路径 PATH 指定（默认为/）：

CPU BOGOMIPS
所有 CPU 的 bogomips 总和。

REGEX/SECOND
每秒正则表达式（perl 性能测试）测试结果。应大于 300000。

HD SIZE
硬盘容量。

BUFFERED READS
简单硬盘读测试。主流硬盘至少应达到 40MB/秒。

AVERAGE SEEK TIME
平均寻道时间测试。快速 SCSI 硬盘应<8 毫秒。一般 IDE/SATA 硬盘应在 15-20 毫秒。

FSYNCS/SECOND
该测试值应高于 200（如使用 RAID 卡，在具备后备电池缓存时，应启用 writeback 缓存模式）。

DNS EXT
解析外部 DNS 域名的平均时间。

DNS INT
解析本地 DNS 域名的平均时间。

17.3 Proxmox VE API 的命令行工具

Proxmox VE 管理工具（pvesh）可以直接调用 API 函数，无需通过 REST/HTTPS 服务器。

➢ 注意

只有 root 用户有此权限。

17.3.1 示例

显示集群节点列表
#pvesh get /nodes
显示数据中心可用选项
#pvesh usage cluster/options -v
将 HTML5 NoVNC 控制台设置为数据中心默认控制台
#pvesh set cluster/options -console html5

17.4 Proxmox 节点管理

Proxmox VE 节点管理工具（pvenode）可以管理节点特定设置和资源。目前，pvenode 可以设置节点描述信息，并通过 pveproxy 管理节点的 SSL 证书。

17.4.1 示例

pvenode cert set certificate.crt certificate.key -force
以上命令用于安装外部证书文件。两个文件都需要使用 PEM 编码形式。certificate.key 是私钥文件，certificate.crt 包含证书链信息。
pvenode acme account register default mail@example.invalid
pvenode config set --acme domains=example.invalid
pvenode acme cert order
systemctl restart pveproxy
以上命令用于建立 ACME 账户，并为本地节点设置证书。

17.4.2 局域网唤醒

局域网唤醒可以通过网络发送一个魔术数据将计算机从睡眠状态唤醒。该功能需要至少拥有一个支持局域网唤醒的网卡，并在计算机固件（BIOS/UEFI）配置相应参数以启用。常见参数名有多种可能，包括 Enable Wake-on-Lan，Power On By PCIE Device 等，如不确定名称，请查看主板厂商手册。可使用 ethtool 工具检查网卡<interface>的局域网唤醒配置，示例如下：
Ethtool <interface> | grep Wake-on
命令行 pvenode 支持通过局域网唤醒集群内的睡眠节点，示例如下：
pvenode wakeonlan <node>
该命令将通过 UDP 19 端口广播局域网唤醒魔术数据包，数据包含有<node>节点的 MAC 地址。节点对应的 wakeonlan 属性可以通过以下命令设置：
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX

第 18 章 常见问题

➢ 注意

更新的常见问题将在追加在本节最后。

---

1. Proxmox VE 基于哪个发行版？

   Proxmox VE 基于 Debian GNU/Linux。

2. Proxmox VE 项目采用哪种开源协议？

   Promxox VE 代码采用开源协议 GNU Affero General Public License，version 3。

3. Proxmox VE 支持 32 位 CPU 么？

   Proxmox VE 仅支持 64 位 CPU（AMD 或 Intel）。目前没有计划支持 32 位 CPU。

➢ 注意

虚拟机和容器可以采用 32 位或 64 位操作系统。

---

4. 我的 CPU 支持虚拟化么？

   检测 CPU 的虚拟化兼容性，可用以下命令检测 vmx 或 svm 标记

   egrep '(vmx|svm)' /proc/cpuinfo

5. 支持的 Intel CPU 列表

   支持 Intel 虚拟化技术（Intel VT-x）的 64 位 CPU。（同时支持 Intel VT 和 64 位的 IntelCPU 列表）

6. 支持的 AMD CPU

   支持 AMD 虚拟化技术（AMD-V）的 64 位 CPU。

7. 容器，CT，VE，虚拟个人服务器，VPS 都是什么？

操作系统虚拟化是一种服务器虚拟化技术，也就是利用一个操作系统内核同时运行多个彼此隔离的操作系统用户空间实例，而不是仅运行一个操作系统用户空间实例。我们将每个实例称为容器。由于共享操作系统内核，容器仅限于运行 Linux 系统。

8. QEMU/KVM 客户机（或 VM）是什么？

   QEMU/KVM 客户机（或 VM）是一个虚拟化客户机系统，利用 QEMU 和 Linux KVM 内核模块运行在 Proxmox VE 上。

9. QEMU 是什么？

   QEMU 是一个通用的开源模拟器和虚拟化软件。QEMU 利用 Linux KVM 内核模块直接在主机 CPU 运行客户机代码，从而获得接近于本地物理服务器的效率和性能。QEMU 不仅能运行 Linux 客户机，还能运行任意操作系统客户机。

10. 各版本的 Proxmox VE 最终支持期限是？

PVE 版本	Debian 版本	发行日期	Debian EOL	Proxmox EOL
6.x	Debian 10（buster）	2019-07	tba	tba
5.x	Debian 9（stretch）	2017-07	2020-07	2020-07
4.x	Debian 8（jessie）	2015-10	2018-06	2018-06
3.x	Debian 7（Wheezy）	2013-05	2016-04	2017-02
2.x	Debian 6（Squeeze）	2012-04	2014-05	2014-05
1.x	Debian 5（Lenny）	2008-10	2012-03	2013-01

11. 如何升级 Proxmox VE

小版本升级，例如从 Proxmox VE 5.1 升级到 5.2，可以按日常升级操作进行。具体可以通过 Web GUI 的 Node→Updates 控制面板进行，也可以执行以下命令

apt update

apt full-upgrade

➢ 注意

无论何时，在正式升级前都请务必确保按照 3.1 节正确设置了软件源，并且 apt update 命令没有任何报错。

大版本升级，例如从 Proxmox VE 4.4 升级到 5.0，也是可以做到的，但是必须要预先进行充分的准备，包括认真规划升级方案，测试方案可行性，做好备份。永远不要在未做备份的情况下升级。根据部署情况的不同，具体升级过程会有个性化步骤，但官方还是有一个一般性的升级建议方案：

- 从 Proxmox VE 5.x 升级到 6.0
- 从 Proxmox VE 4.x 升级到 5.0
- 从 Proxmox VE 3.x 升级到 4.0

12. LXC vs LXD vs Proxmox 容器 vs Docker

LXC 是 Linux 内核容器的用户空间接口。通过强大的 API 和易用的工具，Linux 用户能够轻松地创建并管理系统容器。LXC，及其前任 OpenVZ，专注于系统虚拟化，也就是让你在容器内运行完整的操作系统，其中你可以 ssh 方式登录，增加用户，运行 apache 服务器等。

LXD 基于 LXC 创建，并提供了更好的用户体验。在底层，LXD 通过 liblxc 调用 LXC 及其 Go 绑定来创建和管理容器。LXD 基本上是 LXC 工具和模板系统的的另一个选择，只是增加了诸如远程网络控制等新的特性。

Proxmox 容器也专注于系统虚拟化，并使用 LXC 作为其底层服务。Proxmox 容器工具称为 pct，并和 Proxmox VE 紧密集成在一起。这意味着 pct 能够利用集群特性，并像虚拟机那样充分利用相同的网络和存储服务。你甚至可以使用 Proxmox VE 防火墙，备份和恢复，设置容器 HA。可以使用 Proxmox VE API 通过网络管理容器的全部功能。

Docker 专注于在容器内运行单一应用。你可以用 docker 工具在主机上管理 docker 实例。但不推荐直接在 Proxmox VE 主机上运行 docker。

- 注意

  你可以在 Proxmox Qemu 虚拟机内部安装使用 docker，这样既可以享受软件容器带来的便利，也可以得到虚拟机的强隔离特性。

第 19 章 参考文献

19.1 Proxmox VE 技术书籍

[1] [Ahmed16] Wasim Ahmed. Mastering Proxmox - Third Edition. Packt Publishing, 2017. ISBN 978-1788397605

[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

19.2 相关技术的书籍

[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 Pub-lications 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] [Singh16] Karan Signh. Ceph Cookbook Packt Publishing, 2016. ISBN 978-1784393502

[12] [Mauerer08] Wolfgang Mauerer. Professional Linux Kernel Architecture. John Wiley & Sons, 2008. ISBN 978-0470343432

[13] [Loshin03] Pete Loshin, IPv6: Theory, Protocol, and Practice, 2nd Edition. Morgan Kauf-mann, 2003. ISBN 978-1558608108

[14] [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

[15] [Kreibich10] Jay A. Kreibich. Using SQLite, O'Reilly and Associates, 2010. ISBN 978-0596521189

19.3 相关主题的书籍

[16] [Bessen09] James Bessen & Michael J. Meurer, Patent Failure: How Judges, Bureaucrats, and Lawyers Put Innovators at Risk. Princeton Univ Press, 2009. ISBN 978-0691143217