Promxox VE 中文文档

发行版本 *7.1*

bingsin

2023 年 02 月 01 日

Contents

本手册是网友自发翻译，属于第三方性质，请知晓。

原版文档地址为：https://pve.proxmox.com/pve-docs/pve-admin-guide.html

由于 PVE 官方文档一是英文，二是没有太多示例，导致入门很困难。所以一直想做一个中文，且又有更多示例说明（文本或者多媒体）的教程。以便更能了解 PVE，使用 PVE。

所以整个流程为——> 初步翻译——> 校对——> 确认完稿——> 添加额外教程在初步翻译过程中，会有很多错误，请谅解。

欢迎各位 PVE USER，对本翻译反馈各种问题和建议。如果可以，也可以参与到翻译中。本项目 github 地址：https://github.com/jiangcuo/pve-doc-cn QQ 群：90475453

翻译版本	贡献人	完成时间	对应英文版本
V1.0	🄵眼镜	2017-02-16	V4.4 December 9, 2016
V1.01	Tinkering	2017-06-13	V4.4 December 9, 2016
V5.2.0	🄵眼镜	2019-02-20	V5.2 May 16, 2018
V5.3.0	🄵眼镜	2019-06-03	V5.3 November 29, 2018
V5.4.0	🄵眼镜	2019-06-18	V5.4 April 8, 2019
V6.0.0	🄵眼镜	2019-12-10	V6.0 July 15, 2019
V6.2.0	soraelric	2020-09-15	V6.2 July 17,2020
v7.1.0	jiangcuo	2022-10-17	v7.1.0 Nov 15,2021

感谢参与此项目的所有人。

CHAPTER 1

第一章介绍

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并使用 Proxmox VE。

1.1 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。

1.1.1 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。

1.1.2 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。

1.1.3 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。

1.1.4 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。

1.1.5 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。

1.1.6 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。

1.1.7 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 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 1.3 虚拟机备份和恢复

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 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 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 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 1.7 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。

1.7.1 1.7.1 Proxmox 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。

1.7.2 1.7.2 超融合基础设施：储存

Proxmox VE 紧密集成了对部署超融合存储基础架构的支持。例如，您可以仅使用 Web 界面部署和管理以下两种存储技术：

- ceph：既可自我修复又可自我管理的共享、可靠且高度可扩展的存储系统。了解如何在 Proxmox VE 节点上管理 ceph 服务第 8 章。
- ZFS：组合的文件系统和逻辑卷管理器，具有针对数据损坏、各种 RAID 模式、快速且低成本的快照等功能的广泛保护。了解如何在 Proxmox VE 节点上利用 ZFS 的功能第 3.8 节。

除此之外，Proxmox VE 还支持集成多种附加存储技术。您可以在存储管理器第 7 章中找到有关它们的信息。

1.8 1.8 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。

1.9 1.9 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

1.10 1.10 获取支持

1.10.1 1.10.1 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。

1.10.2 1.10.2 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。

1.10.3 1.10.3 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

1.10.4 1.10.4 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。

1.10.5 1.10.5 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。

1.11 1.11 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。

1.12 1.12 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。

CHAPTER 2

第二章 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 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 台虚拟机就会全部宕机。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 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 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 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 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 2.2 使用安装介质

可以从 http://www.proxmox.com/en/downloads/category/iso-images-pve 下载 ISO 镜像。

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。

2.2.1 2.2.1 使用一个 U 盘作为安装介质

U 盘需要至少有 1 GB 的可用存储。

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。

2.2.2 2.2.2 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。

2.2.3 2.2.3 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。

2.2.4 2.2.4 Windows 下的制作过程

使用 Etcher

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

使用 Rufus

Rufus 是一个更轻量级的替代方案，但您需要使用 DD 模式才能使其工作。从 https://rufus.ie/下载 Rufus。要么直接安装，要么使用便携版本。选择目标驱动器和 Proxmox VE ISO 文件。

重要启动后，您必须在要求下载不同版本的 GRUB 的对话框中单击否。在下一个对话框中，选择 DD 模式。

CHAPTER 3

第三章系统管理

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 3.1 软件源

Proxmox VE 像其他 debian 发行版一样，使用 APT 作为软件包管理器。

3.1.1 3.1.1. Proxmox VE 的软件仓库

软件仓库收集了大量的软件，他们可以用来安装新程序或者更新旧程序。

注意需要有效的 Debian 和 Proxmox 仓库才能获得安全更新、bug 修复程序和新功能

/etc/apt/sources.list 文件和/etc/apt/sources.list.d/目录下的.list 文件，定义了软件仓库源列表。

软件仓库管理

从 Proxmox VE 7.0 以来，用户可以在网页上检查仓库状态。在节点摘要面板显示一个高级的状态视图，而仓库选项中可以看到更加详细的软件源列表、配置和状态。这里仅是基础的软件仓库管理，比如启用和禁用软件仓库。

Sources.list

软件源文件 sources.list 的每一行定义了一个软件源，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。可以通过命令 apt-get 获取更新或者在 GUI 面板上点击节点——更新。

/etc/apt/sources.list 文件

```
deb http://ftp.debian.org/debian bullseye main contrib
deb http://ftp.debian.org/debian bullseye-updates main contrib

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

Proxmox VE 提供了 3 个不同的软件仓库。

3.1.2 3.1.2. 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/pve bullseye pve-enterprise
```

root@pam 用户将通过电子邮件收到有关可用更新的通知。单击 GUI 中的”更改日志”按钮可查看有关所选更新的更多详细信息。

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。

注意您可以通过使用 # 注释掉上面的行（在行首）来禁用此存储库。这可以防止在没有订阅密钥时出现错误消息。在这种情况下，请配置 pve-no-subscription 存储库。

3.1.3 3.1.3. Proxmox VE 无订阅储存库

这是推荐用于测试和非生产用途的存储库。它的软件包没有经过严格的测试和验证。您不需要订阅密钥即可访问 pve-no-subscription 存储库。

我们建议在 /etc/apt/sources.list 中配置此存储库。

/etc/apt/sources.list 文件

```
deb http://ftp.debian.org/debian bullseye main contrib
deb http://ftp.debian.org/debian bullseye-updates main contrib

# Proxmox VE 无订阅储存库由proxmox.com提供,
# 不建议在生产环境中使用
deb http://download.proxmox.com/debian/pve bullseye pve-no-subscription

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

3.1.4 3.1.4. Proxmox VE 测试存储库

此存储库包含最新的包，主要由开发人员用于测试新功能。要配置它，请将以下行添加到 /etc/apt/sources.list

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

仅用于测试新功能或错误修复。

3.1.5 3.1.5. Ceph 太平洋仓库

Ceph Pacific（16.2）在 Proxmox VE 7.0 中被宣布稳定

此存储库包含主要的 Proxmox VE Ceph Pacific 软件包。它们适用于生产。如果您在 Proxmox VE 上运行 Ceph 客户端或完整的 Ceph 集群，请使用此存储库。

`/etc/apt/sources.list.d/ceph.list` 文件

```
deb http://download.proxmox.com/debian/ceph-pacific bullseye main
```

3.1.6 3.1.6. Ceph Pacific 测试仓库

此 Ceph 存储库包含 Ceph Pacific 软件包，然后再将其移动到主存储库。它用于在 Proxmox VE 上测试新的 Ceph 版本。`/etc/apt/sources.list.d/ceph.list`

```
deb http://download.proxmox.com/debian/ceph-pacific bullseye test
```

3.1.7 3.1.7. Ceph Octopus 仓库

Ceph Octopus（15.2）在 Proxmox VE 6.3 中宣布稳定，它将继续在 6.x 版本的剩余生命周期内获得更新，并且 Proxmox VE 7.x 将继续获得更新，直到 Ceph Octopus 上游 EOL（🄵2022-07）

此储存库包含主要的 Proxmox VE Ceph Octopus 软件包。它们适用于生产。如果您在 Proxmox VE 上运行 Ceph 客户端或完整的 Ceph 集群，请使用此存储库。

`/etc/apt/sources.list.d/ceph.list`

```
deb http://download.proxmox.com/debian/ceph-octopus bullseye main
```

请注意，在较旧的 Proxmox VE 6.x 上，您需要在上面的存储库规范中 bullseye 将更改为 buster

3.1.8 3.1.8. Ceph Octopus 测试仓库

此 Ceph 存储库包含 Ceph 包，然后再将其移动到主存储库。它用于在 Proxmox VE 上测试新的 Ceph 版本。

`/etc/apt/sources.list.d/ceph.list`

```
deb http://download.proxmox.com/debian/ceph-octopus bullseye test
```

3.1.9 3.1.9. 安全安装

存储库中的发布文件使用 GnuPG 进行签名。APT 正在使用这些签名来验证所有包是否都来自受信任的源。

如果您从官方 ISO 映像安装 Proxmox VE，则验证密钥已安装。

如果您在 Debian 之上安装 Proxmox VE，请使用以下命令下载并安装密钥：

`# wget https://enterprise.proxmox.com/debian/proxmox-release-bullseye.gpg -O /etc/apt/trusted.gpg.d/proxmox-release-bullseye.gpg` 之后使用 sha512sum CLI 工具验证校验和：

```
# sha512sum /etc/apt/trusted.gpg.d/proxmox-release-bullseye.gpg
7fb03ec8a1675723d2853b84aa4fdb49a46a3bb72b9951361488bfd19b29aab0a789a4f8c7406e71a69aabbc727c936d3549
↪/etc/apt/trusted.gpg.d/proxmox-release-bullseye.gpg
```

或 md5sum CLI 工具：

```
# md5sum /etc/apt/trusted.gpg.d/proxmox-release-bullseye.gpg
bcc35c7173e0845c0d6ad6470b70f50e /etc/apt/trusted.gpg.d/proxmox-release-bullseye.gpg
```

3.2 3.2. 系统软件更新

Proxmox 定期为所有存储库提供更新。要安装更新，请使用基于 Web 的 GUI 或以下 CLI 命令：

```
# 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。

3.3 3.3. 网络配置

网络配置可以通过 GUI 完成，也可以通过手动编辑包含整个网络配置的文件`/etc/network/interfaces` 来完成。interfaces(5) 手册页包含完整的格式说明。所有 Proxmox VE 工具都努力保持直接的用户修改，但使用 GUI 仍然是可取的，因为它可以保护您免受错误的影响。

一旦配置了网络，您可以使用 Debian 传统工具 ifup 和 ifdown 命令来启动和关闭接口。

3.3.1 3.3.1. 应用网络更改

Proxmox VE 不会将更改直接写入 /etc/network/interfaces· 。相反，我们写入一个名为/etc/network/interfaces.new‘的临时文件，这样你就可以一次做许多相关的更改。这还允许在应用之前确保您的更改是正确的，因为错误的网络配置可能会导致节点无法访问。

重启生效

使用默认安装的 ifupdown 网络管理包，您需要重新启动才能提交任何挂起的网络更改。大多数时候，基本的 Proxmox VE 网络设置是稳定的，不会经常更改，因此不需要经常重新启动。

使用 ifupdown2 重新加载网络

使用可选的 ifupdown2 网络管理软件包，您还可以实时重新加载网络配置，而无需重新启动。

从 Proxmox VE 6.1 开始，您可以使用节点的" 网络" 面板中的" 应用配置" 按钮，通过 Web 界面应用挂起的网络更改。

要安装 ifupdown2 确保您安装了最新的 Proxmox VE 更新，然后

安装 ifupdown2 将删除 ifupdown，但由于版本 0.8.35+pve1 之前的 ifupdown 的删除脚本存在一个问题，即网络在删除时完全停止 [1] 您必须确保您拥有最新的 ifupdown 软件包版本。

对于安装本身，您可以简单地执行以下操作：

apt install ifupdown2 有了它，一切就绪。如果遇到问题，您也可以随时切换回 ifupdown 变体。

3.3.2 3.3.2 网卡命名规范

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[n<phys_port_name>|d<dev_port>]—板载设备命名
- s[f][n<phys_port_name>|d<dev_port>]—按设备热插拔 ID 命名
- [P]ps[f][n<phys_port_name>|d<dev_port>]—按设备总线 ID 命名
- x—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。

3.3.3 3.3.3 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。

3.3.4 3.3.4 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
```

(续下页)

(接上页)

```
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。

3.3.5 3.3.5 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
```

(续下页)

(接上页)

```
netmask 255.255.255.248
bridge_ports none
bridge_stp off
bridge_fd 0
```

3.3.6 3.3.6 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
```

3.3.7 3.3.7 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

```

(续下页)

(接上页)

```
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
```

3.3.8 3.3.8 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
```

(续下页)

(接上页)

```
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
```

3.3.9 3.3.9. 禁用 IPV6

Proxmox VE 在所有环境中都能正常工作，无论是否部署了 IPv6。我们建议将所有设置保留为提供的默认值。

如果仍需要在节点上禁用对 IPv6 的支持，请通过创建适当的 sysctl.conf （5）代码段文件并设置正确的 sysctl 来执行此操作，例如添加 /etc/sysctl.d/disable-ipv6.conf 和内容：

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
```

此方法比在内核命令行上禁用 IPv6 模块的加载更可取。

3.4 3.4. 时间同步

Proxmox VE 集群堆栈本身在很大程度上依赖于所有节点都具有精确同步的时间这一事实。如果所有节点上的本地时间不同步，其他一些组件（如 Ceph）也无法正常工作。

可以使用”网络时间协议”（NTP）实现节点之间的时间同步。从 Proxmox VE 7 开始，chrony 用作默认的 NTP 守护程序，而 Proxmox VE 6 使用 systemd-timesyncd。两者都预先配置为使用一组公共服务器。

如果您将系统升级到 Proxmox VE 7，建议您手动安装 chrony，ntp 或 openntpd。

3.4.1 3.4.1. 使用自定义 NTP 服务器

在某些情况下，可能需要使用非默认 NTP 服务器。例如，如果您的 Proxmox VE 节点由于限制性防火墙规则而无法访问公共互联网，则需要设置本地 NTP 服务器并告诉 NTP 守护程序使用它们。

对于使用 chrony 的系统

在 `/etc/chrony/chrony.conf` 中指定 `chrony` 应使用的服务器：

```
server ntp1.example.com iburst
server ntp2.example.com iburst
server ntp3.example.com iburst
```

重新启动时间：

```
# systemctl restart chronyd
```

检查日志以确认正在使用新配置的 NTP 服务器：

```
# journalctl --since -1h -u chrony
...
Aug 26 13:00:09 node1 systemd[1]: Started chrony, an NTP client/server.
```

(续下页)

(接上页)

```
Aug 26 13:00:15 node1 chronyd[4873]: Selected source 10.0.0.1 (ntp1.example.com)
Aug 26 13:00:15 node1 chronyd[4873]: System clock TAI offset set to 37 seconds
...
```

对于使用 systemd-timesyncd 的系统

在 /etc/systemd/timesyncd.conf 中指定 systemd-timesyncd 应使用的服务器:

```
[Time]
NTP=ntp1.example.com ntp2.example.com ntp3.example.com ntp4.example.com
```

然后, 重新启动同步服务(systemctl restart systemd-timesyncd), 并通过检查日志(journalctl ---since -1h -u systemd-timesyncd) 来验证新配置的 NTP 服务器是否正在使用中:

```
...
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 node1 systemd-timesyncd[13514]: interval/delta/delay/jitter/drift 64s/
↪-0.002s/0.020s/0.000s/-31ppm
...
```

3.5 3.5. 外部监控服务器

在 Proxmox VE 中，您可以定义外部监控服务器，这些服务器将定期接收有关主机，虚拟来宾和存储的各种统计信息。

当前支持的有如下两种:

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

外部监控服务器配置保存在/etc/pve/status.cfg 中，并且可以通过 Web 界面进行编辑。

3.5.1 3.5.1. Graphite 服务器配置

默认端口设置为 2003，默认 Graphite 的路径为 proxmox。默认情况下，Proxmox VE 通过 UDP 发送数据，因此必须将 Graphite 服务器配置为接受此参数。在这里，可以根据实际情况配置，而无需采用默认的 1500 MTU 。

您也可以配置插件使用 TCP。为了不阻塞 `pvestatd` 统计收集守护进程，需要一个超时来处理网络问题。

3.5.2 3.5.2. Influxdb 配置

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。也可以在这里配置 mtu 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"
```

使用此配置，您的服务器将侦听端口 8089 上的所有 IP 地址，并将数据写入 proxmox 数据库中。

3.6 3.6. 磁盘健康检查

尽管建议使用可靠且冗余的存储，但监视本地磁盘的运行状况会很有帮助。

从 Proxmox VE 4.3 开始，集成 smartmontools 。这是用于监视和控制本地硬盘的 S.M.A.R.T. 系统的工具

您可以通过发出以下命令来获取磁盘的状态：

```
# smartctl -a /dev/sdX
```

其中 /dev/sdX 是指向其中一个本地磁盘的路径。如果输出显示：

```
SMART support is: Disabled
```

您可以使用以下命令启用它：

```
# smartctl -s on /dev/sdX
```

有关如何使用 `smartctl` 的更多信息，请参阅 `man smartctl`。

默认情况下，`smartmontools` 守护程序 `smartd` 处于活动状态并处于启用状态，并且每 30 分钟扫描一次 /dev/sdX 和 /dev/hdX 下的磁盘以查找错误和警告，并在检测到问题时向 root 发送电子邮件。

有关如何配置 `smartd` 的更多信息，请参阅 `man smartd` 和 `man smartd.conf`。

如果将硬盘与硬件 raid 控制器配合使用，最好使用 raid 监控工具。有关此内容的更多信息，请咨询 RAID 控制器的供应商。

3.7 3.7. 逻辑卷管理（LVM）

大多数人直接在本地磁盘上安装 Proxmox VE。Proxmox VE 安装 CD 提供了多个本地磁盘管理选项，并且默认使用 LVM。安装程序允许您为此类设置选择单个磁盘，并将该磁盘用作 Volume Group （VG）pve 的物理卷。以下输出来自使用 8GB 小磁盘的测试安装。

```
# 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
```

安装程序在此 VG 中分配三个 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，并包含 Proxmox VE 的系统
- swap 交换分区
- data 格式化为 lvm 精简卷（lvm-thin），，用于存储 VM 映像。因它为快照和克隆提供了有效的支持，所以 LVM-thin 更适合此场景。

对于 Proxmox VE 4.1 以前版本（包括 4.1），安装程序会创建一个名为 `data` 的标准逻辑卷，该逻辑卷挂载在 `/var/lib/vz`。

从版本 4.2 开始，逻辑卷 `data` 是一个 LVM 精简池，用于存储基于块的客户机映像，而 `/var/lib/vz` 只是根文件系统上的一个目录。

3.7.1 3.7.1. 硬件

我们强烈建议使用硬件 RAID 控制器（带电池）进行此类设置。这样不仅可以提高性能，而且还可以提供冗余，并且磁盘可热插拔更换。

LVM 本身不需要任何特殊的硬件，内存要求非常低

3.7.2 3.7.2. 创建卷组

假设我们有一个空磁盘 /dev/sdb，我们要在它上面创建一个名为”vmdata”的卷组。

注意：请注意，以下命令将清空 /dev/sdb 上的所有现有数据。

首先创建一个分区。

```
sgdisk -N 1 /dev/sdb
```

创建一个未确认的 Physical Volume （PV）和 250K 元数据大小。

```
pvcreate --metadatasize 250k -y -ff /dev/sdb1
```

在 /dev/sdb1 上创建名为”vmdata”的卷组

```
vgcreate vmdata /dev/sdb1
```

3.7.3 3.7.4. 为 /var/lib/vz 创建一个额外的 LV

下面命令可以轻松创建一个 LV。

```
lvcreate -n <Name> -L <Size[M,G,T]> <VG>
```

如在 pve 卷组下创建一个名为 vz 的 10g lv

```
 lvcreate -n vz -L 10G vmdata
```

接着在做个。

```
 mkfs.ext4 /dev/vmdata/vz
```

最后，必须安装它。

注意确保 /var/lib/vz 为空。如果是默认安装，那么不为空。

为了使其始终可访问，请在 /etc/fstab 中添加以下行。

```
echo '/dev/pve/vz /var/lib/vz ext4 defaults 0 2' >> /etc/fstab
```

3.7.4 3.7.6. 创建 LVM 精简池

在 vmdata 卷组上，创建一个名为 data，大小为 10G 的精简池。

```
lvcreate -L 10G -T -n data vmdata
```

3.7.5 3.7.5. 调整精简池的大小

可以使用以下命令调整 LV 和元数据池的大小。

注意：扩展数据池时，还必须扩展元数据池。

```
lvresize --size +<size[\M,G,T]> --poolmetadatasize +<size[\M,G]> <VG>/<LVThin_pool>
```

将 data 精简池增加 10G，同时将元数据池增加 1G

```
 lvresize --size +10G --poolmetadatasize +1G vmdata/data
```

3.8 3.8 Linux 上的 ZFS

ZFS 是由 Sun Microsystems 设计的文件系统和逻辑卷管理器的组合。从 Proxmox VE 3.4 开始，ZFS 文件系统的本机 Linux 内核端口作为可选文件系统引入，也作为根文件系统的附加选择引入。无需手动编译 ZFS 模块——已包含所有包。

使用 ZFS，可以降低硬件预算硬件，同时实现企业功能，也可以通过利用 SSD 缓存甚至全闪存来实现高性能系统。ZFS 对 CPU 和内存消耗小可以替代硬件阵列卡，同时易于管理。

使用 ZFS 优势

- 可使用 Proxmox VE GUI 和 CLI 轻松配置和管理。
- 高可靠性
- 防止数据损坏
- 文件系统级别的数据压缩
- 快照
- 写入时拷贝克隆
- 支持各种阵列级别：RAID0、RAID1、RAID10、RAIDZ-1、RAIDZ-2 和 RAIDZ-3
- 可以使用 SSD 进行缓存

- 自我修复
- 持续的完整性检查
- 专为高存储容量而设计
- 通过网络异步复制
- 开源

3.8.1 3.8.1. 硬件

ZFS 在很大程度上依赖于内存，因此至少需要 8GB 才能启动。在实践中，尽可能多地使用您的硬件/预算。为了防止数据损坏，我们建议使用安全性高的 ECC RAM。

如果您使用专用缓存和/或日志磁盘，则应使用企业级固态盘（例如英特尔固态盘 DC S3700 系列）。这可以显著提高整体安全和性能。

注意

不要在具有自己的缓存管理的硬件 RAID 控制器之上使用 ZFS。ZFS 需要直接与磁盘通信。使用 HBA 卡或者将硬盘设置成 IT 模式会更好。

如果您正在在 VM（嵌套虚拟化）中安装 Proxmox VE，请不要对该 VM 的磁盘使用 virtio，因为 ZFS 不支持这些磁盘。请改用 IDE 或 SCSI（也适用于 virtio SCSI 控制器类型）。

3.8.2 3.8.2. 以根文件系统形式安装

使用 Proxmox VE 安装程序进行安装时，可以为根文件系统选择 ZFS。您需要在安装时选择 RAID 类型：

- RAID0：也叫作条带，此类卷的容量是所有磁盘容量的总和。但是 RAID0 没有冗余，单块盘故障会导致整个卷无法使用。但性能是最好的。
- RAID1：通常被称作镜像。数据以相同的方式写入所有磁盘。此模式至少需要 2 个相同大小的磁盘。生成的容量是单个磁盘的容量。
- RAID10：RAID0 和 RAID1 的组合。至少需要 4 个磁盘
- RAIDZ-1：RAID-5 的变体，单奇偶校验。至少需要 3 个磁盘。
- RAIDZ-2：RAID-5 的变体，双奇偶校验。至少需要 4 个磁盘。
- RAIDZ-3：RAID-5 的变体，三重奇偶校验。至少需要 5 个磁盘。

安装程序会自动对磁盘进行分区，创建一个名为 rpool 的 ZFS 池，并将根文件系统安装在 ZFS 子卷 rpool/ROOT/pve-1 上。

将创建另一个名为 rpool/data 的子卷来存储 VM 映像。为了将其与 Proxmox VE 工具一起使用，安装程序在/etc/pve/storage 中创建以下配置条目.cfg：

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

安装后，您可以使用 zpool 命令查看 ZFS 池状态：

```
# 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
```

zfs 命令用于配置和管理 ZFS 文件系统。以下命令列出了安装后的所有文件系统：

```
# 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 3.8.3. ZFS RAID 级别注意事项

在选择 ZFS 池的布局时，需要考虑几个因素。ZFS 池的基本构建块是虚拟设备或 vdev。池中的所有 vdev 均等使用，数据在它们之间条带化（RAID0）。查看 zpool（8）手册页，了解有关 vdevs 的更多详细信息。

性能

每个 vdev 类型具有不同的性能行为。最常见的两个参数是 IOPS（每秒输入/输出操作数）和写入或读取数据的带宽。

在写入数据时，RAID1 有 1 块盘的速度，读取数据相当于 2 块盘的速度。

当有 4 个磁盘时。当将其设置为 2 个镜像 vdevs（RAID10）时，池的 IOPS 和带宽相当于两个单磁盘的写入。对于读取，相当于 4 个单磁盘读取。

任何冗余级别的 RAIDZ（如 RAIDZ1, RAIDZ2…）在 IOPS 方面的类似于单个磁盘。带宽的大小取决于 RAIDZ vdev 的大小和冗余级别。

对于正在运行的 VM，在大多数情况下，IOPS 是更重要的指标。

大小、空间使用情况和冗余

虽然由镜像 vdev 组成的池将具有最佳性能表现，但可用空间将是总磁盘的 50%。每个镜像至少需要一个正常运行的磁盘，池才能保持正常运行。

N 个磁盘的 RAIDZ 类型 vdev 的可用空间大致为 N-P，其中 P 是 RAIDZ 级别。RAIDZ 级别指示在不丢失数据的情况下，有多少个任意磁盘可以出现故障。例如 3 块硬盘的 RAIDZ 也就是 RAIDZ-1，最多允许 1 块硬盘故障。可用容量为 2 块盘的容量。

使用任何 RAIDZ 级别的另一个重要因素是用于 VM 磁盘的 ZVOL 数据集的行为方式。对于每个数据块，池需要奇偶校验数据，该数据至少是池的 ashift 值定义的最小块的大小。如果 ashift 为 12，则池的块大小为 4k。ZVOL 的默认块大小为 8k。因此，在 RAIDZ2 中，写入的每个 8k 块都将导致写入两个额外的 4k 奇偶校验块，即 8k + 4k + 4k = 16k。这当然是一种简化的方法，实际情况会略有不同，元数据，压缩等在本例中未被考虑在内。

在检查 ZVOL 的以下属性时，可以观察到这种情况：

- volsize
- refreservation
- used

```
zfs get volsize,refreservation,used <pool>/vm-<vmid>-disk-X
```

volsize 是呈现给 VM 的磁盘的大小，而 refreservation 保留显示池上的保留空间，其中包括奇偶校验数据所需的预期空间。如果池已精简置备，则重新预留将设置为 0。观察这种情况的另一种方法是比较 VM 中已用磁盘空间和 used 属性。请注意，如果有快照可能数据不准。

有几个选项可以减少奇偶校验的空间消耗：

- 增加 volblocksize
- 使用镜像而不是 RAIDZ
- 使用 ashit=9，这样 blocksize=512b

volblocksize 属性只能在创建 ZVOL 时设置。可以在存储配置中更改。执行此操作时，需要相应地在虚拟机内部调整容量，并且根据用例，如果只是从 ZFS 层移动到来宾层，则会出现写入放大问题。

在创建池时使用 ashift=9 可能会导致性能下降，具体取决于下面的磁盘，并且以后无法更改。

镜像 vdev（RAID1、RAID10）有利于 VM 负载。除非您的环境具有特定需求和特征，必须使用镜像 vdev，不然 RAIDZ 的特性也是可以接受的。

3.8.4 3.8.4 系统引导程序

Proxmox VE 使用 proxmox-boot-tool 来管理引导加载程序配置。有关详细信息，请参阅有关 Proxmox VE 主机引导加载程序的章节。

3.8.5 3.8.5. ZFS 管理

本节为您提供了一些常见任务的使用示例。ZFS 本身非常强大，并提供了许多选项。管理 ZFS 的主要命令是 zfs 和 zpool。这两个命令都带有出色的手册页，可以使用以下命令阅读：

```
man zpool
man zfs
```

3.8.6 3.8.6. 创建新的 zpool

若要创建新池，至少需要一个磁盘。ashift 应具有与底层磁盘相同的扇区大小（2 次方位）或更大。

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

要激活压缩（请参阅 ZFS 中的压缩部分）：

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

使用 RAID-0 创建新池最少 1 个磁盘

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

使用 RAID-1 创建新池最少 2 个磁盘

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

使用 RAID-10 创建新池最少 4 个磁盘

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

使用 RAIDZ-1 创建新池最少 3 个磁盘

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

使用 RAIDZ-2 创建新池最少 4 个磁盘

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

创建具有缓存的新池（L2ARC）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。

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

使用日志创建新池（ZIL）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。

```
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

注意需要使用 GPT 分区表。

日志盘的大小应约为物理内存大小的一半。SSD 的其余部分可以用作缓存。

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

更改故障设备

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

在使用 systemd-boot 时更换故障的系统磁盘

根据 Proxmox VE 的安装方式，它要么使用 proxmox-boot-tool [3]，要么使用普通的 grub 作为引导加载程序（参见主机引导加载程序）。您可以通过运行以下命令进行检查：

```
proxmox-boot-tool status
```

复制分区表、重新颁发 GUID 和替换 ZFS 分区的第一步是相同的。要使系统可从新磁盘引导，需要执行不同的步骤，具体取决于所使用的引导加载程序。

```
sgdisk <healthy bootable device> -R <new device>
sgdisk -G <new device>
zpool replace -f <pool> <old zfs partition> <new zfs partition>
```

注意使用 zpool status -v 命令监视新磁盘的重新同步过程的进展程度。

使用 proxmox-boot-tool:

```
proxmox-boot-tool format <new disk's ESP>
proxmox-boot-tool init <new disk's ESP>
```

注意 ESP 代表 EFI 系统分区，它是从版本 5.4 开始由 Proxmox VE 安装程序设置的可启动磁盘上的分区 #2。有关详细信息，请参阅设置新分区以用作同步的 ESP。

使用 grub:

```
grub-install <new disk>
```

3.8.7 3.8.6. 激活电子邮件通知

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。

3.8.8 3.8.6 配置 ZFS 内存使用上限

默认情况下，ZFS 会使用宿主机 50% 的内存做为 ARC 缓存。

为 ARC 分配足够的内存对于 IO 性能至关重要，因此请谨慎减少内存。根据一般情况，建议，1TB 空间使用 1G 内存，同时预留 2G 基本内存。如果池有 8TB 空间，那应该给 ARC 分配 8G+2G 共 10G 的内存。

您可以通过直接写入 zfs_arc_max 模块参数来更改当前引导的 ARC 使用限制（重新启动会失效）:

```
 echo "$[10 * 1024*1024*1024]" >/sys/module/zfs/parameters/zfs_arc_max
```

要永久生效，请将以下行添加到 /etc/modprobe.d/zfs.conf 中。如要限制为 8G 内存，则添加如下:

```
options zfs zfs_arc_max=8589934592
```

注意: 如果所需的 zfs_arc_max 值小于或等于 zfs_arc_min（默认为系统内存的 1/32），则将忽略 zfs_arc_max，除非您还将 zfs_arc_min 设置为最多 zfs_arc_max - 1。

如下，在 256G 内存的系统上，限制 ARC 为 8GB，需要设置 zfs_arc_min -1。只设置 zfs_arc_max 是不行的

```
echo "$[8 * 1024*1024*1024 - 1]" >/sys/module/zfs/parameters/zfs_arc_min
echo "$[8 * 1024*1024*1024]" >/sys/module/zfs/parameters/zfs_arc_max
```

如果根文件系统是 ZFS，则每次更改此值时都必须更新初始化接口： `update-initramfs -u`，同时重新启动才能激活这些更改。

3.8.9 3.8.7 ZFS 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`

下表示 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|

3.8.10 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
```

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。

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

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。

创建数据集/zvols 时需要设置加密，并且默认情况下会继承到子数据集。例如，要创建加密的数据集 tank/encrypted_data 并将其配置为 Proxmox VE 中的存储，请运行以下命令：

```
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
```

在此存储上创建的所有来宾卷/磁盘都将使用父数据集的共享密钥材料进行加密。

要实际使用存储，需要加载关联的密钥材料并装载数据集。这可以通过以下步骤一步完成：

```
zfs mount -l tank/encrypted_data
Enter passphrase for 'tank/encrypted_data':
```

还可以使用（随机）密钥文件，而不是通过设置密钥分配和密钥格式属性来提示输入密码，无论是在创建时还是在现有数据集上使用 zfs 更改键：

```
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
```

警告使用密钥文件时，需要特别注意保护密钥文件，防止未经授权的访问或意外丢失。没有密钥文件，则无法访问纯文本数据！

在加密数据集下创建的来宾卷将相应地设置其 encryptionroot 属性。密钥材料只需在每个加密根加载一次，即可用于其下的所有加密数据集。

有关更多详细信息和高级用法，请参阅加 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

3.8.11 3.8.10. ZFS 中的压缩

在数据集上启用压缩后，ZFS 会尝试在写入所有新块之前对其进行压缩，并在读取时解压缩它们。现有数据将不会被追溯压缩。

您可以使用以下命令启用压缩：

```
zfs set compression=<algorithm> <dataset>
```

我们建议使用 lz4 算法，因为它增加的 CPU 开销非常少。其他算法，如 lzjb 和 gzip-N，其中 N 是从 1（最快）到 9（最佳压缩比）的整数，也可用。根据算法和数据的可压缩性，启用压缩甚至可以提高 I/O 性能。

您可以随时禁用压缩：

```
zfs set compression=off <dataset>
```

同样，只有新块会受到此更改的影响。

3.8.12 3.8.11. ZFS 特殊设备

由于版本 0.8.0 ZFS 支持特殊设备。池中的特殊设备用于存储元数据、重复数据删除表和可选的小型文件块。

特殊设备可以提高由具有大量元数据更改的慢速旋转硬盘组成的池的速度。例如，涉及创建、更新或删除大量文件的工作负载将受益于特殊设备的存在。ZFS 数据集还可以配置为在特殊设备上存储整个小文件，这可以进一步提高性能。特殊设备应使用快速 SSD。

特殊设备的冗余应与池中的冗余相匹配，因为特殊设备是整个池的故障点。

注意无法撤消将特殊设备添加到池中的过程！

使用特殊设备和 RAID-1 创建池：

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

使用 RAID-1 将特殊设备添加到现有池中：

```
zpool add <pool> special mirror <device1> <device2>
```

ZFS 数据集支持 special_small_blocks=< 大小 > 属性。size 可以为 0，这样可以禁止在特殊设备上存储小文件，或者设置 512B 到 128K 之间的 2 的幂值。设置后，将在特殊设备上分配小于该大小的新文件块。

注意，如果 special_small_blocks 的值大于或等于数据集的记录大小（默认为 128K），则所有数据都将写入特殊设备，因此请小心设置。

在池上设置 special_small_blocks，所有的子数据集会继承此值。（例如，池中的所有容器都将选择加入小文件块）。

为整个池设置值：

```
zfs set special_small_blocks=4K <pool>
```

为单个数据集设置值：

```
zfs set special_small_blocks=4K <pool>/<filesystem>
```

禁止某个数据集存储小文件块：

```
zfs set special_small_blocks=0 <pool>/<filesystem>
```

3.8.13 3.8.12. ZFS 池功能

对 ZFS 中磁盘格式的更改仅在主要版本更改之间进行，并通过功能指定。所有特征以及一般机制都在 zpool-features（5）手册页中查询到。

由于启用新功能会使池无法由旧版本的 ZFS 导入，因此需要管理员通过在池上运行 zpool 升级来主动完成此操作（请参阅 zpool-upgrade（8）手册页）。

除非您需要使用其中一项新功能，否则启用它们没有任何好处。

实际上，启用新功能有一些缺点：

- 如果 rpool 上激活了新功能，则仍然使用 grub 引导的 ZFS 根目录的系统将变得无法启动，因为 grub 中 ZFS 的实现不兼容。
- 使用较旧的内核引导时，系统将无法导入任何升级的池，该内核仍随旧的 ZFS 模块一起提供。
- 引导较旧的 Proxmox VE ISO 来修复非引导系统同样不起作用。

注意：如果您的系统仍然使用 grub 引导，请不要升级 rpool，因为这会使您的系统无法引导。这包括在 Proxmox VE 5.4 之前安装的系统，以及使用旧版 BIOS 引导的系统（请参阅如何确定引导加载程序）。

为 ZFS 池启用新功能：

```
zpool upgrade <pool>
```

3.9 3.9. BTRFS

注意，BTRFS 集成目前是 Proxmox VE 中的技术预览版。

BTRFS 是一个由 Linux 内核支持的写入时复制的先进文件系统。通过数据和元数据的校验和实现快照功能，支持 RAID 和自我修复。

从 Proxmox VE 7.0 开始，引入 BTRFS 作为根文件系统。可在安装的时候选择。

一般 BTRFS 优势

- 主系统设置与传统的基于 ext4 的设置几乎相同
- 快照
- 文件系统级别的数据压缩
- 写入时拷贝克隆
- RAID0、RAID1 和 RAID10
- 防止数据损坏

- 自我修复
- Linux 内核原生支持
- …

** 警告 RAID 5/6 是实验性的和危险的

3.10 3.9.1. 作为根文件系统安装

使用 Proxmox VE 安装程序进行安装时，可以为根文件系统选择 BTRFS。

您需要在安装时选择 RAID 类型：

- RAID0：也称为”条带化”。此类卷的容量是所有磁盘容量的总和。但是 RAID0 不会添加任何冗余，因此单个驱动器的故障会使卷无法使用。
- RAID1: 也称为”镜像”。数据以相同的方式写入所有磁盘。此模式至少需要 2 个相同大小的磁盘。生成的容量是单个磁盘的容量.
- RAID10: RAID0 和 RAID1 的组合。至少需要 4 个磁盘。

安装程序会自动对磁盘进行分区，并在 `/var/lib/pve/local-btrfs` 上创建一个附加子卷。为了将其与 Proxmox VE 工具一起使用，安装程序在`/etc/pve/storage` 中创建以下配置条目.cfg：

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

btrfs: local-btrfs
        path /var/lib/pve/local-btrfs
        content iso,vztmpl,backup,images,rootdir
```

这将禁用默认的 `local` 存储，支持子卷上的 `local-btrfs` 存储条目。

btrfs 命令用于配置和管理 btrfs 文件系统，安装后，以下命令列出了所有其他子卷：

```
btrfs subvolume list /
ID 256 gen 6 top level 5 path var/lib/pve/local-btrfs
```

3.11 3.9.2. BTRFS 管理

本节为您提供了一些常见任务的使用示例。

3.11.1 创建 BTRFS 文件系统

要创建 BTRFS 文件系统，请使用 mkfs.btrfs。-d 和 -m 参数分别用于设置元数据和数据的配置文件。使用可选的 -L 参数，可以设置标签。

通常支持以下模式：单一，raid0，raid1，raid10。

在单个磁盘 /dev/sdb 上创建一个 BTRFS 文件系统，标签为 My-Storage：

```
mkfs.btrfs -m single -d single -L My-Storage /dev/sdb
```

或者在两个分区 /dev/sdb1 和 /dev/sdc1 上创建 RAID1：

```
mkfs.btrfs -m raid1 -d raid1 -L My-Storage /dev/sdb1 /dev/sdc1
```

3.11.2 挂载 BTRFS 文件系统

分区之后，可以挂载新的文件系统，如下：

```
mkdir /my-storage
mount /dev/sdb /my-storage
```

BTRFS 也可以像任何其他挂载点一样添加到 /etc/fstab 中，自动将其挂载到引导时。建议避免使用块设备路径，但使用打印的 mkfs.btrfs 命令的 UUID 值，尤其是在 BTRFS 设置中有多个磁盘。

例如/etc/fstab：

```
# using the UUID from the mkfs.btrfs output is highly recommended
UUID=e2c0c3ff-2114-4f54-b767-3a203e49f6f3 /my-storage btrfs defaults 0 0
```

如果您找不到 UUID，则可以使用 blkid 工具列出块设备的所有属性。

之后，您可以通过执行以下命令来触发第一次挂载：

```
mount /my-storage
```

下次重新启动后，系统将在启动时自动完成挂载操作。

3.11.3 将 BTRFS 文件系统添加到 Proxmox VE

您可以通过 Web 界面或使用 CLI 将现有的 BTRFS 文件系统添加到 Proxmox VE，例如：

```
pvesm add btrfs my-storage --path /my-storage
```

3.11.4 创建子卷

创建子卷会将其链接到 btrfs 文件系统中的路径，在该路径中，它将显示为常规目录。

```
btrfs subvolume create /some/path
```

之后/some/path 将像常规目录一样工作。

3.11.5 删除子卷

与通过 rmdir 删除的目录相反，子卷不需要为空即可通过 btrfs 命令删除。

```
btrfs subvolume delete /some/path
```

3.11.6 创建子卷的快照

BTRFS 实际上并不区分快照和普通子卷，因此拍摄快照也可以被视为创建子卷的任意副本。按照惯例，Proxmox VE 在创建来宾磁盘或子卷的快照时将使用只读标志，但此标志也可以在以后更改。

```
btrfs subvolume snapshot -r /some/path /a/new/path
```

这将在 `/a/new/path` 的 `/some/path` 上创建子卷的只读”克隆”。将来对 /some/path 的任何修改都会导致在修改之前复制修改后的数据。如果省略了只读（-r）选项，则两个子卷都是可写的。

3.11.7 启用压缩

默认情况下，BTRFS 不压缩数据。要启用压缩，可以添加压缩装载选项。请注意，已经写入的数据在事后不会被压缩。

默认情况下，rootfs 将在 `/etc/fstab` 中列出，如下所示：

```
UUID=<uuid of your root file system> / btrfs defaults 0 1
```

您可以简单地将 compress=zstd、compress=lzo 或 compress=zlib 附加到上面的默认值，如下所示：

```
UUID=<uuid of your root file system> / btrfs defaults,compress=zstd 0 1
```

此更改将在重新启动后生效。

3.11.8 检查空间使用情况

经典的 df 工具可能会为某些 btrfs 设置输出令人困惑的值。为了更好地估计，请使用 btrfs 文件系统使用情况 /PATH 命令，例如：

```
btrfs fi usage /my-storage
```

3.12 3.10. Proxmox 节点管理

Proxmox VE 节点管理工具（pvenode）允许您控制节点特定的设置和资源。

目前，pvenode 允许您设置节点的描述，对节点的客户机运行各种批量操作，查看节点的任务历史记录，以及管理节点的 SSL 证书，这些证书通过 pveproxy 用于 API 和 Web GUI。

3.12.1 3.10.1. 网络唤醒

LAN 唤醒（WoL）允许您通过发送幻数据包来打开网络中处于睡眠状态的计算机。必须至少有一个 NIC 支持此功能，并且需要在计算机的固件（BIOS/UEFI）配置中启用相应的选项。选项名称可能从”启用局域网唤醒”到”通过 PCIE 设备打开电源”不等。如果您不确定，请查看主板的供应商手册。ethtool 可用于检查 < 接口 > 的 WoL 配置，方法是运行：

```
ethtool <interface> | grep Wake-on
```

pvenode 允许您通过 WoL 唤醒集群中处于睡眠状态的节点，使用以下命令：

```
pvenode wakeonlan <node>
```

这将在 UDP 端口 9 上广播 WoL 幻数据包，其中包含从 wakeonlan 属性获取的<node> 的 MAC 地址。可以使用以下命令设置特定于节点的 wakeonlan 属性：

```
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX
```

3.12.2 3.10.2. 任务历史

要对服务器问题（例如，失败的备份作业）进行故障排除时，通常查看之前的任务历史。

使用 Proxmox VE，您可以通过 `pvenode task` 命令访问节点的任务历史记录。

您可以使用 `list` 表子命令获取节点已完成任务的筛选列表。例如，若要获取 VM 100 错误的相关任务的列表，命令为：

```
pvenode task list --errors --vmid 100
```

然后可以使用其 UPID 打印任务的日志：

```
pvenode task log UPID:pve1:00010D94:001CA6EA:6124E1B9:vzdump:100:root@pam:
```

3.12.3 3.10.3. 批量客户机电源管理

如果有许多 VM/容器，则可以使用 pvenode 的 `startall` 和 `stopall` 子命令在批量操作中启动和停止来宾。默认情况下，`pvenode startall` 将仅启动已设置为在启动时自动启动的 VM/容器（请参阅虚拟机的自动启动和关闭），但是，可以使用 `--force` 绕过此限制。这两个命令还具有–vms 选项，该选项将停止/启动的客户机限制为指定的 VMID。

例如，要启动 VM 100、101 和 102，无论它们是否设置了 onboot，都可以使用：

```
pvenode startall --vms 100,101,102 --force
```

要停止这些来宾（以及可能正在运行的任何其他来宾），请使用以下命令：

```
pvenode stopall
```

3.12.4 3.10.4. 第一个客户机引导延迟

如果您的虚拟机/容器依赖于启动缓慢的外部资源（例如 NFS 服务器），您还可以设置在 Proxmox VE 开机后与第一个自动启动的虚拟机或容器之间的引导延迟（请参阅虚拟机的自动启动和关闭）。

您可以通过设置以下内容（其中 10 表示以秒为单位的延迟）来实现此目的：

```
pvenode config set --startall-onboot-delay 10
```

3.12.5 3.10.5. 批量客户迁移

如果升级情况需要您将所有来宾从一个节点迁移到另一个节点，pvenode 还提供了用于批量迁移的 migrateall 子命令。默认情况下，此命令会将系统上的每个客户机迁移到目标节点。但是，可以将其设置为仅迁移一组来宾。

例如，要将 VM 100、101 和 102 迁移到节点 pve2 并启用本地磁盘的实时迁移，可以运行：

```
pvenode migrateall pve2 --vms 100,101,102 --with-local-disks
```

3.13 3.11. 证书管理

3.13.1 3.11.1. 集群内通信的证书

默认情况下，每个 Proxmox VE 集群都会创建自己的（自签名）证书颁发机构（CA），并为每个节点生成一个证书，该证书由上述 CA 签名。这些证书用于与集群的 pveproxy 服务和命令行管理程序/控制台功能（如果使用 SPICE）进行加密通信。

CA 证书和密钥存储在 Proxmox Cluster File System （pmxcfs）中。

3.13.2 3.11.2. API 和 Web GUI 的证书

REST API 和 Web GUI 由 pveproxy 服务提供，该服务在每个节点上运行。

对于 pveproxy 使用的证书，您有以下选项：

- 1. 默认情况下，使用 /etc/pve/nodes/NODENAME/pve-ssl.pem 中特定于节点的证书。此证书由群集 CA 签名，因此浏览器和操作系统不会自动信任此证书。
- 1. 使用外部提供的证书（例如，由商业 CA 签名）。
- 1. 使用 ACME（Let’s Encrypt）获得具有自动续订功能的可信证书，这也集成在 Proxmox VE API 和 Webinterface 中。

对于选项 2 和 3，使用文件 /etc/pve/local/pveproxy-ssl.pem 和 /etc/pve/local/pveproxy-ssl.key（私钥需要没有密码）。

注意请记住，/etc/pve/local 是指向 /etc/pve/nodes/NODENAME 的特定于节点的符号链接。

证书使用 Proxmox VE 节点管理命令进行管理（参见 pvenode（1）手册页）。

```
请勿替换或手动修改 /etc/pve/local/pve-ssl.pem 和 /etc/pve/local/pve-ssl.
↪key 中自动生成的节点证书文件，也不要替换或手动修改 /etc/pve/pve-root-ca.pem 和 /etc/
↪pve/priv/pve-root-ca.key 中的群集 CA 文件。
```

3.13.3 3.11.3. 上传自定义证书

如果您已经有一个要用于 Proxmox VE 节点的证书，您只需通过 Web 界面上传该证书即可。

请注意，证书密钥文件（如果提供）不得受密码保护。

3.13.4 3.11.4. 通过 Let’s Encrypt （ACME）获得的可信证书

Proxmox VE 包括 Automatic Certificate Management Environment ACME 协议的实现，允许 Proxmox VE 管理员使用像 Let’s Encrypt 这样的 ACME 提供程序，以便轻松设置 TLS 证书，这些证书在现代操作系统和 Web 浏览器上是开箱即用的接受和信任的。

目前，实现的两个 ACME 终结点是 Let’s Encrypt （LE）生产及其过渡环境。我们的 ACME 客户端支持使用内置 Web 服务器验证 http-01 挑战，并使用支持所有 DNS API 端点的 DNS 插件验证 dns-01 挑战，acme.sh 就是做的。

acme 账户

您需要向要使用的终端节点注册每个集群的 ACME 帐户。用于该帐户的电子邮件地址将用作来自 ACME 终结点的续订到期通知或类似通知的联系点。

您可以通过 Web 界面”数据中心-> ACME”或使用 pvenode 命令行工具注册和停用 ACME 帐户。

```
pvenode acme account register account-name mail@example.com
```

注意由于速率限制，您应该使用 LE 暂存进行实验，或者如果您是第一次使用 ACME。

ACME 插件

ACME 插件任务是提供自动验证，证明您以及您操作下的 Proxmox VE 集群是域的真正所有者。这是自动证书管理的基础构建基块。

ACME 协议指定不同类型的挑战，例如 http-01，其中 Web 服务器提供具有特定内容的文件以证明它控制域。有时这是不可能的，要么是因为技术限制，要么是因为无法从公共互联网访问记录的地址。在这些情况下，可以使用 dns-01 挑战。通过在域的区域中创建特定的 DNS 记录，可以应对此挑战。

Proxmox VE 开箱即用地支持这两种挑战类型，您可以通过数据中心->ACME 下的 Web 界面或使用 pvenode acme 插件 add 命令配置插件。

ACME 插件配置存储在 /etc/pve/priv/acme/plugins.cfg。插件可用于群集中的所有节点。

节点域

每个域都是特定于节点的。您可以在”节点 -> 证书”下或使用 pvenode config 命令添加新的域条目或管理现有域条目。

screenshot/gui-node-certs-add-domain.png 为节点配置所需的域并确保选择了所需的 ACME 帐户后，可以通过 Web 界面订购新证书。成功后，接口将在 10 秒后重新加载。

续订将自动进行

3.13.5 3.11.5. ACME HTTP 挑战插件

始终有一个隐式配置的独立插件，用于通过端口 80 上生成的内置 Web 服务器验证 http-01 挑战。独立名称意味着它可以自己提供验证，而无需任何第三方服务。因此，此插件也适用于群集节点。

有一些先决条件，可以使用它与 Let’s Encrypts ACME 进行证书管理。

- 您必须接受 Let’s Encrypt 的 ToS 才能注册一个帐户。
- 节点的端口 80 需要可从互联网访问。
- 端口 80 上不得有其他侦听器。
- 请求的（子）域需要解析为节点的公共 IP。

3.13.6 3.11.6. ACME DNS API 挑战插件

在无法或不希望通过 http-01 方法进行外部访问以进行验证的系统上，可以使用 dns-01 验证方法。此验证方法需要一个允许通过 API 预配 TXT 记录的 DNS 服务器。

配置用于验证的 ACME DNS API

Proxmox VE 重用了为 acme.sh[4] 项目开发的 DNS 插件，有关特定 API 配置的详细信息，请参阅其文档。

使用 DNS API 配置新插件的最简单方法是使用 Web 界面（数据中心 -> ACME）。

选择 DNS 作为质询类型。然后，您可以选择您的 API 提供商，输入凭据数据以通过其 API 访问您的帐户。

请参阅 acme.sh 如何使用 DNS API wiki，了解有关获取提供商的 API 凭据的更多详细信息。

由于有许多 DNS 提供商和 API 端点，Proxmox VE 会自动为某些提供商的凭据生成表单。对于其他人，您将看到一个更大的文本区域，只需复制其中的所有凭据 KEY=VALUE 对即可。

通过别名进行 DNS 验证

可以使用特殊的别名模式来处理不同域/DNS 服务器上的验证，以防您的主/真实 DNS 不支持通过 API 进行预配。手动为指向 _acme-challenge.domain2.example _acme-challenge.domain1.example 设置永久 CNAME 记录，并将 Proxmox VE 节点配置文件中的别名属性设置为 domain2.example，以允许 domain2.example 的 DNS 服务器验证 domain1.example 的所有质询。

插件组合

如果您的节点可以通过具有不同要求/DNS 配置功能的多个域访问，则可以将 http-01 和 dns-01 验证相结合。通过为每个域指定不同的插件实例，也可以混合使用来自多个提供商或实例的 DNS API。通过多个域访问同一服务会增加复杂性，应尽可能避免使用。

3.13.7 3.11.7. 自动续订 ACME 证书

如果已使用 ACME 提供的证书（通过 pvenode 或 GUI）成功配置了节点，则该证书将由 pve-daily-update.service 自动续订。目前，如果证书已过期，或者将在接下来的 30 天内过期，则将尝试续订。

3.13.8 3.11.8. pvenode 配置 ACME 示例

例 1：使用 Let’s Encrypt 证书的示例 pvenode 调用

```
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

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
...
Task OK
root@proxmox:~# pvenode config set --acme domains=example.invalid
root@proxmox:~# pvenode acme cert order
Loading ACME account details
Placing ACME order
...
Status is 'valid'!

All domains validated!
...
```

(续下页)

(接上页)

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

例 2：设置用于验证域的 OVH API

无论使用哪个插件，帐户注册步骤都是相同的，并且此处不再重复。根据 OVH API 文档，需要从 OVH 获取 OVH_AK 和 OVH_AS 首先，您需要获取所有信息，以便您和 Proxmox VE 可以访问 API。

```
root@proxmox:~# cat /path/to/api-token
OVH_AK=XXXXXXXXXXXXXXXX
OVH_AS=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
root@proxmox:~# source /path/to/api-token
root@proxmox:~# curl -XPOST -H"X-Ovh-Application: $OVH_AK" -H "Content-type:␣
↪application/json" \
https://eu.api.ovh.com/1.0/auth/credential  -d '{
  "accessRules": [
    {"method": "GET","path": "/auth/time"},
    {"method": "GET","path": "/domain"},
    {"method": "GET","path": "/domain/zone/*"},
    {"method": "GET","path": "/domain/zone/*/record"},
    {"method": "POST","path": "/domain/zone/*/record"},
    {"method": "POST","path": "/domain/zone/*/refresh"},
    {"method": "PUT","path": "/domain/zone/*/record/"},
    {"method": "DELETE","path": "/domain/zone/*/record/*"}
]
}'
{"consumerKey":"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ","state":"pendingValidation",
↪"validationUrl":"https://eu.api.ovh.com/auth/?
↪credentialToken=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"}

(open validation URL and follow instructions to link Application Key with account/
↪Consumer Key)

root@proxmox:~# echo "OVH_CK=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ" >> /path/to/api-token
```

现在，您可以设置 ACME 插件：

```
root@proxmox:~# pvenode acme plugin add dns example_plugin --api ovh --data /path/to/
↪api_token
root@proxmox:~# pvenode acme plugin config example_plugin
┌──────┬──────────────────────────────────────────────────────┐
```

(续下页)

(接上页)

```
| key    | value                                          |
╞════════╪════════════════════════════════════════════════╡
| api    | ovh                                            |
├────────┼────────────────────────────────────────────────┤
| data   | OVH_AK=XXXXXXXXXXXXXXXX                        |
|        | OVH_AS=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY        |
|        | OVH_CK=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ        |
├────────┼────────────────────────────────────────────────┤
| digest | 867fcf556363ca1bea866863093fcab83edf47a1       |
├────────┼────────────────────────────────────────────────┤
| plugin | example_plugin                                 |
├────────┼────────────────────────────────────────────────┤
| type   | dns                                            |
└────────┴────────────────────────────────────────────────┘
```

最后，您可以配置要为其获取证书的域并为其下达证书顺序：

```
root@proxmox:~# pvenode config set -acmedomain0 example.proxmox.com,plugin=example_
↪plugin
root@proxmox:~# pvenode acme cert order
Loading ACME account details
Placing ACME order
Order URL: https://acme-staging-v02.api.letsencrypt.org/acme/order/11111111/22222222

Getting authorization details from 'https://acme-staging-v02.api.letsencrypt.org/acme/
↪authz-v3/33333333'
The validation for example.proxmox.com is pending!
[Wed Apr 22 09:25:30 CEST 2020] Using OVH endpoint: ovh-eu
[Wed Apr 22 09:25:30 CEST 2020] Checking authentication
[Wed Apr 22 09:25:30 CEST 2020] Consumer key is ok.
[Wed Apr 22 09:25:31 CEST 2020] Adding record
[Wed Apr 22 09:25:32 CEST 2020] Added, sleep 10 seconds.
Add TXT record: _acme-challenge.example.proxmox.com
Triggering validation
Sleeping for 5 seconds
Status is 'valid'!
[Wed Apr 22 09:25:48 CEST 2020] Using OVH endpoint: ovh-eu
[Wed Apr 22 09:25:48 CEST 2020] Checking authentication
[Wed Apr 22 09:25:48 CEST 2020] Consumer key is ok.
Remove TXT record: _acme-challenge.example.proxmox.com

All domains validated!

Creating CSR
```

(续下页)

(接上页)

```
Checking order status
Order is ready, finalizing order
valid!

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

例 3 从暂存目录切换到常规 ACME 目录

不支持更改帐户的 ACME 目录，但由于 Proxmox VE 支持多个帐户，因此您只需创建一个以生产（受信任的）ACME 目录作为终结点的新帐户。您还可以停用暂存帐户并重新创建它。

例 4 使用 pvenode 将默认 ACME 帐户从暂存更改为目录

```
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

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
...
Task OK
```

3.14 3.12. 主机引导加载程序

Proxmox VE 目前使用两个引导加载程序之一，具体取决于安装程序中选择的磁盘设置。

对于随 ZFS 一起安装的 EFI 系统，将使用根文件系统 systemd-boot。所有其他部署都使用标准的 grub 引导加载程序（这通常也适用于安装在 Debian 之上的系统）

3.14.1 3.12.1. 安装程序使用的分区方案

Proxmox VE 安装程序在选择安装的所有磁盘上创建 3 个分区。

创建的分区包括：

- 一个 1 MB BIOS 启动分区（gdisk 类型 EF02）
- 一个 512 MB 的 EFI 系统分区（ESP，gdisk 类型 EF00）
- 第三个分区，跨越设置的 hdsize 参数或用于所选存储类型的剩余空间

使用 ZFS 作为根文件系统的系统使用存储在 512 MB EFI 系统分区上的内核和 initrd 映像进行引导。对于旧版 BIOS 系统，使用 grub，对于 EFI 系统，则使用系统启动。两者都已安装并配置为指向 ESP。

在 BIOS 模式（–target i386-pc）下，grub 被安装到使用 grub [5] 引导的所有系统上所有选定磁盘的 BIOS 引导分区上。

3.14.2 3.12.2. 使用 proxmox-boot-tool 同步 ESP 的内容

proxmox-boot-tool 是一个实用程序，用于保持 EFI 系统分区的内容正确配置和同步。它将某些内核版本复制到所有 ESP，并将相应的引导加载程序配置为从 vfat 格式的 ESP 引导。在 ZFS 作为根文件系统的上下文中，这意味着您可以使用根池上的所有可选功能，而不是在 grub 中的 ZFS 实现中也存在的子集，或者必须创建一个单独的小型引导池 [6]。

在具有冗余的设置中，安装程序使用 ESP 对所有磁盘进行分区。这可确保即使第一个引导设备出现故障或 BIOS 只能从特定磁盘引导，系统也能引导。

在正常操作期间，不会保持安装 ESP。这有助于防止在发生系统崩溃时文件系统损坏 vfat 格式的 ESP，并且无需在主引导设备出现故障时手动调整 /etc/fstab。

proxmox-boot-tool 处理以下任务：

- 格式化和设置新分区
- 将新的内核映像和初始化映像复制和配置到所有列出的 ESP
- 同步内核升级和其他维护任务的配置
- 管理已同步的内核版本列表

您可以通过运行以下命令来查看当前配置的 ESP 及其状态：

```
proxmox-boot-tool status
```

设置新分区以用作同步的 ESP

要将分区格式化并初始化为同步的 ESP，例如，在替换 rpool 中出现故障的 vdev 之后，或者在转换早于同步机制的现有系统时，可以使用 `pve-kernel-helpers` 中的 `proxmox-boot-tool`。

format 命令将格式化 < 分区 >，请确保传入正确的设备/分区！

例如，要将空分区 /dev/sda2 格式化为 ESP，请运行以下命令：

```
proxmox-boot-tool format /dev/sda2
```

要设置位于 /dev/sda2 上的现有未挂载 ESP 以包含在 Proxmox VE 的内核更新同步机制中，请使用以下命令：

```
proxmox-boot-tool init /dev/sda2
```

之后，`/etc/kernel/proxmox-boot-uuids` 应该包含一个新行，其中包含新添加分区的 UUID。init 命令还将自动触发所有已配置的 ESP 的刷新。

更新所有 ESP 上的配置

要复制和配置所有可引导内核，并使 /etc/kernel/proxmox-boot-uuid 中列出的所有 ESP 保持同步，您只需运行：

```
proxmox-boot-tool refresh
```

（等效于在 root 上运行具有 ext4 或 xfs 的 update-grub 系统）。

如果您要对内核命令行进行更改，或者想要同步所有内核和初始化，这是必需的。

update-initramfs 和 apt（如有必要）都将自动触发刷新。

proxmox-boot-tool 考虑的内核版本

默认情况下配置以下内核版本：

- 当前运行的内核
- 软件包更新中新安装的版本
- 两个最新已安装的内核
- 倒数第二个内核系列的最新版本（例如 5.0、5.3），如果适用
- 任何手动选择的内核

手动保持内核可引导

如果您希望将某个内核和 initrd 映像添加到可引导内核列表中，请使用 proxmox-boot-tool 内核添加。

例如，运行以下命令，将 ABI 版本为 5.0.15-1-pve 的内核添加到内核列表中，以保持安装并同步到所有 ESP：

```
proxmox-boot-tool kernel add 5.0.15-1-pve
```

`proxmox-boot-tool` 内核列表将列出当前选择用于引导的所有内核版本：

运行 proxmox-boot-tool kernel remove 以从手动选择的内核列表中删除内核，例如：

```
# proxmox-boot-tool kernel remove 5.0.15-1-pve
```

需要运行 `proxmox-boot-tool reflash`，以便在手动添加或从上面删除内核后更新所有 EFI 系统分区（ESP）。

3.14.3 3.12.3. 确定使用哪个引导加载程序

确定使用哪个引导加载程序的最简单，最可靠的方法是观察 Proxmox VE 节点的引导过程。

您将看到蓝色的 grub 盒或简单的白色 systemd-boot 上的黑色。

从正在运行的系统确定引导加载程序可能不是 100% 准确的。最安全的方法是运行以下命令：

```
efibootmgr -v
```

如果它返回一条消息，指出 EFI 变量不受支持，则在 BIOS/旧模式中使用 grub。

如果输出包含类似于以下内容的行，则在 UEFI 模式下使用 grub。

```
Boot0005* proxmox       [...] File(\EFI\proxmox\grubx64.efi)
```

如果输出包含类似于以下内容的行，则使用 systemd-boot。

```
Boot0006* Linux Boot Manager    [...] File(\EFI\systemd\systemd-bootx64.efi)
```

通过运行：

```
proxmox-boot-tool status
```

您可以了解是否配置了 `proxmox-boot-tool`，这可以很好地指示系统的引导方式。

3.14.4 3.12.4. Grub

grub 多年来一直是引导 Linux 系统的事实标准，并且有多的文档记录 [7]。

配置

对 grub 配置的更改是通过 /etc/default/grub 中的默认文件 /etc/default/grub 或配置片段完成的。要在更改配置后重新生成配置文件，请运行

```
update-grub
```

3.14.5 3.12.5. Systemd-boot

systemd-boot 是一个轻量级的 EFI 引导加载程序。它直接从安装它的 EFI 服务分区（ESP）读取内核和 initrd 映像。从 ESP 直接加载内核的主要优点是，它不需要重新实现用于访问存储的驱动程序。在 Proxmox VE 中，proxmox-boot-tool 用于使 ESP 上的配置保持同步。

配置

systemd-boot 是通过 EFI 系统分区（ESP）根目录中的文件加载程序/loader.conf 配置的。有关详细信息，请参阅 loader.conf（5）手册页。

每个引导加载程序条目都放在目录加载程序/条目/中自己的文件中

一个示例 entry.conf 如下所示（/ 引用 ESP 的根）：

```
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
```

3.14.6 3.12.6. 编辑内核命令行

您可以在以下位置修改内核命令行，具体取决于所使用的引导加载程序：

GRUB

内核命令行需要放在变量 `GRUB_CMDLINE_LINUX_DEFAULT` 文件中 `/etc/default/grub` 中。运行 `update-grub` 会将其内容附加到 `/boot/grub/grub.cfg` 中的所有 linux 条目。

Systemd-boot

内核命令行需要作为一行放在 `/etc/kernel/cmdline` 中。要应用更改，请运行 `proxmox-boot-tool` 刷新，这会将其设置为 `loader/entries/proxmox-*.conf` 中所有配置文件的选项行。

CHAPTER 4

第四章图形用户界面

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

4.1 4.1 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。

4.2 4.2 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。

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。

4.3 4.3 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

4.3.1 4.3.1 标题栏

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

4.3.2 4.3.2 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

4.3.3 4.3.3 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

4.3.4 4.3.4 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

4.4 4.4 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。

4.4.1 4.4.1 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

4.4.2 4.4.2 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。
- 订阅：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

4.4.3 4.4.3 虚拟机

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。

4.4.4 4.4.4 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

4.4.5 4.4.5 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

CHAPTER 5

第五章集群管理

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。集群没有限制节点数量。实际上，集群内的节点数量会受到主机和网络影响。现在（2021）有用户反馈了，有超过 50 个节点的生产集群（使用的是企业级硬件）。

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。

5.1 5.1 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。
- 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。
- 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。

5.2 5.2 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 5.7.3 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。

5.3 5.3 创建集群

您可以在控制台上创建集群 (通过 SSH 登录)，也可以通过使用 Proxmox VE Webinterface 的 API 创建集群 (数据中心 → 集群)。

警告

- 请为您的集群使用唯一的名称。此名称以后无法更改。集群命名遵循与节点命名相同的规则。

5.3.1 5.3.1 通过网页界面创建集群

在“数据中心”→“集群”下，单击创建集群。输入集群名称，然后从下拉列表中选择一个网络连接作为主集群网络 (Link 0)。它默认为通过节点的主机名解析的 IP。

从 Proxmox VE 6.2 开始，可以为集群添加 8 条备用链路。要添加第二条链路作为备用链路，您可以选中 Advanced 复选框，然后选择一个额外的网络接口 (Link 1，另见第 5.8 节Corosync Redundancy)。

警告

- 确保为集群通信选择的网络未用于任何高流量负载，如 (网络) 存储或实时迁移。虽然集群网络本身只产生少量数据，但它对延迟非常敏感。详见第 5.7.1 节集群网络要求。

5.3.2 5.3.2 通过命令行创建集群

通过 ssh 登录到第一个 Proxmox VE 节点并运行以下命令：

```
hp1# pvecm create CLUSTERNAME
```

创建集群后，可以用如下命令查看集群状态：

```
hp1# pvecm status
```

5.3.3 5.3.3 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。

5.4 5.4 新增集群节点

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。

5.4.1 5.4.1 通过界面新增集群节点

登录到现有集群节点上的 Web 界面。在“数据中心 → 集群”下，单击顶部的“加入信息”按钮。然后，单击“复制信息”按钮。或者，从 Information 字段手动复制字符串。

接下来，登录到要添加的节点上的 Web 界面。在“数据中心 → 集群”下，单击“加入集群”。用您之前复制的“加入信息”文本填写 Information 字段。加入集群所需的大多数设置将自动填写。出于安全原因，必须手动输入群集密码。

注意

- 要手动输入所有必需的数据，可以禁用“Assisted Join”复选框。

单击加入按钮后，集群加入过程将立即开始。节点加入集群后，其当前节点证书将被集群证书颁发机构 (CA) 签名的证书替换，这意味着当前会话将在几秒钟后停止工作。然后，您可能需要强制重新加载 Web 界面，并使用集群凭据重新登录。

您现在应该可以在“数据中心 → 集群”下看见您的节点。

5.4.2 5.4.2 通过命令行新增集群节点

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）。加入集群后可以查看集群状态：

```
# pvecm status
```

CHAPTER 6

第六章 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。

6.1 6.1. POSIX 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）

6.2 6.2. 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

6.3 6.3 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。

6.4 6.4 文件系统布局

文件系统挂载点为：/etc/pve

6.4.1 6.4.1 文件

6.4.2 6.4.2 符号链接

注意，openvz 即将移除

6.4.3 6.4.3 用于调试的特殊状态文件（JSON）

6.4.4 6.4.4 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
```

6.5 6.5 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）。

6.5.1 6.5.1 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。

6.5.2 6.5.2 从故障节点恢复/迁移虚拟机

对于 nodes//qemu-server（虚拟机）和 nodes//lxc（容器）中的虚拟机配置文件，Proxmox VE 认为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。

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。

CHAPTER 7

第七章 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个或者多个本地存储上，也可以保存在多种共享存储上，例如 NFS 或 iSCSI（NAS，SAN）。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。

7.1 7.1. 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 2. 可用的存储类型

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

7.1.1 7.1.1. 精简置备

一些存储方案，以及 Qemu 镜像文件 qcow2，支持精简置备（thin provisioning）。在精简置备模式下，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，精简置备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。

具有”快照”功能的所有存储类型也支持精简置备。

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

7.2 7.2 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

7.2.1 7.2.1 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
```

7.2.2 7.2.2 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（rawlqcow2lvmdk）。

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。

7.3 7.3 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>
```

7.3.1 7.3.1 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。

7.4 7.4 命令行使用方法

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。

7.4.1 7.4.1 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 (该操作将删除存储卷上的所有数据。)

```
pvesm free <VOLUME_ID>
```

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>
```

将卷 `local:103/vm-103-disk-0.qcow2` 导出到文件 target。这主要在内部与 pvesm 导入一起使用。流格式 qcow2+size 与 qcow2 格式不同。

因此，导出的文件不能简单地附加到 VM。这也适用于其他格式。

```
pvesm export local:103/vm-103-disk-0.qcow2 qcow2+size target --with-snapshots 1
```

7.5 7.5 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 7.2 目录后端存储子目录

7.5.1 7.5.1 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/

7.5.2 7.5.2 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）。

7.5.3 7.5.3 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

7.5.4 7.5.4. 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
```

7.6 7.6. 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。

7.6.1 7.6.1 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,vztmp
```

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。

7.6.2 7.6.2 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 7.4 NFS 后端存储功能

7.6.3 7.6.3 示例

可用如下命令列出 NFS 共享路径：

```
pvesm nfsscan <server>
```

7.7 7.7 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。

7.7.1 7.7.1 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 。

- 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
```

7.7.2 7.7.2 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 7.5 cifs 后端存储功能

7.7.3 7.7.3 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]
```

7.8 7.8 Proxmox 备份服务器

存储池类型：pbs

此后端允许将 Proxmox 备份服务器直接集成到 Proxmox VE 中，就像任何其他存储类型一样。pbs 存储可以直接通过 Proxmox VE api，cli 或者 web 界面上直接添加。

7.8.1 7.8.1 配置

后端存储支持全部的公共存储服务属性，但 shared 标识例外。此外具有以下特殊属性：

- server

服务器的 IP 或者 DNS 名称，必填。

- usernmae

Proxmox 备份服务器的用户名，必填。

提示：不要忘记将领域添加到用户名中。例如，root@pam 或 archiver@pbs。

- password

用户的密码，该值将保存在 /etc/pve/priv/storage/<STORAGE-ID>.pw 下的文件中，并且访问权限仅限于 root 用户。必填。

- datastore

要使用的 Proxmox 备份服务器数据存储的 ID。必填。

- fingerprint

Proxmox Backup Server API TLS 证书的指纹。您可以在服务器仪表板中或使用 proxmox-backup-manager cert info 命令获取它。对于自签名证书或不被主机信任的 ca 证书，都需要。

- encryption-key

用于从客户端加密备份数据的密钥。目前仅支持非密码保护（无密钥派生函数（kdf））。将保存在 /etc/pve/priv/storage/<STORAGE-ID>.enc 下的文件中，访问权限仅限于 root 用户。使用 proxmox-backup-client key create --kdf none <path> 自动生成一个新值。可选。

- master-pubkey

用于在备份任务中加密备份加密密钥的公用 RSA 密钥。加密的副本将附加到备份中，并存储在 Proxmox 备份服务器实例上以进行恢复。可选，需要加密密钥。

配置示例（/etc/pve/storage.cfg)

```
pbs: backup
        datastore main
        server enya.proxmox.com
        content backup
        fingerprint 09:54:ef:..snip..:88:af:47:fe:4c:3b:cf:8b:26:88:0b:4e:3c:b2
        maxfiles 0
        username archiver@pbs
```

7.8.2 7.8.2. 存储功能

7.8.3 7.8.3. 加密

（可选）您可以在 GCM 模式下使用 AES-256 配置客户端加密。加密可以通过 Web 界面进行配置，也可以使用加密密钥选项在 CLI 上进行配置（见上文）。密钥将保存在文件 /etc/pve/priv/storage/.enc 中，该文件只能由 root 用户访问。

注意：

如果没有其密钥，将无法访问备份。因此，您应该将密钥保持有序，并且与要备份的内容分开。例如，您可能会使用整个系统上的密钥备份整个系统。如果系统由于任何原因而无法访问并且需要恢复，则这是不可能的，因为加密密钥将与损坏的系统一起丢失。

建议您确保密钥安全，但易于访问，以便快速灾难恢复。因此，将其存储在密码管理器中的最佳位置，可以立即恢复。作为对此的备份，您还应该将密钥保存到 USB 驱动器并将其存储在安全的地方。这样，它可以与任何系统分离，使在紧急情况下能够很容易恢复。最后，为了应对最坏的情况，您还应该考虑将钥匙的纸质副本锁在安全的地方。`paperkey` 子命令可用于创建密钥的 QR 编码版本。以下命令将 `paperkey` 命令的输出发送到文本文件，以便于打印。

```
proxmox-backup-client key paperkey /etc/pve/priv/storage/<STORAGE-ID>.enc --output-
↪format text > qrkey.txt
```

此外，还可以使用单个 RSA 主密钥对进行密钥恢复：将执行加密备份的所有客户端配置为使用单个公钥，并且所有后续加密备份将包含已用 AES 加密密钥的 RSA 加密副本。相应的私有主密钥允许恢复 AES 密钥并解密备份，即使客户端系统不再可用。

注意：与常规加密密钥一样，主密钥对的安全保存规则也适用。没有私钥的副本，恢复是不可能的！paperkey 命令支持生成私有主密钥的纸质副本，以便存储在安全的物理位置。

由于加密是在客户端管理的，因此您可以在服务器上使用相同的数据存储进行未加密的备份和加密的备份，即使它们是使用不同的密钥加密的。但是，在具有不同密钥的备份之间进行重复数据删除是不可能的，因此通常最好创建单独的数据存储。

提示：如果加密没有好处，请不要使用加密，例如，当您在受信任的网络中本地运行服务器时。从未加密的备份中恢复总是更容易。

7.8.4 7.8.4. 示例：通过 CLI 添加存储

然后，您可以使用以下命令将此共享作为存储添加到整个 Proxmox VE 集群中：

```
pvesm add pbs <id> --server <server> --datastore <datastore> --username <username> --
↪fingerprint 00:B4:... --password
```

7.9 7.9 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。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。

7.9.1 7.9.1 配置

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
```

7.9.2 7.9.2 文件命名规则

目录布局和文件命名规则与 dir 后端相同。

7.9.3 7.9.3 存储功能

glusterf 提供文件级共享，无法在 PVE 上实现快照和链接克隆。但可利用 qcow2 文件格式的支持进行虚拟机快照和链接克隆。

7.10 7.10 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。

7.10.1 7.10.1 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
```

7.10.2 7.10.2 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-- // 普通虚拟机镜像 base-- // 虚拟机模板（只读）subvol-- // 容器镜像（使用 ZFS 文件系统存储容器）

-

镜像文件所属的虚拟机 ID.

-

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。

7.10.3 7.10.3 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。

7.10.4 7.10.4 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
```

7.11 7.11 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。

7.11.1 7.11.1 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
```

7.11.2 7.11.2 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-- //普通虚拟机镜像

7.11.3 7.11.3 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 10. 后端 lvm 的存储功能

7.11.4 7.11.4. 例子

列出可用的卷组：

```
pvesm lvmscan
```

7.12 7.12 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
```

7.12.1 7.12.1 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
```

7.12.2 7.12.2 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-- //普通虚拟机镜像

7.12.3 7.12.3 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 8.10 LVM-thin 后端存储功能

7.13 7.13 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。

7.13.1 7.13.1 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。

7.13.2 7.13.2 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。

7.13.3 7.13.3 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，最佳实践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 12. 后端 iscsi 的存储功能

7.13.4 7.13.4 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]>
```

7.14 7.14 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。

7.14.1 7.14.1 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
```

7.14.2 7.14.2 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 13. 后端 iscsidirect 的存储功能

7.15 7.15 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

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。

7.15.1 7.15.1 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。

7.15.2 7.15.2 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。

7.15.3 7.15.3 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 14. 后端 rbd 的存储功能

7.16 7.16 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。

7.16.1 7.16.1 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。

7.16.2 7.16.2 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。

7.16.3 7.16.3 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。

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

[1] 虽然不存在已知的错误，但快照还不能保证是稳定的，因为它们缺乏足够的测试。

7.17 7.17. 基于 BTRFS 后端

存储池类型：btrfs

从表面上看，这种存储类型与目录类型非常相似，因此请参阅目录后端部分以获得一般介绍。

两者主要区别在于这种存储类型的 raw 磁盘将被放置在一个子卷中，以便允许快照并支持离线存储迁移并保留快照。

注意：BTRFS 在打开文件时将遵循 O_DIRECT 标志，这意味着 VM 不应使用缓存（建议为 none），否则会出现校验和错误。

7.17.1 7.17.1 配置

此后端的配置类似于目录存储。请注意，当添加一个目录作为 BTRFS 存储时，它本身并不是挂载点，强烈建议通过 is_mountpoint 选项指定实际的挂载点。

例如，如果一个 BTRFS 文件系统挂载在/mnt/data2，并且它的子目录 `pve-storage`（可能是快照）应该添加为一个名为 data2 的存储池，如下：

```
btrfs: data2
        path /mnt/data2/pve-storage
        content rootdir,images
        is_mountpoint /mnt/data2
```

7.17.2 7.17.2. 快照

拍摄子卷或 raw 文件的快照时，快照将创建为具有相同路径的只读子卷，后跟 @ 和快照的名称。

7.17.3 7.17.3. 存储功能

7.17.4 表 15. Btrfs 的存储功能（官方未正式说明）

7.18 7.18. 基于 ISCSI 后端的 ZFS

存储池类型：zfs

此后端通过 ssh 访问具有 ZFS 池作为存储和 iSCSI 目标实现的远程机器。它为每个来宾磁盘创建一个 ZVOL，并将其导出为 iSCSI LUN。Proxmox VE 将这个 LUN 用于来宾磁盘。

支持以下 ISCSI 目标提供者：

- LIO (Linux)
- LET (Linux)

- ISTGT (FreeBSD)
- Comstar (Solaris)

此插件需要支持 ZFS 的远程存储设备，您不能使用它在常规存储设备/SAN 上创建 ZFS 池

7.18.1 7.18.1. 配置

为了使用 ZFS over iSCSI 插件，您需要将远程机器（目标）配置为接受来自 Proxmox VE 节点的 ssh 连接。Proxmox VE 连接到目标以创建 ZVOL 并通过 iSCSI 导出它们。

身份验证通过存储在/etc/pve/priv/zfs/<target_ip>_id_rsa 中的 ssh 密钥（无密码保护）.

以下步骤创建一个 ssh-key 并将其传递到 IP 为 192.0.2.1 的存储机器：

```
mkdir /etc/pve/priv/zfs
ssh-keygen -f /etc/pve/priv/zfs/192.0.2.1_id_rsa
ssh-copy-id -i /etc/pve/priv/zfs/192.0.2.1_id_rsa.pub root @192.0.2.1
ssh -i /etc/pve/priv/zfs/192.0.2.1_id_rsa root@192.0.2.1
```

后端支持常见的存储属性 content、nodes、disable 和以下 ZFS over ISCSI 特定属性：

- 资源池

在 ISCSI 目标上的 ZFS 下的 pool 或 pool 下的文件系统，所有分配都在该池中完成。

- 门户

iSCSI 门户（带有可选端口的 IP 或 DNS 名称）。

- 目标

iSCSI 目标

- iSCSI 提供者

远程机器上使用的 iSCSI 提供程序

- 目标群组

使用 comstar 时的目标群组

- 主机群组

使用 comstar 时的主机群组

- 目标门户组

Linux LIO 目标的目标门户组

- 写缓存

在目标上禁用或者开启写缓存

- 块大小

设置 ZFS 块大小

- 精简配置

使用 ZFS 精简配置。实际大小不等于卷大小的卷。

7.18.2 配置示例 (/etc/pve/storage.cfg)

```
zfs: lio
   blocksize 4k
   iscsiprovider LIO
   pool tank
   portal 192.0.2.111
   target iqn.2003-01.org.linux-iscsi.lio.x8664:sn.xxxxxxxxxxxx
   content images
   lio_tpg tpg1
   sparse 1

zfs: solaris
   blocksize 4k
   target iqn.2010-08.org.illumos:02:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx:tank1
   pool tank
   iscsiprovider comstar
   portal 192.0.2.112
   content images

zfs: freebsd
   blocksize 4k
   target iqn.2007-09.jp.ne.peach.istgt:tank1
   pool tank
   iscsiprovider istgt
   portal 192.0.2.113
   content images

zfs: iet
   blocksize 4k
   target iqn.2001-04.com.example:tank1
   pool tank
   iscsiprovider iet
   portal 192.0.2.114
   content images
```

7.18.3 7.18.2. 存储功能

ZFS over iSCSI 插件提供了一个可以创建快照的共享存储。您需要确保 ZFS 设备不会成为部署中的单点故障。

表 17. 后端 iSCSI 的存储功能

CHAPTER 8

第八章部署超融合 Ceph 集群

Proxmox VE 统一了您的计算和存储系统，也就是说，您可以将集群中的相同物理节点用于计算（处理虚拟机和容器）和复制存储。传统的计算和存储资源可以集成到单个超融合设备中。独立的存储网络 (SAN) 和通过网络附加存储 (NAS) 的连接消失了。通过集成开源软件定义存储平台 Ceph，Proxmox VE 能够直接在管理程序节点上运行和管理 Ceph 存储。

Ceph 是一个分布式对象存储和文件系统，旨在提供出色的性能、可靠性和可扩展性。

Proxmox VE 上 Ceph 的一些优点是：

- 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

对于中小型部署可以直接在 Proxmox VE 集群节点上安装用于 RADOS 块设备 (RBD) 的 Ceph 服务器（请参阅 (Ceph RADOS 块设备 (RBD))[https://pve.proxmox.com/pve-docs/pve-admin-guide.html#ceph_rados_block_

devices])。最近的硬件有很多 CPU 能力和 RAM，因此在同一个节点上运行存储服务和 VM 是可能的。

为了简化管理，我们提供了 pveceph——一个用于在 Proxmox VE 节点上安装和管理 Ceph 服务的工具。

Ceph 由多个守护进程组成，用作 RBD 存储：

- Ceph 监视器 (ceph-mon)
- Ceph 管理器 (ceph-mgr)
- Ceph OSD（ceph-osd；对象存储守护进程）

8.1 8.1. 前提

要部署超融合 Proxmox + Ceph 集群，你必须使用至少 3 个相同的服务器进行设置。可以查看Ceph 网站的建议

8.1.1 CPU

高 CPU 频率可以减少延迟，应该是首选。根据经验，你应该为每个 Ceph 服务分配（预留）一个 CPU 内核（或线程），以便为稳定和持久的 Ceph 性能提供足够的资源。

8.1.2 内存

在超融合设置中，尤其需要仔细监控内存消耗。除了预测的虚拟机和容器的内存使用量之外，您还必须考虑有足够的内存可供 Ceph 使用，以提供出色而稳定的性能。

根据经验，对于大约 1 TiB 的数据，OSD 将使用 1 GiB 的内存。特别是在恢复、重新平衡或回填期间。

守护进程本身将使用额外的内存。默认情况下，守护程序的 Bluestore 后端需要 3-5 GiB 的内存（可调整）。相比之下，传统的 Filestore 后端使用 OS 页面缓存，内存消耗通常与 OSD 守护进程的 PG 有关。

8.1.3 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 [Ceph 网状网络配置参见 https://pve.proxmox.com/wiki/Full_Mesh_Network_for_Ceph_Server]。

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。

8.1.4 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。

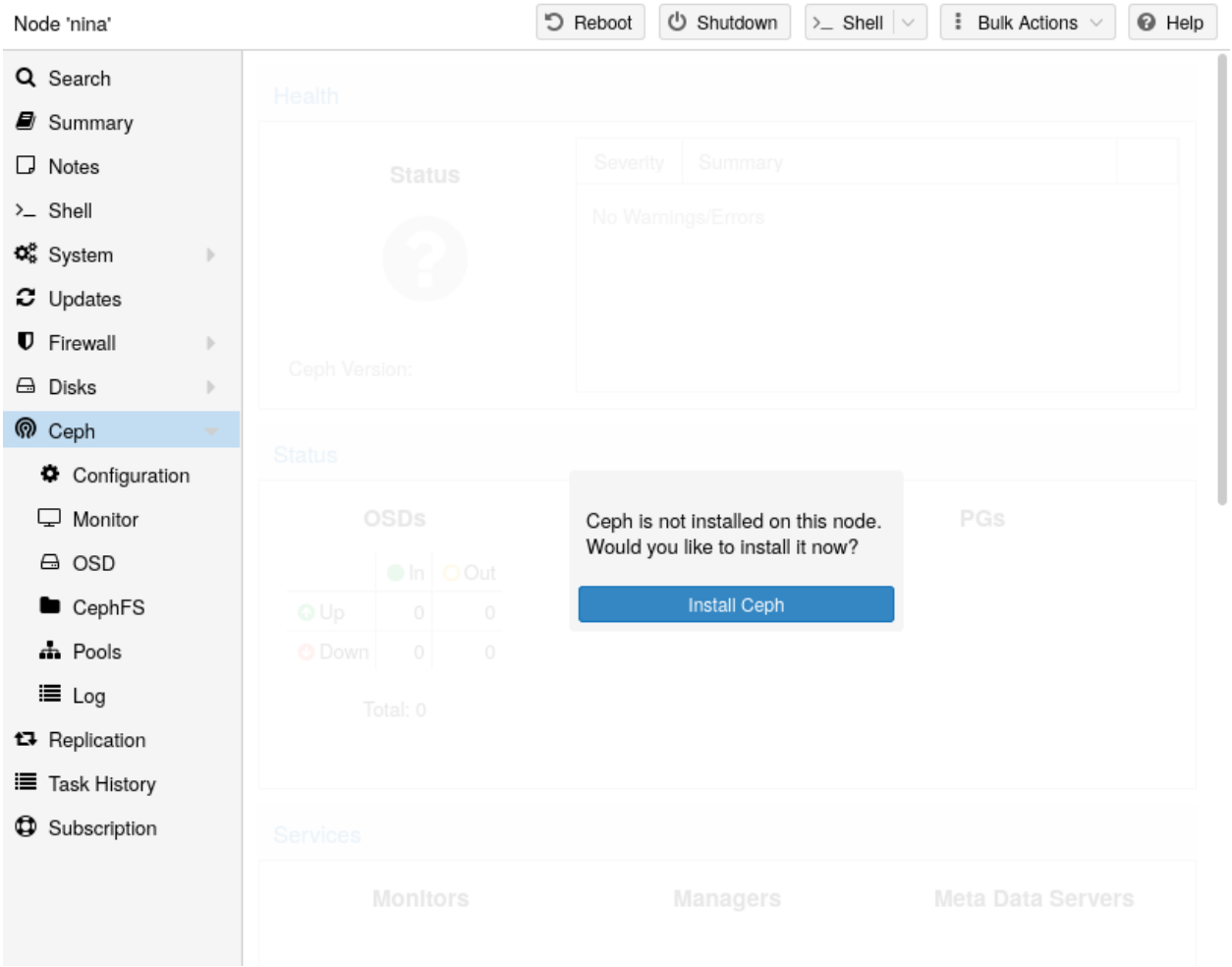
警告不要使用硬件 RAID 控制器，可改用主机 HBA 卡。

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。

8.2 8.2 初始化 Ceph 安装和配置

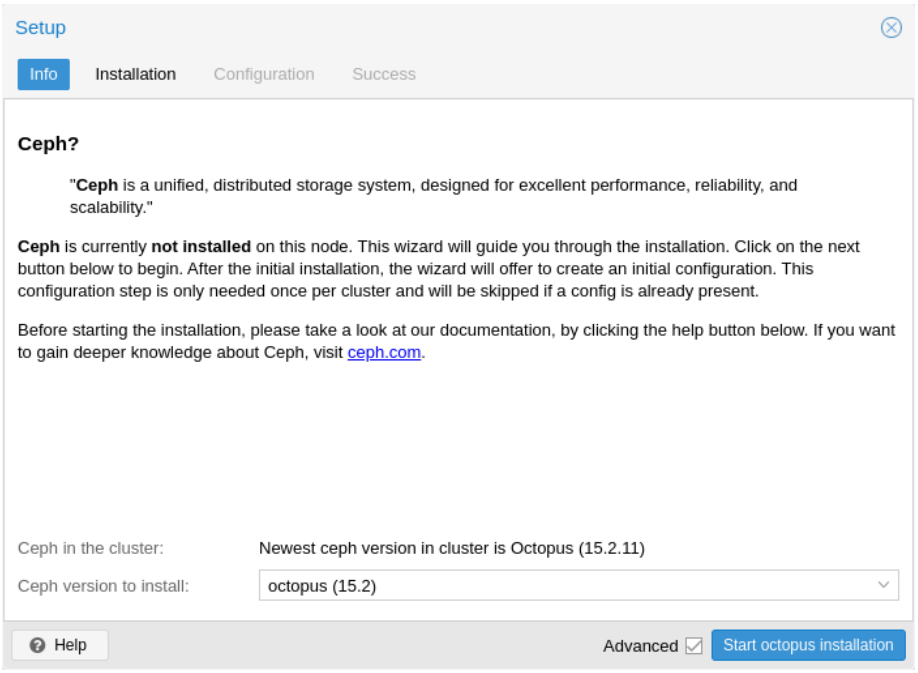
8.2.1 8.2.1. 使用基于 Web 的向导

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，您将看到一个提示您这样做的提示。



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。

首先，您需要选择要安装的 Ceph 版本。首选其他节点中的一个。如果这是您安装 Ceph 的第一个节点，则选择最新的。



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。

完成安装步骤后，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（必须）。

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。

此时系统现在已准备好开始使用 Ceph。首先需要创建一些额外的监视器、OSD 和一个池。

本章的其余部分将指导您充分利用基于 Proxmox VE 的 Ceph 设置。这包括前面提到的技巧和更多内容，例如 CephFS，它对您的新 Ceph 集群很有帮助。

8.2.2 通过 CLI 安装 Ceph

除了在 Web 界面上使用推荐的 Proxmox VE Ceph 安装向导之外，您还可以在每个节点上使用以下 CLI 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。

8.2.3 8.2.3. 通过 CLI 进行初始 Ceph 配置

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 6 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。

8.3 8.3. Ceph 监视器

Ceph Monitor（MON）[Ceph Monitor 详见 http://docs.ceph.com/docs/luminous/start/intro/] 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。

8.3.1 8.3.1 创建监视器

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 mon create
```

8.3.2 销毁监视器

要通过 GUI 删除 Ceph 监视器，首先在树视图中选择一个节点，然后转到 Ceph → 监视器面板。选择 MON 并单击 Destroy 按钮。

要通过 CLI 删除 Ceph Monitor，首先连接到运行 MON 的节点。然后执行以下命令：

```
pveceph mon destroy
```

提示: quroum 至少需要 3 个监视器

8.4 8.5 Ceph OSD

Ceph 对象存储守护进程 (Ceph Object Storage Daemons) 通过网络为 Ceph 存储对象。建议每个物理磁盘使用一个 OSD。

8.4.1 8.5.1 创建 OSD

您可以通过 Proxmox VE Web 界面或通过 CLI 使用 pveceph 创建 OSD 。例如：

```
pveceph osd create /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 Ceph Kraken 版本开始，引入了一种新的 Ceph OSD 存储类型，称为 Bluestore [16]。这是自 Ceph Luminous 以来创建 OSD 时的默认设置。

```
pveceph osd create /dev/sd[X]
```

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)

提示：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，Filestore 被用作 Ceph OSD 的默认存储类型。从 Ceph Nautilus 开始，Proxmox VE 不再支持使用 pveceph 创建此类 OSD 。如果您仍想创建文件存储 OSD，请直接使用 ceph-volume。

```
ceph-volume lvm create --filestore --data /dev/sd[X] --journal /dev/sd[Y]
```

8.4.2 8.5.2. 销毁 OSD

通过界面删除 OSD，首先在树视图中选择一个 Proxmox VE 节点，然后转到 Ceph→OSD 面板。选择要销毁的 OSD。接下来，单击 Out 按钮。一旦 OSD 状态从输入更改为输出，请点击停止按钮。状态从 up 更改为 down 后，立即从 More(更多) 下拉菜单中选择 Destroy(销毁)。

通过 CLI 删除 OSD，请运行以下命令。

```
ceph osd out <ID>
systemctl stop ceph-osd@<ID>.service
```

8.5 8.6 Ceph Pool

存储池 pool 是一组存储对象的逻辑集合。具体是由一组 Placement Groups（PG，pg_num）的对象集合组成。

8.5.1 8.6.1 创建 Ceph Pool

https://10.13.14.4:8006/pve-docs/images/screenshot/gui-ceph-pools.png

可以在 GUI 主机管理界面选择 Ceph→Pools 或直接用命令行创建 pool。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 **

- 不要将 min_size 设置为 1。min_size 为 1 的复制池允许在对象只有 1 个副本时进行 I/O，这可能导致数据丢失、不完整的 PG 或未找到的对象。

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 [PG 计算机详见 http://ceph.com/pgcalc/]。不过从 Ceph Nautilus 开始，允许在部署之后更改 PG 的数量。

PG-Autoscaler 可以在后台自动缩放池的 PG 数量，设置 Target Size 或 Target Ratio 高级参数有助于 PG-Autoscaler 自动设置最优值。

通过命令行创建 Pool 示例

```
pveceph pool create <name> --add_storages
```

如果想自动将 ceph pool 添加到 Proxmox VE 存储，请保持`--add_storages` 参数。或者在 GUI 上勾选添加存储。

Pool 选项

下面选项在创建 Pool 时可用，有些也可以在编辑池时可用。

- 名称

  池的名称，这是唯一的，且之后无法更改

- 大小

  每个对象的副本数，Ceph 根据这个值调整副本的数量。默认值为 3

- PG Autoscale Mode

  PG 自动缩放模式。如果设置为 warn，它会在池具有非最佳 PG 计数时产生警告消息。默认值：警告。

- 添加存储

  自动将 Ceph Pool 添加为 Proxmox VE 存储

高级选项

- 最小尺寸

  每个对象的最小副本数。如果 PG 的副本数少于此数量，Ceph 将拒绝池上的 I/O 请求。默认为 2

- Crush Rules

  用于在集群中映射对象放置的规则。这些规则定义了数据在集群中的放置方式。有关基于设备的规则的信息，请参阅 Ceph CRUSH 和设备类。

- of PGs

  池在创建时的 pg 数量

- Target Ratio

  池中预期的数据比率。PG 自动缩放器使用相对于其他比率集的比率。如果两者都设置，它优先于`Target Size`

- Target Size

  池中预期的估计数据量。PG 自动缩放器使用此大小来估计最佳 PG 计数。

- Min. # of PGs

  PG 的最小数量，此设置用于微调该池的 PG 计数的下限。PG Autoscale 不会合并低于此阈值的 PG。

8.5.2 8.6.2. 销毁池

要通过 GUI 销毁池，请在树视图中选择一个节点，然后转到 Ceph → Pools 面板。选择要销毁的池，然后单击 Destroy 按钮。要确认池的销毁，您需要输入池名称。

运行以下命令以销毁池。指定-remove_storages 以同时删除关联的存储。

```
pveceph pool destroy <name>
```

- 注意
  - 删除 pool 的数据是一项后台任务，可能需要一些时间。您将注意到集群中的数据使用率正在减少。

8.5.3 8.6.3. PG Autoscale

PG Autoscale 允许集群预估存储在每个池中的数据量，并自动选择适当的 pg_num 值。它从 Ceph Nautilus 开始可用。

需要激活 PG Autoscale Mode 才能使调整生效。

```
ceph mgr module enable pg_autoscaler
```

自动缩放模式基于每个池进行配置，并具有以下选项

- warn
  - 如果建议的 pg_num 值与当前值相差太大，则会发出健康警告
- on
  - pg_num 会自动调整，无需干预
- off
  - 不会自动调整 pg_num，即使当前的 pg_num 值不是最优值，也不会发出警告。

可以使用 `target_size`、`target_size_ratio` 和 `pg_num_min` 选项调整缩放因子以优化未来的数据存储。

- 注意
  - 默认情况下，如果池的 PG 数量偏离 3 倍，自动所放弃会考虑调整它。这将导致数据放置发生相当大的变化，并可能在集群上引入高负载。

您可以在 Ceph-blog New in Nautilus: PG merging and autotuning中找到对 PG 自动缩放器的更深入介绍。

8.6 8.7 Ceph CRUSH 和设备类别

Ceph 是以算法 Controlled Replication Under Scalable Hashing（CRUSH[CRUSH 详见 https://ceph.com/wp-content/uploads/2016/08/weil-crush-sc06.pdf]）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 map 的进一步信息，可以查看 Ceph 官方文档中 CRUSH map[CRUSH map 参见 http://docs.ceph.com/docs/luminous/rados/operations/crush-map/] 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。

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）

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。

8.7 8.8 Ceph 客户端

完成上面的步骤后，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7.15 节”）

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
```

8.8 8.9. 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。

8.8.1 8.9.1. 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

8.8.2 8.9.2 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：

- 安装 Ceph 软件包
- 配置好 Ceph 监视器
- 配置好 OSD
- 至少设置一个元数据服务器

完成以上操作后，就可以通过 Web GUI 的节点 →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 pg，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8.6 Ceph Pool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。

8.8.3 8.9.3. 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。

要完全且正确的删除 CephFS，需要执行以下步骤：

- 断开所有非 Proxmox VE 客户端的连接，（如虚拟机中的 Cephfs）
- 禁用所有相关的 CephFS Proxmox VE 存储条目（以防止它被自动挂载）。
- 从您要销毁的 CephFS 上的来宾（例如 ISO）中删除所有已使用的资源。
- 手动卸载所有集群节点上的 CephFS 存储
  - `umount /mnt/pve/<storage-name>` storage-name 是 Proxmox VE 中 CephFS 存储的名称。
- 现在确保没有元数据服务器 (MDS) 正在为该 CephFS 运行，方法是停止或销毁它们。这可以通过 Web 界面或命令行界面完成，对于后者，您将发出以下命令：
  - `pveceph stop --service mds.NAME` 然后 `pveceph mds destroy NAME` 请注意，当一个活动的 MDS 停止或删除时，备用服务器将自动提升为活动的，因此最好先停止所有备用服务器。
- 现在您可以使用以下命令销毁 CephFS
- `pveceph fs destroy NAME --remove-storages --remove-pools` 这将彻底删除 Cephfs 池，并从 Proxmox VE 中删除存储

在这些步骤之后，应该完全删除 CephFS，如果您有其他 CephFS 实例，可以再次启动停止的元数据服务器以充当备用服务器。

8.9 8.10. Ceph 维护

8.9.1 8.10.1 更换 OSD

Ceph 中的常见维护任务之一是更换 OSD 的磁盘。如果磁盘已经处于故障状态，则可以继续执行销毁 OSD中的步骤。如果可能，Ceph 将在剩余的 OSD 上重新创建这些副本。一旦检测到 OSD 故障或主动停止了 OSD，重新平衡将立即开始。

注意

- 使用 pool 的默认容量/最小容量 (3/2)，恢复仅在‘size+1’节点可用时开始。这样做的原因是 Ceph 对象平衡器Ceph crush缺省为一个完整的节点作为“故障域”。

要更换仍然正常工作的磁盘，请在 GUI 上执行销毁 OSD中的步骤。唯一需要添加的是等待群集显示 HANTON_OK，然后再停止 OSD 将其销毁。

在命令行上使用以下命令。

```
ceph osd out osd.<id>
```

您可以使用下面的命令检查是否可以安全地移除 OSD。

```
ceph osd safe-to-destroy osd.<id>
```

一旦上面的检查告诉您已保存以删除 OSD，您就可以继续执行以下命令。

```
systemctl stop ceph-osd@<id>.service
pveceph osd destroy <id>
```

用新磁盘替换旧磁盘，并使用创建 OSD中描述的相同过程。

8.9.2 8.10.2 Trim/Discard

在虚拟机或容器上定期运行 fstrim(丢弃) 是一种很好的措施。这会释放文件系统不再使用的数据块。它减少了数据使用和资源负载。大多数操作系统定期向其磁盘发出这样的丢弃命令。您只需确保虚拟机启用磁盘丢弃选项第 10.2.4 节。

8.9.3 8.10.3 Scrub & Deep Scrub

Ceph 通过清理放置组来确保数据完整性。Ceph 会检查 PG 中的每个对象的健康状况。有两种形式的清理，每日简单元数据检查和每周深度数据检查。每周深度清理读取对象并使用校验和来确保数据完整性。如果正在运行的擦除干扰业务 (性能) 需求，您可以调整执行擦除的时间。

8.10 8.11. 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。

CHAPTER 9

第九章存储复制

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。

注意：允许与存储复制结合使用高可用性，但在上次同步时间和节点失败时间之间可能会有一些数据丢失。

9.1 9.1 支持的存储类型

9.2 9.2 调度格式

复制使用日历事件 (需要引用 24.1) 来配置计划。

9.3 9.3 错误处理

如果复制作业遇到问题，则会将其置于错误状态。在这种状态下，配置的复制间隔会被暂时挂起。失败的复制会以 30 分钟的间隔再次尝试。一旦成功，原始计划将再次激活。

9.3.1 9.3.1. 可能的问题

下面列出了一些最常见的问题。根据您的设置，可能还有其他原因。

- 网络不工作。
- 复制目标存储上没有可用空间。
- 目标节点上可用的具有相同存储 ID 的存储

提示：可以使用复制日志找出导致问题的原因。

9.3.2 9.3.2. 发生错误时迁移来宾

在出现严重错误的情况下，虚拟客户可能会卡在故障节点上。然后，您需要再次手动将其移动到工作节点。

9.3.3 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 9.4 调度任务

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。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。

复制作业由集群范围的唯一 ID 标识。此 ID 由 VMID 和作业编号组成。如果使用 CLI 工具，则必须手动指定此 ID。

9.5 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'
```

CHAPTER 10

第十章 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 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 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 10.2.1. 常规设置

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 10.2.2 操作系统设置

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 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。如果计划安装 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 10.2.4 硬盘

磁盘控制器

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, 通常简称为 VirtIO 或 virtio-blk，是一种较旧的半虚拟化控制器。就功能而言，它已被 VirtIO SCSI 控制器所取代。

磁盘格式

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。

缓存模式

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。

Trim/丢弃

如果存储服务支持薄模式存储（参见存储服务一章），可以启用丢弃配置。启用丢弃配置后，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，丢弃参数在 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 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：

如果所有 VM 的内核总数大于服务器上的核心数（例如，在只有 8 个内核的计算机上有 4 个 VM，每个 4 个内核），则是完全安全的。在这种情况下，主机系统将在服务器内核之间平衡 Qemu 执行线程，就像您运行标准的多线程应用程序一样。但是，Proxmox VE 将阻止您启动虚拟 CPU 内核数多于物理可用内核的虚拟机，因为这只会由于上下文切换的成本而降低性能。

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。

自定义 CPU 类型

您可以使用一组可配置的功能指定自定义 CPU 类型。这些由管理员在配置文件 /etc/pve/virtual-guest/cpu-models.conf 中维护。有关格式的详细信息，请参阅 man cpu-models.conf。

在 /nodes 上具有 Sys.Audit 特权的任何用户都可以选择指定的自定义类型。通过 CLI 或 API 为 VM 配置自定义 CPU 类型时，名称需要以 custom- 为前缀。

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 10.2.6 内存

对于每个虚拟机，您可以选择设置固定大小的内存，也可以选择要求 Proxmox VE 根据主机的当前 RAM 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 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。

虚拟显示器

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。

10.2.8 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）。

10.2.9 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>,efitype=4m,pre-enrolled-keys=1
```

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。

efitype 选项指定应使用哪个版本的 OVMF 固件。对于新 VM，此值应始终为 4m，因为它支持安全启动，并且分配了更多空间来支持将来的开发（这是 GUI 中的默认设置）。

预注册密钥指定电子硬盘是否应预加载特定于分发的密钥和 Microsoft 标准安全启动密钥。默认情况下，它还启用安全启动（可以在 VM 内的 OVMF 菜单中禁用它）。

- 提示：
  - 如果要开始在现有 VM（仍使用 2m 电子硬盘）中使用安全启动，则需要重新创建电子硬盘。为此，请删除旧的（`qm set <vmid> -delete efidisk0`）并添加一个新，如上所述。这将重置您在 OVMF 菜单中所做的任何自定义配置！

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。

10.2.10 10.2.11 可信平台模块（TPM）

受信任的平台模块是一种设备，它安全地存储机密数据（例如加密密钥），并提供用于验证系统启动的防篡改功能。

某些操作系统（例如 Windows 11）要求将此类设备连接到计算机（无论是物理还是虚拟）。

通过指定 tpm 状态卷来添加 TPM。这类似于电子硬盘，因为一旦创建，就无法更改（只能删除）。您可以通过以下命令添加一个：

```
qm set <vmid> -tpmstate0 <storage>:1,version=<version>
```

其中<storage> 是要将状态置于其上的存储，<version> 为 v1.2 或 v2.0。还可以通过 Web 界面添加一个，方法是在 VM 的硬件部分中选择“添加 → TPM 状态”。

v2.0 TPM 规范更新且受更好的支持，因此，除非你必须要使用 v1.2 TPM ，否则建议首选 v2.0。

- 注意
  - 与物理 TPM 相比，模拟 TPM 不提供任何真正的安全优势。TPM 的要点是，除非通过指定为 TPM 规范一部分的命令，否则无法轻松修改其上的数据。由于使用模拟设备时，数据存储发生在常规卷上，因此任何有权访问它的人都可以对其进行编辑。

10.2.11 10.2.11 内部虚拟机共享内存

您可以添加 VM 间共享内存设备（ivshmem），该设备允许在主机和来宾之间共享内存，也可以在多个来宾之间共享内存。

要添加此类设备，可以使用 qm：

```
# qm set <vmid> -ivshmem size=32,name=foo
```

其中大小以 MiB 为单位。该文件将位于 /dev/shm/pve-shm-$name 下（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 10.2.12 音频设备

10.2.12 音频设备

要添加音频设备，请运行以下命令：

```
qm set <vmid> -audio0 device=<device>
```

支持的音频设备包括：

- ich9-intel-hda: Intel HD Audio Controller, emulates ICH9
- intel-hda: Intel HD Audio Controller, emulates ICH6
- AC97: Audio Codec ’ 97, 对较旧的操作系统 (如 Windows XP) 非常有用
- 注意
  - 音频设备只能与 SPICE 结合使用。像微软的 RDP 这样的远程协议可以选择播放声音。要使用主机的物理音频设备，请使用 Device Passthrough(参见第 10.9 节 PCI Passthrough 和第 10.2.9 节 USB Passthrough)。

10.2.13 10.2.13 VirtIO RNG

RNG(随机数生成器) 是向系统提供熵 (随机性) 的设备。虚拟硬件-RNG 可用于将这种熵从主机系统提供给客户 VM。这有助于避免来宾中出现熵匮乏问题 (没有足够的熵可用，系统可能会变慢或遇到问题)，特别是在来宾引导过程中。要添加基于 VirtIO 的模拟 RNG，请运行以下命令：

```
qm set <vmid> -rng0 source=<source>[,max_bytes=X,period=Y]
```

source 指定熵在主机上的读取位置，必须为以下值之一：

- /dev/urandom: 非阻塞内核熵池 (首选)
- /dev/random: 阻塞内核池 (不推荐，可能会导致主机系统上的熵匮乏)
- /dev/hwrng: 通过连接到主机的硬件 RNG(如果有多个可用硬件 RNG，将使用在/sys/devices/virtual/misc/hw_random/rng_current 中选择的硬件 RNG)

可以通过 max_bytes 和 Period 参数指定限制，它们以毫秒为单位读取为每个周期的 max_bytes。但是，它不代表线性关系：1024B/1000ms 意味着在 1 秒计时器上最多有 1 KiB 的数据可用，而不是在 1 秒的过程中将 1 KiB 的数据流式传输到来宾。因此，可以使用减少周期来以更快的速率向客户注入熵。

默认情况下，该限制设置为每 1000 毫秒 (1 KiB/s)1024 字节。建议始终使用限制器，以避免来宾使用过多的主机资源。如果需要，max_bytes 的值 0 可用于禁用所有限制。

10.2.14 10.2.15. 设备启动顺序

QEMU 可以设置虚拟机应该从哪些设备启动，以及以什么顺序启动。这可以通过引导属性在配置中指定，例如：

```
boot: order=scsi0;net0;hostpci0
```

如上配置，虚拟机将首先尝试从磁盘 scsi0 引导，如果失败，它将继续尝试从 net0 引导网络，如果这也失败了，最后尝试从通过 PCIe 的设备引导（在 NVMe 的情况下被视为磁盘，否则尝试启动到选项 ROM）。

在 GUI 上，您可以使用拖放编辑器指定引导顺序，并使用复选框完全启用或禁用某些设备以进行引导。

- 注意：
- 如果您的客户机使用多个磁盘来引导操作系统或加载引导加载程序，则必须将它们全部标记为可引导（即，它们必须启用复选框或出现在配置的列表中），客户机才能引导。这是因为最近的 SeaBIOS 和 OVMF 版本仅在磁盘标记为可引导时才初始化磁盘。

在任何情况下，即使设备未出现在列表中或禁用了复选标记，只要操作系统已启动并初始化它们，在虚拟机中依旧可用。可引导标志仅影响虚拟机的 BIOS 和引导程序，并不会影响系统使用他们。

10.2.15 10.2.16. 自动启动和关闭虚拟机

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。

如果需要在主机引导和引导第一个虚拟机之间有一个延迟，请参阅 Proxmox VE 节点管理部分。

10.2.16 10.2.17. Qemu 代理

Qemu 代理是一种在 VM 内部运行的服务，在主机和虚拟机之间提供通信通道。它用于交换信息，并允许主机向虚拟机发出命令。

例如，VM 摘要面板中的 IP 地址是通过 Qemu 代理获取的。

或者在启动备份时，虚拟机通过 Qemu 代理被告知需要使用 fs-freeze 和 fs-thaw 命令同步未完成的写入。

要使 Qemu 代理正常工作，必须执行以下步骤：

- 在虚拟机中安装代理并确保它正在运行
- 在 Proxmox VE 中的启用代理

安装 Qemu 代理

对于大多数 Linux 发行版，来宾代理可用在官方软件仓库中。该软件包通常被命名为 qemu-guest-agent。

对于 Windows，它可以从Fedora VirtIO 驱动程序 ISO安装。

启用来宾代理通信

可以在虚拟机的“选项”面板中勾选 QEMU Guest Agent。要使更改生效，必须冷启动虚拟机。

可以启用 `guest-trim` 选项。启用此功能后，Proxmox VE 将在下面操作后，向虚拟机发出 TRIM 命令。

- 将磁盘移动到另一个存储
- 将 VM 实时迁移到具有本地存储的另一个节点

在精简置备的存储上，这有助于释放未使用的空间。

故障排除

虚拟机无法关闭

确保 Qemu 代理已经在虚拟机中安装并运行。

启用 Qemu 代理后，Proxmox VE 将通过 Qemu 代理向虚拟机发送电源命令，如关机。

如果 Qemu 代理未运行，则命令无法正确执行，并且 shutdown 命令将超时。

10.2.17 10.2.18 SPICE 增强

SPICE 增强可以改善远程查看器体验的可选功能。

要通过界面启用，请转到虚拟机的 Options 面板。运行以下命令以通过 CLI 启用它们：

```
qm set <vmid> -spice_enhancements foldersharing=1,videostreaming=all
```

- 注意
- 要使用这些功能，虚拟机的显示必须设置为 SPICE(Qxl)。

文件夹共享

与来宾共享本地文件夹。需要在来宾系统中安装 SPICE-webdavd 守护进程。使共享文件夹可通过位于 http://localhost:9843 的本地 WebDAV 服务器访问。对于 Windows 客户，可以从 SPICE 官方网站下载 Spice WebDAV 守护程序的安装程序。

大多数 Linux 发行版都可以安装一个名为 spice-webdavd 的软件包。

要在虚拟查看器 (远程查看器) 中共享文件夹，请转到 File → Preferences。选择要共享的文件夹，然后启用该复选框。

- 注意
- 文件夹共享目前仅在 Linux 版本的 Virt-Viewer 中有效。
- 警告
- 实验性的！该功能目前工作不可靠。

视频流

快速刷新区域会被编码到视频流中，下面有 2 个设置选项：

- all: 任何快速刷新区域都编码到视频流中
- filter: SPICE 服务器添加了额外的过滤器来决定是否应该激活视频流（目前，只跳过小窗口表面）
- off: 不执行视频检测

无法给出此选项的相关建议，请从当前环境出发。

故障排除

共享文件夹不显示

确保 WebDAV 服务已启用并在客户机中运行。在 Windows 上，它被称为 Spice webdav 代理。在 Linux 中，名称是 spice-webdavd，但可以根据发行版的不同而有所不同。

如果服务正在运行，请通过在来宾的浏览器中打开 http://localhost:9843 来检查 WebDAV 服务器。

它可以帮助重新启动 SPICE 会话。

10.3 10.3. 迁移

如果有群集，则可以使用以下命令将 VM 迁移到另一台主机

```
qm migrate <vmid> <target>
```

有下面 2 种迁移机制：

- 在线迁移
- 离线迁移

10.3.1 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。

10.3.2 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。

10.3.3 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）

10.3.4 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 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。

10.4.1 完整克隆

完整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。

10.4.2 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 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 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 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 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 10.7.2 向虚拟机增加外部磁盘镜像

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。

10.8 10.8 Cloud-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 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 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 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。

10.8.4 10.8.4 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=] [,gw6=] [,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）。

10.9 10.9 PCI(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 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
```

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 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：

1. 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
```

2. 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。

确认配置

为了确认上面的配置生效，你可以在重启之后，执行下面命令

```
    lspci -nnk
```

在输出的结果中，找到设备信息，如果有下面字段：

```
Kernel driver in use: vfio-pci
```

或者没有上面 in use，也说明设备可以用于直通。

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。注意，某些设备不支持 x-vga 参数，则需要 off。

- 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=

  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 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（Single-Root Input/Output Virtualization）技术可以支持硬件同时提供多个 VF（Virtual Function）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（Network Interface Card）。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。

1. 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。

2. 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 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：

```
I915.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 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 10.11 休眠

您可以使用界面选项 Hibernate 或使用

```
qm suspend <vmid> --todisk
```

将 VM 挂起到磁盘，这意味着内存的当前内容将保存到磁盘上，并且 VM 将被停止。在下一次启动时，将加载内存内容，并且 VM 可以从它停止的地方继续。

10.11.1 状态存储选择

如果没有为内存提供目标存储，则会自动选择它，其中第一项为：

- 1. 虚拟机配置中的存储 vmstatestorage。
- 2. 任何虚拟机磁盘中的第一个共享存储。
- 3. 任何 VM 磁盘中的第一个非共享存储。
- 4. 本地存储作为后备。

10.12 10.12 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。

10.12.1 10.12.1 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
```

删除 VM 总是会将其从访问控制列表和防火墙配置中移除，如果你想将虚拟机从备份任务、复制或者 HA 资源中移除，你还需要添加选项 `--purge`

```
qm destroy 300 --purge
```

移动磁盘到不同的存储点。

```
qm move-disk 300 scsi0 other-storage
```

重新分配磁盘到另外的 VM。这把源 VM 的 `scsi1` 重新配置到目标 VM 的 `scsi3`。在移动过程中，会将磁盘重新按照目标 VM 的磁盘格式命令。

```
qm move-disk 300 scsi1 --target-vmid 400 --target-disk scsi3
```

10.13 10.13 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。

10.13.1 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。

10.13.2 虚拟机配置文件示例

```
boot: order=virtio0;net0
cores: 1
sockets: 1
memory: 512
name: webmail
ostype: l26
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。

10.13.3 10.13.1 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。

10.13.4 10.13.2 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：

```
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）。

可以选择使用 `vmstate` 选项保存虚拟机的内存状态。关于如何选择虚拟机内存状态的保存目录，请参阅 10.11 休眠章节.

10.13.5 10.13.3 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:args 仅供专家使用。

- audio0: device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>]

  配置虚拟声卡，与 Spice/QXL 配合很有用。

  device=<ich9-intel-hda|intel-hda|AC97> 选择声卡的类型

  driver=<none | spice> (default = spice) 选择声卡的后端驱动，默认为 spice

- 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: [[legacy=]<[acdn]{1,4}>] [,order=<device[;device...]>]

  legacy=<[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）。现已弃用，请改用 order

  order=<device[;d evice...]>, 虚拟机将按照此规则进行启动。磁盘、光驱和直通存储 USB 设备将直接从中启动，NIC 将加载 PXE，PCIe 设备如果是磁盘（Nvme）就会启动或加载 OPTION ROM（例如 RAID 控制器、硬件 NIC）。

  注意，只有在这里被标记成可启动设备，才能由虚拟机的（BIOS/UEFI）加载。如果您需要多个磁盘进行引导（例如 software-raid），则需要在此处指定所有磁盘。

  使用 order 将会忽略 legacy 的值

- bootdisk: (ide|sata|scsi|virtio)\d+

  指定启动磁盘，已经弃用，请使用 boot: order=xx

- cdrom:<volume>

  光驱，相当于-ide2 的别名。

- cicustom: [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<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 | opennebula>

  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 : (1 -N) (default = 1)

  每个插槽的 CPU 核心数量

- cpu: [[cputype=]<string>] [,flags=<+FLAG[;-FLAG...]>] [,hidden=<1|0>] [,hv-vendor-id=<vendor-id>] [,phys-bits=<8-64|host>] [,reported-model=<enum>]

  - 模拟 CPU 类型。
  - cputype=<string> (default = kvm64)

    模拟的 CPU 类型。可以使用默认类型，也可以自定义类型。自定义类型将以 custom- 开头。

  - 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

  - phys-bits=<8-64|host>

    报告给客户机操作系统的物理内存地址位。应小于或等于主机的物理内存地址位。设置为 host 以使用主机 CPU 中的值

  - reported-model=<enum>

    486 | Broadwell | Broadwell-IBRS | Broadwell-noTSX | Broadwell-noTSX-IBRS | Cascadelake-Server | Cascadelake-Server-noTSX | Conroe | EPYC | EPYC-IBPB | EPYC-Rome | Haswell | Haswell-IBRS | Haswell-noTSX | Haswell-noTSX-IBRS | Icelake-Client | Icelake-Client-noTSX | Icelake-Server | Icelake-Server-noTSX | IvyBridge | IvyBridge-IBRS | KnightsMill | Nehalem | Nehalem-IBRS | Opteron_G1 | Opteron_G2 | Opteron_G3 | Opteron_G4 | Opteron_G5 | Penryn | SandyBridge | SandyBridge-IBRS | Skylake-Client | Skylake-Client-IBRS | Skylake-Client-noTSX-IBRS | Skylake-Server | Skylake-Server-IBRS

| Skylake-Server-noTSX-IBRS | Westmere | Westmere-IBRS | athlon | core2duo | coreduo | host | kvm32 | kvm64 | max | pentium | pentium2 | pentium3 | phenom | qemu32 | qemu64 (default = kvm64)

所选择的型号及厂商，将会传递给虚拟机，但必须选择 QEMU 支持的 CPU 模型。只有自定义 CPU 模型才能将自定义厂商 ID 传递给虚拟机，默认的 CPU 模型会始终传递自身的默认属性。

- 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 = cgroup v1: 1024, cgroup v2: 100)

  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> [,efitype=<2m|4m>] [,format=<enum>] [,pre-enrolled-keys=<1|0>] [,size=<DiskSize>]`

  配置用于存储 EFI 变量的磁盘。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。请注意，此处忽略 SIZE_IN_GiB，而是将默认 EFI 变量复制到卷中。

  - efitype=`<2m | 4m>` (default = 2m)

    EFI 变量的大小，4m 最新且推荐使用，用于安全启动。为了向后兼容性，如果没有制定，则使用 2m。

  - 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。

  - pre-enrolled-keys=`<boolean>` (default = 0)

    预注册密钥。如果与 efitype=4m 一起使用，则使用 EFI 模板并注册特定于分发的密钥和 Microsoft 标准密钥。请注意，这将默认启用安全启动，但仍可以从 VM 中 BIOS 将其关闭。

  - 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>`

回调脚本，将在虚拟机生命周期的各个步骤中执行的脚本。如启动阶段，关闭阶段。

- hostpci[n]: [host=]<HOSTPCIID[;HOSTPCIID2…]> [,mdev=][,pcie=<1|0>] [,rombar=<1|0>] [,romfile=] [,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|io_uring>] [,backup=<1|0>] [,bps=] [,bps_max_length=] [,bps_rd=] [,bps_rd_max_length=] [,bps_wr=] [,bps_wr_max_length=] [,cache=] [,cyls=] [,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=] [,heads=] [,iops=] [,iops_max=] [,iops_max_length=] [,iops_rd=] [,iops_rd_max=] [,iops_rd_max_length=] [,iops_wr=] [,iops_wr_max=] [,iops_wr_max_length=] [,mbps=] [,mbps_max=] [,mbps_rd=] [,mbps_rd_max=] [,mbps_wr=] [,mbps_wr_max=] [,media=<cdrom|disk>] [,model=] [,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,secs=] [,serial=] [,shared=<1|0>] [,size=] [,snapshot=<1|0>] [,ssd=<1|0>] [,trans=<none|lba|auto>] [,werror=] [,wwn=]

  配置 IDE 类型虚拟硬盘或光驱（n 的值为 0-3）。使用 STORAGE_ID:SIZE_IN_GiB 语法去分配虚拟硬盘

- aio=<io_uring | native | threads>

  指定异步 io 模型。默认为 io_uring

- 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>

  突发读操作最大时间长度，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。

  注意：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>]

  loud-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：需要和 ip 配合使用

- gw6=<GatewayIPv6>

  IPv6 的默认网关

  注意：需要和 ip6 配合使用

- 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=]

  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。

- keephugepages: <boolean> (default = 0)

  与 huepages 一起使用。如果启用，则在 VM 关闭后不会删除大页，可用于后续启动。

- 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+)+(+pve\d+)?(.pxe)?|q35|pc-q35-\d+(.\d+)+(+pve\d+)?(.pxe)?|virt(?:-\d+(.\d+)+)?(+pve\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=] [,firewall=<1|0>] [,link_down=<1|0>] [,macaddr=XX:XX:XX:XX:XX:XX] [,mtu=] [,queues=] [,rate=] [,tag=] [,trunks=<vlanid[;vlanid…]>] [,=]

  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>

    是否在此虚拟机网卡上启用防火墙功能。

  - link_down=<boolean>

    将此虚拟网卡设为断开状态。

  - macaddr=<XX:XX:XX:XX:XX:XX>

    MAC 地址。

- model=<e1000 | e1000-82540em | e1000-82544gc | e1000-82545em | e1000e | i82551 | i82557b | i82559er | ne2k_isa | ne2k_pci | pcnet | rtl8139 | virtio | vmxnet3>

  虚拟网卡类型。其中 virtio 性能最好。若虚拟机不支持 virtio，最好使用 e1000。

- mtu=<integer> (1 - 65520)

  只针对于 virtio 类型虚拟机网卡，强制设置 mtu。设为 1 则继承网桥的 mtu。

- 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。

- tag=<integer> (1 - 4094)

  在此虚拟网卡的数据包上自动标记的 vlan 标签。

- trunks=<vlanid[;vlanid...]>

  虚拟网卡上允许通过的 vlan 标签

- numa: <boolean> (default = 0)

  启用/禁用 NUMA。

- numa[n]: cpus=<id[-id];...> [,hostnodes=<id[-id];...>] [,memory=<number>] [,policy=<preferred|bind|interleave>]

  设置 NUMA 拓扑

  - 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 | win11 | 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 10/2016/2019

- win11

  Microsoft Windows 11/2022

- l24

  Linux 2.4 Kernel

- l26

  Linux 2.6 - 5.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。

- rng0: [source=]</dev/urandom|/dev/random|/dev/hwrng> [,max_bytes=<integer>] [,period=<integer>]

  配置基于 VirtIO 的随机数生成器。

  - max_bytes=<integer> (default = 1024)

    每毫秒内允许向虚拟机内注入的最大熵字节数，使用/dev/random 作为源时，首选较低的值。使用 0 禁用限制（可能很危险!）。

  - period=<integer> (default = 1000)

    每隔几毫秒，熵注入配额就会重置，允许来宾检索另一个 max_bytes 熵。

  - source=</dev/hwrng | /dev/random | /dev/urandom>

    主机上要从中收集熵的文件。在大多数情况下，/dev/urandom 应该优先于/dev/random 以避免主机上的熵饥饿问题。使用 urandom 不会以任何有意义的方式降低安全性，因为它仍然是从真实熵中播种的，并且提供的字节很可能也会与来宾上的真实熵混合。/dev/hwrng 可用于从主机传递硬件 RNG。

- sata[n]: [file=]<volume> [,aio=<native|threads|io_uring>] [,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）。使用 STORAGE_ID:SIZE_IN_GiB 语法去分配虚拟硬盘

- aio=<io_uring | native | threads>

  指定异步 io 模型。默认为 io_uring

- 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>

  突发读操作最大时间长度，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。

  注意：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|io_uring>] [,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>] [,ro=<1|0>] [,
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 0-30）。使用 STORAGE_ID:SIZE_IN_GiB 语法去分配虚拟硬盘

- aio=<io_uring | native | threads>

  指定异步 io 模型。默认为 io_uring

- 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>

  SCSI 虚拟硬盘所基于的存储服务卷名称。

- format=<cloop | cow | qcow | qcow2 | qed | raw | vmdk>

  SCSI 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>

  突发读操作最大时间长度，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。

- ro=<boolean>

  设置磁盘只读

- scsiblock= (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。

  注意：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:

  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 : (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=][，manufacturer=] [,product=] [,serial=] [,sku=] [,uuid=] [,version=]

  设置 SMBIOS 的类型 1 字段。

  - base64=

    用于设置 SMBIOS 是否采用 base64 编码的参数值。

  - family =

    设置 SMBIOS1 家族名称。

  - manufacturer =

    设置 SMBIOS1 厂商名。

  - product =

    设置 SMBIOS1 产品 ID。

  - serial =

    设置 SMBIOS1 序列号。

  - sku =

    设置 SMBIOS1 的 SKU 字符串。

  - uuid =

    设置 SMBIOS1 的 UUID。

  - version =

    设置 SMBIOS1 版本。

- smp : (1 -N) (default = 1)

  设置 CPU 数量。请使用选项-sockets 代替。

- sockets : (1 -N) (default = 1)

  设置 CPU 的 sockets 数量。

- sshkeys:

  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 : (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 : (default = 0)

  启用/禁用时间漂移修正。

- template : (default = 0)

  启用/禁用模板。

- tpmstate0：[file=] [，size=] [，version=<v1.2|v2.0>]

  配置用于存储 TPM 状态的磁盘。使用特殊语法 STORAGE_ID：SIZE_IN_GiB 分配新卷。请注意，此处忽略 SIZE_IN_GiB，将始终使用默认大小 4 MiB。格式也固定为原始格式。

  - file=

    TMP 虚拟硬盘所基于的存储服务卷名称

  - size=

    TPM 虚拟硬盘容量。仅供显示使用，并不能影响实际容量大小。

  - version=<v1.2 | v2.0> (default = v2.0)

    TPM 版本。v2.0 是最新的，因此是首选版本。注意，设置之后无法修改。

- unused[n]: [file=]

  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。

  - file=

    虚拟硬盘所基于的存储服务卷名称。

- 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）’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 = (default = 0)

    标识是否为 USB3 设备或端口。

- vcpus : (1 -N) (default = 0)

  可热插拔的虚拟 cpu 数量。

- vga: [[type=]] [,memory=]

  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= (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=] [,aio=<native|threads|io_uring>] [,backup=<1|0>] [,bps=] [,bps_max_length=] [,bps_rd=] [,bps_rd_max_length=] [,bps_wr=] [,bps_wr_max_length=] [,cache=] [,cyls=] [,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=] [,heads=] [,iops=] [,iops_max=] [,iops_max_length=] [,iops_rd=] [,iops_rd_max=] [,iops_rd_max_length=] [,iops_wr=] [,iops_wr_max=] [,iops_wr_max_length=] [,iothread=<1|0>] [,mbps=] [,mbps_max=] [,mbps_rd=] [,mbps_rd_max=] [,mbps_wr=] [,mbps_wr_max=] [,media=<cdrom|disk>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,ro=<1|0>] [,secs=] [,serial=] [,shared=<1|0>] [,size=] [,snapshot=<1|0>] [,trans=<none|lba|auto>] [,werror=]

  配置 virtio 类型虚拟硬盘或光驱（n 的值为 0-30）。使用 STORAGE_ID:SIZE_IN_GiB 语法去分配虚拟硬盘

  - aio=<io_uring | native | threads>

    指定异步 io 模型。默认为 io_uring

  - 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>

  SCSI 虚拟硬盘所基于的存储服务卷名称。

- format=<cloop | cow | qcow | qcow2 | qed | raw | vmdk>

  virtio 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>

  突发读操作最大时间长度，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。

- ro=<boolean>

  设置磁盘只读

- 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。

  注意：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。

- vmgenid: (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:

  虚拟机状态卷/文件的默认存储。

- watchdog : [[model=]<i6300esb|ib700>] [,action=]

  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。

10.14 10.14 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。

CHAPTER 11

第十一章 Proxmox 容器管理工具

容器是完全虚拟化计算机（VM）的轻量级替代方案。它们使用运行它们的主机系统的内核，而不是模拟完整的操作系统（OS）。这意味着容器可以直接访问主机系统上的资源。

容器的运行时成本很低，通常可以忽略不计。但是，需要考虑一些缺点：

- 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 Linux Containers（LXC）作为其底层容器技术。“Proxmox Container Toolkit”（pct）通过提供详细的任务界面，简化了 LXC 的使用和管理。

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”，而不是“应用容器”

如果你想运行微容器（如 docker），建议在 KVM 虚拟机中运行。这将为你提供应用程序容器化的所有功能，同时还提供 VM 所拥有的优势，例如与主机的强隔离和实时迁移功能，容器无法做到这一点。

11.1 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 11.2 支持的发行版

官方支持的发行版列表可以在下面找到。

以下发行版的模板可通过我们的仓库获得。您可以使用 pveam 工具或图形用户界面来下载它们。

11.2.1 11.2.1. Alpine Linux

> Alpine Linux is a security-oriented, lightweight Linux distribution based on musl libc and busybox.
>
> —https://alpinelinux.org

有关当前支持的版本，请参阅：

https://alpinelinux.org/releases/

11.2.2 11.2.2. Arch Linux

> Arch Linux, a lightweight and flexible Linux® distribution that tries to Keep It Simple.
>
> —https://archlinux.org/

Arch Linux 正在使用滚动发布，有关更多详细信息，请参阅其 wiki:

https://wiki.archlinux.org/title/Arch_Linux

11.2.3 11.2.3. CentOS, Almalinux, Rocky Linux

CentOS / CentOS Stream

> The CentOS Linux distribution is a stable, predictable, manageable and reproducible platform derived from the sources of Red Hat Enterprise Linux (RHEL)
>
> —https://centos.org

有关当前支持的版本，请参阅：

https://wiki.centos.org/About/Product

Almalinux

> An Open Source, community owned and governed, forever-free enterprise Linux distribution, focused on long-term stability, providing a robust production-grade platform. AlmaLinux OS is 1:1 binary compatible with RHEL® and pre-Stream CentOS.
>
> —https://almalinux.org

有关当前支持的版本，请参阅：

https://en.wikipedia.org/wiki/AlmaLinux#Releases

Rocky Linux

> Rocky Linux is a community enterprise operating system designed to be 100% bug-for-bug compatible with America’s top enterprise Linux distribution now that its downstream partner has shifted direction.
>
> —https://rockylinux.org

有关当前支持的版本，请参阅：

https://en.wikipedia.org/wiki/Rocky_Linux#Releases

11.2.4 11.2.4. Debian

> Debian is a free operating system, developed and maintained by the Debian project. A free Linux distribution with thousands of applications to meet our users’ needs.
>
> —https://www.debian.org/intro/index#software

有关当前支持的版本，请参阅：

https://www.debian.org/releases/stable/releasenotes

11.2.5 11.2.5. Devuan

> Devuan GNU+Linux is a fork of Debian without systemd that allows users to reclaim control over their system by avoiding unnecessary entanglements and ensuring Init Freedom.
>
> —https://www.devuan.org

有关当前支持的版本，请参阅：

https://www.devuan.org/os/releases

11.2.6 11.2.6. Fedora

> Fedora creates an innovative, free, and open source platform for hardware, clouds, and containers that enables software developers and community members to build tailored solutions for their users.
>
> —https://getfedora.org

有关当前支持的版本，请参阅：

https://fedoraproject.org/wiki/Releases

11.2.7 11.2.7. Gentoo

> a highly flexible, source-based Linux distribution.
>
> —https://www.gentoo.org

Gentoo 正在使用滚动发布

11.2.8 11.2.8. OpenSUSE

> The makers’ choice for sysadmins, developers and desktop users.
>
> —https://www.opensuse.org

有关当前支持的版本，请参阅：

https://get.opensuse.org/leap/

11.2.9 11.2.9. Ubuntu

> Ubuntu is the modern, open source operating system on Linux for the enterprise server, desktop, cloud, and IoT.
>
> —https://ubuntu.com/

有关当前支持的版本，请参阅：

https://wiki.ubuntu.com/Releases

11.3 11.3 容器映像

容器镜像（有时也称为“模板”或“应用程序”）是包含运行容器的所有内容的 tar 包。

Proxmox VE 本身为最常见的 Linux 发行版提供了多种基本模板。可以使用 GUI 或 pveam（Proxmox VE Appliance Manager 的缩写）命令行实用程序下载它们。此外，还提供 TurnKey Linux 容器模板下载。

可用模板的列表通过 pve-daily-update 计时器每天更新。您还可以通过执行以下命令手动触发更新：

```
pveam update
```

要查看可用映像的列表，请运行：

```
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，例如可以指定查看系统镜像：

列出可用镜像

```
pveam available --section system
system           alpine-3.12-default_20200823_amd64.tar.xz
system           alpine-3.13-default_20210419_amd64.tar.xz
system           alpine-3.14-default_20210623_amd64.tar.xz
system           archlinux-base_20210420-1_amd64.tar.gz
system           centos-7-default_20190926_amd64.tar.xz
system           centos-8-default_20201210_amd64.tar.xz
system           debian-9.0-standard_9.7-1_amd64.tar.gz
system           debian-10-standard_10.7-1_amd64.tar.gz
system           devuan-3.0-standard_3.0_amd64.tar.gz
system           fedora-33-default_20201115_amd64.tar.xz
system           fedora-34-default_20210427_amd64.tar.xz
system           gentoo-current-default_20200310_amd64.tar.xz
system           opensuse-15.2-default_20200824_amd64.tar.xz
system           ubuntu-16.04-standard_16.04.5-1_amd64.tar.gz
system           ubuntu-18.04-standard_18.04.1-1_amd64.tar.gz
```

(续下页)

(接上页)

```
system          ubuntu-20.04-standard_20.04-1_amd64.tar.gz
system          ubuntu-20.10-standard_20.10-1_amd64.tar.gz
system          ubuntu-21.04-standard_21.04-1_amd64.tar.gz
```

在使用此类模板之前，需要将它们下载到其中一个存储上。如果不确定是哪一个，可以 local 存储。对于群集环境，最好使用共享存储，以便所有节点都能访问这些镜像。

```
pveam download local debian-10.0-standard_10.0-1_amd64.tar.gz
```

现在，你已准备好使用该映像创建容器，并且可以在本地存储上列出所有下载的映像，如下所示：

```
pveam list local
local:vztmpl/debian-10.0-standard_10.0-1_amd64.tar.gz  219.95MB
```

提示: 您还可以使用 Proxmox VE Web 界面 GUI 下载，列出和删除容器模板。

pct 使用它们来创建新容器，例如：

```
 pct create 999 local:vztmpl/debian-10.0-standard_10.0-1_amd64.tar.gz
```

上述命令显示完整的 Proxmox VE 卷标识符。它们包括存储名称，大多数其他 Proxmox VE 命令都可以使用它们。例如，您可以稍后通过以下方式删除该图像：

```
 pveam remove local:vztmpl/debian-10.0-standard_10.0-1_amd64.tar.gz
```

11.4 11.4. 容器设置

11.4.1 11.4.1 通用设置

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。

非特权容器

非特权容器采用了名为用户命名空间 usernamespaces 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 init 系统，请注意，在容器内运行的 systemd 版本应等于或大于 220。

特权容器

容器中的安全性是通过使用强制访问控制（AppArmor）、seccomp 筛选器和 Linux 内核 namespace 来实现的。LXC 团队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。。这就是为什么特权容器应仅在受信任的环境中使用的原因。

11.4.2 11.4.2 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。

11.4.3 11.4.3 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）。

11.4.4 11.4.4 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 256 个挂载点，分别对应于参数 mp0 到 mp255。具体设置项目如下：

- rootfs: `[volume=]<volume> [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<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。使用 STORAGE_ID:SIZE_IN_GiB 语法分配新的存储卷。

  - 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，同时在容器内的 =/path/in/container 创建挂载点。

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 `mp0:/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。

11.4.5 11.4.5 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。

11.4.6 11.4.6 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。

11.5 11.5 安全注意事项

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。

11.5.1 11.5.1 AppArmor

AppArmor 配置文件用于限制对可能存在危险的操作的访问。某些系统调用 (即 mount) 被禁止执行。

要跟踪 AppArmor 活动，请使用：

```
dmesg | grep apparmor
```

尽管不建议使用 AppArmor，可以对容器禁用 AppArmor。但这就带来了安全风险。如果系统配置错误或存在 LXC 或 Linux 内核漏洞，则某些 syscall 在容器内执行时可能会导致权限提升。

要禁用容器的 AppArmor，请将以下行添加到位于/etc/pve/lxc/CTID.conf 的容器配置文件中：

```
lxc.apparmor.profile = unconfined
```

- 注意：

  请不要用于生产环境!

11.5.2 11.5.2. Control Groups (cgroup)

`cgroup` 是 Linux 内核的一个功能，用来限制、控制与分离一个进程组的资源。

通过 cgroup 控制的主要资源是 CPU、内存和 swap 限制以及对主机设备的访问。cgroups 还用于在拍摄快照之前“冻结”容器。

目前有 2 个版本的 cgroups 可用，`legacy` 和 `cgroupv2`.

从 Proxmox VE 7.0 开始，默认的是纯 `cgroupv2` 环境。以前使用“混合”设置，其中资源控制主要在 cgroupv1 中完成，并使用额外的 cgroupv2 控制器，该控制器可以通过 cgroup_no_v1 内核命令行参数接管一些子系统。(有关详细信息，请参阅内核参数文档。)

CGroup 版本兼容性

关于 Proxmox VE 的纯 cgroupv2 和旧混合环境的主要区别在于，内存和 swap 现在由 cgroupv2 独立控制。容器的内存和 swap 设置可以直接映射到这些值，而以前只能限制内存限制以及内存和交换的总和限制。

另一个重要区别是，设备控制的配置方式完全不同。因此，目前在纯 cgroupv2 环境中不支持文件系统配额。

在纯 cgroupv2 环境中运行需要容器操作系统支持 cgroupv2。运行 systemd 231 或更高版本的容器支持 cgroupv2 [44]，不使用 systemd 作为 init 系统的容器也不支持。

> CentOS 7 和 Ubuntu 16.10 是两个著名的 Linux 发行版，它们的系统版本太旧了，无法在 cgroupv2 环境中运行，您可以执行下面方案：
>
> - 将整个发行版升级到新版本。对于上面的例子，可以是 Ubuntu 18.04 或 20.04，以及 CentOS 8（或 RHEL/CentOS 衍生产品，如 AlmaLinux 或 Rocky Linux）。这有利于获得最新的错误和安全修复，通常还有新功能，并延长了 EOL 日期。

- 升级容器 systemd 版本。如果发行版提供了一个 `backports` 存储库，这可能是一个简单快捷的权宜之计。
- 将容器或其服务移动到 kvm 虚拟机。虚拟机与主机的交互要少得多，这也是大家要用虚拟机装远古系统的原因。
- 切换回旧版 cgroup 控制器。请注意，虽然它可能是一个有效的解决方案，但它不是一个永久性的解决方案。未来的 Proxmox VE 主要版本（例如 8.0）很可能无法再支持 legacy 控制器。

更改 cgroup 版本

- 注意：如果不需要文件系统配额，并且所有容器都支持 cgroupv2，建议坚持新的默认值（cgroupv2）。

要切换回以前的版本，可以使用以下内核命令行参数：

```
systemd.unified_cgroup_hierarchy=0
```

编辑内核引导命令行请参考 3.12.6，以了解在哪里添加参数。

11.6 11.6 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。

OS 类型检测是通过测试容器中的某些文件来完成的。Proxmox VE 首先检查/etc/os-release 文件 [46]。如果该文件不存在，或者它不包含可明确识别的分发标识符，则检查以下特定于分发的发布文件。

- Ubuntu

  test/etc/lsb-release (DISTRIB_ID=Ubuntu)

- Debian

  test/etc/debian_version

- Fedora

  test/etc/fedora-release

- RedHat or CentOS

  test/etc/redhat-release

- ArchLinux

  test/etc/arch-release

- Alpine

  test/etc/alpine-release

- Gentoo

  test/etc/gentoo-release

- 注意

  如果配置的 Ostype 与自动检测的类型不同，则容器启动失败。

11.7 11.7. 容器存储

Proxmox VE LXC 容器存储模型比传统的容器存储模型更灵活。一个容器可以有多个挂载点。这能为每个应用程序使用最适合的存储空间.

例如，容器的根文件系统可以使用慢速、经济的存储，而数据库可以通过第二个挂载点使用快速的分布式存储。有关更多详细信息，请参阅 Mount Points 部分。

任何被 Proxmox VE 支持的存储类型都可以被容器使用。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。

此外，可以使用绑定装载直接装载本地设备或本地目录。这使得访问容器内的本地资源几乎没有开销。绑定挂载可以作为在容器之间共享数据的简单方法。

11.7.1 11.7.1 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。

11.7.2 11.7.2 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。

- 注意

  这需要使用 legacy cgroups。只有基于 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 /。

11.7.3 11.7.3 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。

11.7.4 11.7.4 备份容器挂载点

要在备份中包括挂载点，请在容器配置中为其启用备份选项。对于现有装载点 mp0

```
mp0: guests:subvol-100-disk-1,mp=/root/files,size=8G
```

添加 `backup=1` 以启用备份功能。

```
mp0: guests:subvol-100-disk-1,mp=/root/files,size=8G,backup=1
```

- 注意：

  当在 GUI 上创建新的挂载点时，备份功能将自动启用。

要禁用备份，请在上面的配置选项中添加 `backup=0`，或者 GUI 面板上取消勾选 backup。

11.7.5 11.7.5 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。

11.8 11.8 备份和恢复

11.8.1 11.8.1 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。

11.8.2 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）。

11.9 11.9 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。

11.9.1 11.9.1 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
```

删除容器总是会将其从访问控制列表和防火墙配置中移除，如果你想将容器从备份任务、复制或者 HA 资源中移除，你还需要添加选项–purge

```
 pct destroy 100 --purge
```

移动挂载点到其他的存储

```
pct move-volume 100 mp0 other-storage
```

重新分配磁盘到另外的容器。这把源容器的 mp0 重新配置到目标 CT 的 mp0。在移动过程中，会将磁盘重新按照目标容器的磁盘格式命令。

```
pct move-volume 100 mp0 --target-vmid 200 --target-volume mp1
```

11.9.2 11.9.2 获取调试日志

如果 `pct start` 无法启动特定容器，通过添加–debug 标志（将 CTID 替换为容器的 CTID）来收集调试输出，可能会有所帮助：

```
pct start CTID --debug
```

或者，您可以使用下面 `lxc-start` 命苦，该命令将调试日志保存到-o 输出选项指定的文件中：

```
lxc-start -n CTID -F -l DEBUG -o /tmp/lxc-CTID.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。

11.10 11.10 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。

由于技术限制，运行的容器无法实时迁移。您可以进行重启迁移，该迁移会先将容器关闭再迁移，最后在目标节点上再次启动容器。由于容器非常轻巧，这通常只会产生数百毫秒的停机时间。

重启迁移可以通过 Web 界面或使用带有 pct 迁移命令的–restart 标志来完成。

重启迁移将关闭容器或者在超时后（默认为 180 秒）强制关闭。然后，它将像离线迁移一样迁移容器，完成后，它会在目标节点上启动。

11.11 11.11 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。

如果更改无法热插拔，它将标记为待处理的更改（在 GUI 中以红色显示）。它们只有在重新启动容器后才会应用。

11.11.1 11.11.1 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。

11.11.2 11.11.2 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）。

11.11.3 11.11.3 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
↪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sys
↪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
↪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/.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。

11.12 11.12 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。

CHAPTER 12

第十二章软件定义网络

软件定义网络 (SDN) 功能允许用户在数据中心级别创建虚拟网络 (Vnet)。

- 警告

  SDN 目前是 Proxmox VE 中的一个实验性功能。关于它的文档也在开发中，请在我们的邮件列表或论坛 1.10 部分询问问题和反馈。

12.1 12.1. 安装

你需要在每个节点上安装 `libpve-network-perl` 和 `ifupdown2` 才能启用实验性的 SDN 功能。

```
apt update
apt install libpve-network-perl ifupdown2
```

随后在`/etc/network/interfaces` 末尾添加下面代码，这样 SDN 配置文件才能被读取和启用

```
source /etc/network/interfaces.d/*
```

12.2 12.2 概述

Proxmox VE SDN 使用灵活的软件控制配置，允许对虚拟来宾网络进行分离和细粒度控制。

隔离由区域组成，一个区域就是它自己的虚拟隔离网络区域。VPN 是连接到区域的一种虚拟网络。根据专区使用的类型或插件的不同，它可能会有不同的行为，并提供不同的功能、优点或缺点。通常，vNet 显示为带有 VLAN 或 VXLAN 标签的普通 Linux 网桥，但有些网桥也可以使用第 3 层路由进行控制。从群集范围的数据中心 SDN 管理界面提交配置后，将在每个节点上本地部署 VNET。

12.2.1 12.2.1 主要配置

配置在数据中心 (群集范围) 级别完成，它将保存在共享配置文件系统中的配置文件中：/etc/pve/sdn

在 Web 界面上，SDN 功能有 4 个主要部分用于配置

- SDN：SDN 状态概述。
- 区域：创建和管理虚拟分离的网络区域。
- VNET：为虚拟机提供分区的每节点构造块。

还有一些选项

- 控制器: 适用于控制第 3 层路由的复杂设置
- 子网: 定义在 vnet 上的网段
- IPAM: 使用外部 IP 管理工具管理虚拟机 IP
- DNS: 定义一个 DNS 服务器，使用其 api 注册虚拟机的主机名和 IP。

12.2.2 12.2.2. SDN

这是主状态面板。在这里，您可以看到不同节点上区域的部署状态。这里有一个 Apply 按钮，用于推送和重新加载所有群集节点上的本地配置。

12.3 12.3 区域

区域将定义虚拟分离的网络。

它可以使用不同的技术进行分离：

- VLAN：虚拟 LAN 是细分 LAN 的经典方法
- QinQ：堆叠 VLAN(正式名称为 IEEE 802.1ad)
- VXLAN：(第 2 层 VXLAN)
- Simple: 隔离网桥，简单的 3 层路由网桥 (NAT)

- bgp-evpn：使用第 3 层边界网关协议路由的 VXLAN

您可以将区域限制为特定节点。还可以在区域上添加权限，以限制用户仅使用特定区域且仅使用该区域中的 VNET。

12.3.1 12.3.1. 通用选项

- 节点

  仅在这些节点上部署并允许使用为此区域配置的 VNet。

- ipam

  可选配置，配置在此区域使用的 ip 管理工具

- dns 服务器

  可选配置，选择 DNS 服务器

- 反向 Dns 服务器

  可选配置，选择反向 DNS 服务器

- DNS 域

  可选配置。设置 DNS 域名。这样主机名就会是<hostname>.<domain> 格式。DNS 域需要提前在 DNS 服务上配置。

12.3.2 12.3.2. Simple 区域

这是最简单的插件，它将创建一个隔离的 vnet 桥接器。此网桥不链接到物理接口，VM 流量仅是节点的本地流量。它还可用于 NAT 或路由设置。

12.3.3 12.3.3 VLAN 区域

这是最简单的插件，它将使用现有的本地 Linux 或 OVS 网桥，并在其上管理 VLAN。使用 SDN 模块的好处是，您可以使用特定的 VNet VLAN 标签创建不同的区域，并将虚拟机限制在不同的区域。

指定 VLAN 配置选项：

bridge

使用已在每个本地节点上配置的此本地网桥或 OVS 交换机。

12.3.4 12.3.4. QinQ 区域

QinQ 是堆叠 VLAN。为分区定义的第一个 VLAN 标记 (所谓的 service-VLAN)，以及为 vnet 定义的第二个 VLAN 标记

- 注意

  您的物理网络交换机必须支持堆叠 VLAN!

指定 QinQ 配置选项：

- bridge

  已在每个本地节点上配置的本地 VLAN-aware 网桥

- VLAN 服务

  该 zone 的主 VLAN 标签

- VLAN 服务协议

  允许指定 802.1q（默认）或 802.1ad 作为 vlan 类型。

- mtu

  由于标签的双层堆叠，QinQ VLAN 需要多 4 个字节。例如，如果物理接口 MTU 为 1500，则将 MTU 减少到 1496。

12.3.5 12.3.5. VXLAN 区域

VXLAN 插件将在现有网络 (名为 Underlay) 之上建立隧道 (名为 Overlay)，将第 2 层以太网帧封装在第 4 层 UDP 数据报中，使用 4789 作为默认目的端口。例如，您可以在公共 Internet 网络节点之上创建专用 IPv4 VXLAN 网络。

这只是第 2 层隧道，不可能在不同的 VNET 之间进行路由。

每个 VPN 将使用范围 (1-16777215) 中的特定 VxLAN ID。

指定 EVPN 配置选项：

- 对端地址列表

  要通过其进行通信的所有节点的 IP 列表。也可以是外部节点。

- mtu

  由于 VXLAN 封装使用 50 字节，因此 MTU 需要比传出物理接口低 50 字节。

12.3.6 12.3.6. EVPN 区域

这是所有支持的插件中最复杂的。

BGP-EVPN 允许创建路由的第 3 层网络。EVPN 的 VPN 可以具有任播 IP 地址和 MAC 地址。每个节点上的网桥 IP 都是相同的，因此虚拟访客可以将该地址用作网关。

路由可以通过 VRF(虚拟路由和转发) 接口跨来自不同区域的 VNet 工作。

指定 EVPN 配置选项：

- VRF VXLAN Tag

  这是用于在 vnet 之间路由互连的 vxlan-id，它必须不同于 vnet 的 vxlan-id

- controller

  首先需要定义 EVPN 控制器 (参见控制器插件部分)

- VNet MAC 地址

  此区域中所有 VNet 的唯一任播 MAC 地址。如果未定义，将自动生成。

- 退出节点

  如果要定义一些 proxmox 节点，作为从 evpn 网络到真实网络的出口网关，则使用此选项。配置的节点将在 EVPN 网络中宣布默认路由

- Advertise Subnets（发布子网）

  可选配置。如果存在一些消极 VM/CT（例如，接口上存在多个 ip 地址，并且任播网关看不到来自这些 ip 的流量，则无法在 evpn 网络内访这些 ip 地址）。在这种情况下，此选项将宣布 evpn 网络中的完整子网.

- Exit Nodes local routing（出口节点本地路由）

  可选配置。如果需要从出口节点访问 vm/ct 服务，这是一个特殊选项。（默认情况下，出口节点仅允许在真实网络和 evpn 网络之间转发流量）。

- MTU

  由于 VXLAN 封装使用 50 个字节，因此 MTU 需要比传出物理接口的最大 MTU 低 50 个字节。

12.4 12.4 VNets

VNet 的基本形式只是一个 Linux 网桥，它将本地部署在节点上并用于虚拟机通信。

VNet 属性包括：

- 名称：用于命名和标识 vNet 的 8 个字符的 ID。
- 别名：可选的较长名称，如果 ID 不够。
- 区：此 vNet 的关联区域。

- 标签：唯一的 VLAN 或 VXLAN ID。
- VLAN 感知：允许在虚拟机或容器 vNIC 配置中添加额外的 VLAN 标签，或允许来宾操作系统管理 VLAN 的标签。

12.4.1 12.4.1. 子网

sub-net（子网）可以定义特定的 IP 网络（IPv4 或 IPv6）。对于每个 VNET，可以定义一个或多个子网。

子网可用于：

- 在特定 VNet 上限制 IP 地址
- 在第 3 层区域中的 VNet 上分配路由/网关
- 在第 3 层区域中的 VNet 上启用 SNAT
- 通过 IPAM 插件在虚拟来宾（VM 或 CT）上自动分配 IP
- 通过 DNS 插件进行 DNS 注册

如果 IPAM 服务器与子网区域相关联，则子网前缀将自动在 IPAM 中注册。

子网属性包括：

- 子网:

  一个 cidr 网络地址。例如：10.0.0.0/8

- 网关

  网络默认网关的地址。在第 3 层区域（Simple/evpn 插件）上，它将部署到 vnet。

- Snat

  可选配置，为此子网的第 3 层区域（简单/evpn 插件）启用 Snat。子网源 IP 将 NAT 到服务器传出接口/ip。在 evpn 区域上，它仅在 evpn 网关节点上完成。

- DNS 域前缀

  可选配置。为域添加前缀，如.prefix.

12.5 12.5 控制器

某些 zone 类型需要外部控制器来管理 vNet 控制平面。目前，这只是 bgp-evpn zone 插件所需的。

12.5.1 12.5.1. EVPN 控制器

对于 BGP-EVPN，我们需要一个控制器来管理控制平面。当前支持的软件控制器是“frr”路由器。您可能需要在要部署 EVPN 区域的每个节点上安装它。

```
apt install frr frr-pythontools
```

配置选项：

- asn

  唯一的 BGP ASN 编号。强烈推荐使用私有 ASN 号 (64512-65534,4200000000-4294967294)，否则最终可能会被全局路由错误破坏或被破坏。

- peers

  要通信的所有节点的 IP 列表 (也可以是外部节点或路由反射器服务器)

12.5.2 12.5.2. BGP Controller

bgp 控制器不由区域直接使用。您可以使用它来配置 frr 以管理 bgp 对等体

对于 BGP-evpn，它可用于按节点定义不同的 ASN，因此执行 EBGP。

配置选项：

- 节点

  配置 BGP 所在节点

- asn

  唯一的 BGP ASN 编号。强烈推荐使用私有 ASN 号 (64512-65534,4200000000-4294967294)，否则最终可能会被全局路由错误破坏或被破坏。

- peers

  要与基础 BGP 网络通信的对等方的 IP 列表。

- ebgp

  如果对等方的远程 AS 不同，则启用 EBGP。

- 回环接口

  如果要使用环回或虚拟接口作为 evpn 网络的源。（适用于多路径）

- ebgp-mutltihop

  如果对等体未直接连接或使用环回，则可以增加跃点数以到达它们。

12.6 12.6 IPAM

IPAM(ip 地址管理) 工具用于在网络上管理和分配设备的 ip 地址。例如，它可用于在创建 vm/ct 时查找可用 IP 地址（尚未实现）。

IPAM 可与 1 个或多个区域相关联，以便为此区域中定义的所有子网提供 IP 地址。

12.6.1 12.6.1. Proxmox VE IPAM 插件

这是 proxmox 群集的默认内部 IPAM，如果您没有外部 IPAM 软件

12.6.2 12.6.2. phpIPAM plugin

https://phpipam.net/

您需要在 phpipam 中创建一个应用程序，并添加一个具有管理员权限的 api 令牌

phpIPAM 有如下选项：

- url

  REST-API 地址: http://phpipam.domain.com/api//

- 令牌

  API 访问 token

- 区段

  一个整数 ID。节是 phpIPAM 中的子网组。默认安装对客户使用 sectionid=1。

12.6.3 12.6.3. netbox IPAM 插件

NetBox 是一个 IP 地址管理（IPAM）和数据中心基础设施管理（DCIM）工具，有关详细信息，请参阅源代码存储库:https://github.com/netbox-community/netbox

您需要在 netbox 中创建一个 API 令牌.

> https://netbox.readthedocs.io/en/stable/api/authentication

NetBox 有如下选项：

- url

  REST-API 地址: http://yournetbox.domain.com/api

- 令牌

  API 访问 token

12.7 12.7 DNS

Proxmox VE SDN 中的 DNS 插件用于连接一个 DNS API 服务器，以注册虚拟机的主机名和 IP 地址。DNS 配置与一个或多个区域相关联，可为某个区域，配置的所有子网 IP 提供 DNS 注册。

12.7.1 12.7.1. PowerDNS 插件

https://doc.powerdns.com/authoritative/http-api/index.html

您需要在 PowerDNS 配置中启用 Web 服务器和 API：

```
api=yes
api-key=sui-ji-yi-ge-zi-fu-chuan
webserver=yes
webserver-port=8081
```

Powerdns 有如下属性:

- url

  REST-API 地址: http://yourpowerdnserver.domain.com:8081/api/v1/servers/localhost

- key

  API 访问密钥

- ttl

  默认的 ttl 记录值

12.8 12.8 示例

12.8.1 12.8.1. VLAN 设置示例

虽然我们在这里显示了普通的配置内容，但几乎所有内容都应该只使用 Web 界面进行配置。

Node1: /etc/network/interfaces

```
auto vmbr0
iface vmbr0 inet manual
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        bridge-vlan-aware yes
        bridge-vids 2-4094
```

(续下页)

(接上页)

```
#management ip on vlan100
auto vmbr0.100
        iface vmbr0.100 inet static
        address 192.168.0.1/24

source /etc/network/interfaces.d/*
```

Node2: /etc/network/interfaces

```


auto vmbr0
iface vmbr0 inet manual
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        bridge-vlan-aware yes
        bridge-vids 2-4094

#management ip on vlan100
auto vmbr0.100
iface vmbr0.100 inet static
        address 192.168.0.2/24

source /etc/network/interfaces.d/*
```

创建名为‘myvlanzone’的 VLAN zone：

```
id: myvlanzone
bridge: vmbr0
```

使用‘vlan-id’‘10’创建名为‘myvnet1’的 VNet，并将之前创建的‘myvlanzone’作为其 zone。

```
id: myvnet1
zone: myvlanzone
tag: 10
```

通过主 SDN 面板应用配置，以便在每个节点上本地创建 VNET。在 node1 上创建基于 Debian 的虚拟机 (VM1)，并在“myvnet1”上创建 vNIC。

为此 VM 使用以下网络配置：

```
auto eth0
iface eth0 inet static
        address 10.0.3.100/24
```

在 node2 上创建第二个虚拟机 (Vm2)，并在与 vm1 相同的 vNet‘myvnet1’上使用 vNIC。

对此 VM 使用以下网络配置：

```
auto eth0
iface eth0 inet static
        address 10.0.3.101/24
```

然后，您应该能够通过该网络在两个 VM 之间执行 ping 操作。

12.8.2 12.8.2 QinQ 配置示例

虽然我们在这里显示了普通的配置内容，但几乎所有内容都应该只使用 Web 界面进行配置。

Node1: /etc/network/interfaces

```
auto vmbr0
iface vmbr0 inet manual
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        bridge-vlan-aware yes
        bridge-vids 2-4094

#management ip on vlan100
auto vmbr0.100
iface vmbr0.100 inet static
        address 192.168.0.1/24

source /etc/network/interfaces.d/*
```

Node2: /etc/network/interfaces

```
auto vmbr0
iface vmbr0 inet manual
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        bridge-vlan-aware yes
        bridge-vids 2-4094

#management ip on vlan100
auto vmbr0.100
iface vmbr0.100 inet static
        address 192.168.0.2/24
```

(续下页)

(接上页)

```
source /etc/network/interfaces.d/*
```

使用服务 VLAN 20 创建名为‘qinqzone1’的 QinQ zone

```
id: qinqzone1
bridge: vmbr0
service vlan: 20
```

使用服务 VLAN 30 创建另一个名为‘qinqzone2’的 QinQ 区域

```
id: qinqzone2
bridge: vmbr0
service vlan: 30
```

在之前创建的‘qinqzone1’分区上创建名为‘myvnet1’、客户 VLAN-id 为 100 的 vNet。

```
id: myvnet1
zone: qinqzone1
tag: 100
```

在之前创建的‘qinqzone2’分区上创建一个客户 VLAN-id 为 100 的‘myvnet2’。

```
id: myvnet2
zone: qinqzone2
tag: 100
```

应用主 SDN Web 界面面板上的配置，在每个节点上本地创建 VNET。

在 node1 上创建基于 Debian 的虚拟机 (VM1)，并在‘myvnet1’上创建 vNIC。

为此 VM 使用以下网络配置：

```
auto eth0
iface eth0 inet static
address 10.0.3.100/24
```

在 node2 上创建第二个虚拟机 (Vm2)，并在与 vm1 相同的 VNet‘myvnet1’上安装 vNIC。为此 VM 使用以下网络配置：

```
auto eth0
iface eth0 inet static
address 10.0.3.101/24
```

在节点 1 上创建第三个虚拟机 (Vm3)，并在另一个 VNet‘myvnet2’上创建一个 vNIC。

为此 VM 使用以下网络配置：

```
auto eth0
iface eth0 inet static
address 10.0.3.102/24
```

在节点 2 上创建另一个虚拟机 (Vm4)，使 vNIC 位于与 vm3 相同的 vNet‘myvnet2’上。

为此 VM 使用以下网络配置:

```
auto eth0
iface eth0 inet static
address 10.0.3.103/24
```

然后，您应该能够在虚拟机 vm1 和 vm2 之间执行 ping 操作，也可以在 vm3 和 vm4 之间执行 ping 操作。但是，VM vm1 或 vm2 都不能 ping 通 VM vm3 或 vm4，因为它们位于具有不同服务 VLAN 的不同区域。

12.8.3 12.8.3. VXLAN 配置示例

虽然我们在这里显示了普通的配置内容，但几乎所有内容都应该只使用 Web 界面进行配置。

node1: /etc/network/interfaces

```
auto vmbr0
iface vmbr0 inet static
        address 192.168.0.1/24
        gateway 192.168.0.254
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        mtu 1500

source /etc/network/interfaces.d/*
```

node2: /etc/network/interfaces

```
auto vmbr0
iface vmbr0 inet static
        address 192.168.0.2/24
        gateway 192.168.0.254
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        mtu 1500

source /etc/network/interfaces.d/*
```

node3: /etc/network/interfaces

```
auto vmbr0
iface vmbr0 inet static
        address 192.168.0.3/24
        gateway 192.168.0.254
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        mtu 1500

source /etc/network/interfaces.d/*
```

创建一个名为‘myvxlanzone’的 VXLAN 区域，使用较低的 MTU 确保可以容纳额外的 50 个字节的 VXLAN 报头。将节点之前配置的所有 IP 添加为对等地址列表。

```
id: myvxlanzone
peers address list: 192.168.0.1,192.168.0.2,192.168.0.3
mtu: 1450
```

使用之前创建的 VXLAN 区域‘myvxlanzone’创建名为‘myvnet1’的 vNet。

```
id: myvnet1
zone: myvxlanzone
tag: 100000
```

应用主 SDN Web 界面面板上的配置，在每个节点上本地创建 VNET。

在 node1 上创建基于 Debian 的虚拟机 (VM1)，并在‘myvnet1’上创建 vNIC。

对此虚拟机使用以下网络配置，请注意此处较低的 MTU。

```
auto eth0
iface eth0 inet static
address 10.0.3.100/24
mtu 1450
```

在 node3 上创建第二个虚拟机 (Vm2)，并在与 vm1 相同的 vNet‘myvnet1’上安装 vNIC。为此 VM 使用以下网络配置:

```
auto eth0
iface eth0 inet static
address 10.0.3.101/24
mtu 1450
```

然后，您应该能够在 vm1 和 vm2 之间执行 ping 操作。

12.8.4 12.8.4 EVPN 设置示例

node1: /etc/network/interfaces

```
auto vmbr0
iface vmbr0 inet static
        address 192.168.0.1/24
        gateway 192.168.0.254
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        mtu 1500

source /etc/network/interfaces.d/*
```

node2: /etc/network/interfaces

```
auto vmbr0
iface vmbr0 inet static
        address 192.168.0.2/24
        gateway 192.168.0.254
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        mtu 1500

source /etc/network/interfaces.d/*
```

node3: /etc/network/interfaces

```
auto vmbr0
iface vmbr0 inet static
        address 192.168.0.3/24
        gateway 192.168.0.254
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
        mtu 1500

source /etc/network/interfaces.d/*
```

创建一个 EVPN 控制器，使用私有 ASN 号和以上节点地址作为对等设备。将上面的节点地址加入池。

```
id: myevpnctl
asn: 65000
```

(续下页)

(接上页)

```
peers: 192.168.0.1,192.168.0.2,192.168.0.3
```

使用之前创建的 EVPN-Controller 创建名为‘myevpnzone’的 EVPN 区域。使用 node1 和 node2 作为退出节点

```
id: myevpnzone
vrf vxlan tag: 10000
controller: myevpnctl
mtu: 1450
vnet mac address: 32:F4:05:FE:6C:0A
exitnodes: node1,node2
```

创建一个名为’myvnet1’的 VNET，使用 EVPN 区域’myevpnzone’

```
id: myvnet1
zone: myevpnzone
tag: 11000
```

创建一个子网 10.0.1.0/24，并将 10.0.1.1 作为 vnet1 上的网关

```
subnet: 10.0.1.0/24
gateway: 10.0.1.1
```

创建第二个名为’mynet2’的子网，同样使用 EVPN 区域’myevpnzone’，只是 ipv4 网络不一样。

```
id: myvnet2
zone: myevpnzone
tag: 12000
```

创建一个不同的子网 10.0.2.0/24，并将 10.0.2.1 作为 vnet1 上的网关

```
subnet: 10.0.2.0/24
gateway: 10.0.2.1
```

应用主 SDN Web 界面面板上的配置，在每个节点上本地创建 VNET 并生成 FRR 配置。

在 node1 上创建基于 Debian 的虚拟机 (VM1)，并在‘myvnet1’上创建 vNIC。为此 VM 使用以下网络配置：

```
auto eth0
iface eth0 inet static
        address 10.0.1.100/24
        gateway 10.0.1.1 #this is the ip of the vnet1
        mtu 1450
```

在 node2 上创建第二个虚拟机 (Vm2)，在另一个 vNet‘myvnet2’上创建一个 vNIC。

为此 VM 使用以下网络配置：

```
auto eth0
iface eth0 inet static
        address 10.0.2.100/24
        gateway 10.0.2.1 #this is the ip of the vnet2
        mtu 1450
```

然后，您应该能够从 VM1 ping 通 vm2，并从 vm2 ping 通 VM1。

如果您从非网关节点 3 上的 vm2 ping 外部 IP，则数据包将到达配置的 myvnet2 网关，然后被路由到网关节点 (节点 1 或节点 2)，并从那里通过节点 1 或节点 2 上配置的默认网关离开这些节点。

- 注意

  当然，您需要将 10.0.1.0/24 和 10.0.2.0/24 网络的反向路由添加到您的外部网关上的 node1、node2，这样公网才能回复。

如果您配置了外部 BGP 路由器，BGP-EVPN 路由 (本例中为 10.0.1.0/24 和 10.0.2.0/24) 将被动态通告。

12.9 12.9. Notes

12.9.1 12.9.1. VXLAN IPSEC 加密

如果您需要在 VXLAN 之上添加加密，则可以通过 `strongswan` 方式使用 IPSEC 进行加密。您需要将 MTU 减少 60 个字节（IPv4）或 80 个字节（IPv6）来处理加密。

因此，对于默认的实际 1500 MTU，您需要使用 MTU 1370（1370 + 80 （IPSEC）+ 50 （VXLAN）== 1500）。

安装 strongswan

```
apt install strongswan
```

在‘/etc/ipsec.conf’中添加配置。我们只需要加密来自 VXLAN UDP 4789 端口的流量。

```
conn %default
    ike=aes256-sha1-modp1024!  # the fastest, but reasonably secure cipher on modern␣
↪HW
    esp=aes256-sha1!
    leftfirewall=yes           # this is necessary when using Proxmox VE firewall␣
↪rules

conn output
    rightsubnet=%dynamic[udp/4789]
    right=%any
    type=transport
```

(续下页)

(接上页)

```
    authby=psk
    auto=route

conn input
    leftsubnet=%dynamic[udp/4789]
    type=transport
    authby=psk
    auto=route
```

然后生成一个预共享密钥

```
openssl rand -base64 128
```

并复制密钥到’/etc/ipsec.secrets’中，使文件内容看起来像这样：

```
: PSK <generatedbase64key>
```

您需要在其他节点上复制 PSK 和配置。

CHAPTER 13

第十三章 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。

13.1 13.1 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。

13.2 13.2 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]”。

13.2.1 13.2.1 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。
```

(续下页)

(接上页)

```
[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。

13.2.2 13.2.2 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>
```

(续下页)

(接上页)

```
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。
```

13.2.3 13.2.3 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

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。
```

(续下页)

(接上页)

```
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。

13.3 13.3 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
↪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+:\
```

(续下页)

(接上页)

```
↪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
↪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
```

13.4 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
```

(续下页)

(接上页)

```
[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
```

13.4 13.5. 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

13.4.1 13.5.1 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

13.5 /etc/pve/firewall/cluster.fw

[ALIASES] local_network 1.2.3.4 # use the single ip address

13.6 13.6 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
```

13.6.1 13.6.1 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
```

13.6.2 13.6.2 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
```

13.6.3 13.6.3 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
```

13.7 13.7 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
```

13.8 13.8 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：

13.8.1 13.8.1 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。具体处理结果由防火墙 → 选项中的选项决定，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。

13.8.2 13.8.2 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。

13.9 13.9 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：

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.

13.9.1 13.9.1 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 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
```

13.10 13.10 提示和窍门

13.10.1 13.10.1 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）。

13.10.2 13.10.2 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
```

13.11 13.11 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）。

13.12 13.12 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

CHAPTER 14

第十四章用户管理

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。

14.1 14.1 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

14.1.1 14.1.1 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。

14.2 14.2 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。

14.3 14.3 API Tokens

API Tokens 允许另一个系统、软件或 API 客户端对 REST API 的大部分进行无状态访问。可以为单个用户生成 Tokens，并且可以为其赋予单独的权限和到期日期，以限制访问的范围和持续时间。如果 API Tokens 受到危害，则可以在不禁用用户本身的情况下撤销 Tokens。

API Tokens 有两种基本类型：

- 权限分离：Token 需要通过 ACL 显式访问，其有效权限由用户权限和 Token 权限相交计算得出。
- 完全权限：令牌权限与关联用户的权限相同。

  警告: 在生成令牌时，令牌值仅显示/返回一次。以后不能通过 API 再次检索！

要使用 API Tokens，请在发出 API 请求时将 HTTP 头 Authorization 设置为 PVEAPIToken=User@Realm!TOKENID=UUID 形式的显示值，或参阅 API 客户端文档。

14.4 14.4. 资源池

资源池是一组虚拟机、容器和存储设备。在某些用户应该对一组特定资源具有受控访问权限的情况下，它对于权限处理非常有用，因为它允许将单个权限应用于一组元素，而不必基于每个资源对其进行管理。资源池通常与组串联使用，以便组的成员对一组计算机和存储具有权限。

14.5 14.5 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

Linux PAM 是用于系统范围用户身份验证的框架。这些用户是使用 adduser 等命令在主机系统上创建的。如果 Proxmox VE 主机系统上存在 PAM 用户，则可以将相应的条目添加到 Proxmox VE，以允许这些用户通过其系统用户名和密码登录。

- 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

  LDAP（轻量级目录访问协议）是一种开放的跨平台协议，用于使用目录服务进行身份验证。OpenLDAP 是 LDAP 协议的流行开源实现。

- 微软活动目录 (AD)

  Microsoft Active Directory （AD）是 Windows 域网络的目录服务，支持将其作为 Proxmox VE 的身份验证领域。它支持 LDAP 作为身份验证协议。

- OpenID Connect

  OpenID Connect 是作为 OATH 2.0 协议之上的身份层实现的。它允许客户端根据外部授权服务器执行的身份验证来验证用户的身份。

14.5.1 14.5.1. Linux PAM 标准认证

由于 Linux PAM 对应于主机系统用户，因此允许用户登录的每个节点上都必须存在一个系统用户。用户使用其常用系统密码进行身份验证。

默认情况下，此领域是添加的，无法删除。

在可配置性方面，管理员可以选择要求对来自领域登录名进行双重身份验证，并将领域设置为默认身份验证领域。

14.5.2 14.5.2. Proxmox VE 认证服务器

该领域是默认创建的，与 Linux PAM 一样，唯一可用的配置项目是能够要求该领域的用户进行双重身份验证，并将其设置为用于登录的默认领域。

与其他 Proxmox VE 领域类型不同，用户完全通过 Proxmox VE 创建和身份验证，而不是针对其他系统进行身份验证。因此，您需要在创建时为此类用户设置密码。

14.5.3 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。

可以配置服务器和可选的回退服务器，并且可以通过 SSL 对连接进行加密。此外，可以为目录和组配置过滤器。过滤器允许您进一步限制领域的范围。

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/.pw（例如/etc/pve/priv/ldap/my-ldap.pw），其中仅有一行明文口令信息。

要验证证书，您需要设置 capath。您可以将其直接设置为 LDAP 服务器的 CA 证书，也可以设置为包含所有受信任 CA 证书（/etc/ssl/certs）的系统路径。此外，您需要设置验证选项，也可以通过 Web 界面完成。

LDAP 领域主要配置如下:

- 领域（Realm）：Proxmox VE 用户的领域标识符
- 基本域名（base_dn）：用户所在的目录
- 用户属性名称（user_attr）：包含用户将要登录的用户名的 LDAP 属性
- 服务器（server1）：LDAP 目录所在的服务器
- 后备服务器（server2）：可选的后备服务器地址，以防无法访问主服务器
- 端口（port）：LDAP 服务器监听端口

注意：为了允许特定用户使用 LDAP 服务器进行身份验证，您还必须在 Proxmox VE 服务器将他们添加为该领域的用户。这可以通过同步自动执行。

14.5.4 14.5.4. 微软活动目录 (AD)

要将 Microsoft AD 设置为 Proxmox VE 领域，需要指定服务器地址和身份验证域。Active Directory 支持大多数与 LDAP 相同的属性，例如可选的后备服务器、端口和 SSL 加密。此外，配置后，用户可以通过同步操作自动添加到 Proxmox VE 中。

与 LDAP 一样，如果 Proxmox VE 在绑定到 AD 服务器之前需要进行身份验证，则必须配置绑定用户（bind_dn）属性。默认情况下，Microsoft AD 通常需要此属性。

微软活动目录的主要配置是：

- 领域（Realm）：Proxmox VE 用户的领域标识符
- 域名（domain）：AD 服务器的域名
- 服务器（server1）：LDAP 目录所在的服务器
- 后备服务器（server2）：可选的后备服务器地址，以防无法访问主服务器
- 端口（port）：LDAP 服务器监听端口

14.5.5 14.5.5 同步基于 LDAP 的领域

可以自动同步基于 LDAP 领域的用户和组（LDAP 和 Microsoft Active Directory），而不必手动将它们添加到 Proxmox VE。

您可以从 Web 界面身份验证面板的添加/编辑窗口或通过 `pveum realm add/modfy` 改命令访问同步选项。然后，您可以从 GUI 的身份验证面板或使用以下命令执行同步操作：

```
pveum realm sync <realm>
```

用户和组会同步到集群范围的配置文件`/etc/pve/user.cfg`。

同步配置

同步基于 LDAP 领域的配置选项可以在添加/编辑窗口的同步选项选项卡中找到。

主要配置选项如下：

- 绑定用户（bind_dn）：

  指定用于查询用户和组的 LDAP 帐户。此帐户需要能够访问所有所需的条目。如果设置了，搜索将以账户进行；否则，搜索将匿名进行。用户必须是完整的 LDAP 格式的名称（DN），例如 cn=admin，dc=example，dc=com

- 群组名属性（group_name_attr）：

  组名称，只有符合 user.cfg 的通常字符限制的条目才会同步。组与附加到名称的-$realm 域同步，以避免命名冲突。请确保同步不会覆盖手动创建的组

- 用户类（user_classes）：根据 LDAP 用户类筛选

- 群组对象类（group_classes）：根据 LDAP 群组类筛选
- 用户筛选（filter）：用于针对特定用户的更多筛选器选项。
- 群组筛选器（group_filter）：有关针对特定组的更多筛选器选项。

筛选器允许您创建一组其他匹配条件，以缩小同步范围。有关可用 LDAP 筛选器类型及其用法的信息，请访问 ldap.com。

同步选项

- 范围：要同步的内容的范围。它可以是用户、组或一起。
- 启用新用户：如果设置，则新同步的用户将启用并可以登录。默认值为 true。
- 完整：如果设置，同步将使用 LDAP 目录作为真值来源，覆盖在 user.cfg 中手动设置的信息，并在 Proxmox VE 上删除不在 LDAP 目录中的用户和组。如果未设置，则只会将新数据写入配置，不会删除陈旧用户
- 清除：如果设置，同步会在删除用户和组时删除所有相应的 ACL。仅在勾选了’完整’时生效。
- 预览：不会将任何数据写入配置。如果您想要查看哪些用户和组将同步到 user.cfg，这将非常有用。在单击界面中的预览时设置。

14.5.6 14.5.6. OpenID Connect

主要的 OpenID Connect 配置选项包括：

- 发行人 URL：

  这是授权服务器的 URL。Proxmox 使用 OpenID Connect Discovery 协议来自动配置更多详细信息。

  虽然可以使用未加密的 http:// URL，但我们强烈建议使用加密 https://URL 连接。

- 领域: 取一个领域名
- 客户端 ID：OpenID 客户端 ID
- 客户端密钥：OpenID 客户端密钥
- 自动创建用户：如果用户不存在，则自动创建用户。虽然身份验证是在 OpenID 服务器上完成的，但所有用户仍然需要 Proxmox VE 用户配置中的条目。您可以手动添加它们，也可以使用自动创建选项自动添加新用户。
- 用户名声明：用于生成唯一用户名（主题用户名或电子邮件）的 OpenID 声明。

用户名映射

OpenID Connect 规范定义了一个唯一属性（`claim`）叫做 `subject`。默认情况下，我们使用此属性的值来生成 Proxmox VE 用户名，只需添加 @ 和领域名称：`${subject}@${realm}`

不幸的是，大多数 OpenID 服务器使用随机字符串作为 `subject`，如 DGH76OKH34BNG3245SB，因此典型的用户名看起来像 DGH76OKH34BNG3245SB@yourrealm。虽然不会重复，但人类很难记住这些随机字符串，因此完全不可能将真实用户与此相关联。

username-claim 设置允许您使用其他属性进行用户名映射。如果 OpenID Connect 服务器提供该属性并保证其唯一性，则最好将其设置为 username。

另外一个选项就是使用 email 属性，这也能产生友好的用户名。同样，仅当服务器保证此属性的唯一时，才使用此设置。

示例

下面是使用 Google 创建 OpenID 域的示例。您需要将–client-id 和–client-key 替换为 Google OpenID 设置中的值。

```
pveum realm add myrealm1 --type openid --issuer-url  https://accounts.google.com --
↪client-id XXXX --client-key YYYY --username-claim email
```

上述命令使用--username-claim email，这样 Proxmox VE 上的用户名就会像这样 example.user@google.com@myrealm1

Keycloak (https://www.keycloak.org/) 是一个流行的开源身份和访问管理工具，它支持 OpenID Connect。在以下示例中，您需要将–issuer-url 和–client-id 替换为您的信息

```
pveum realm add myrealm2 --type openid --issuer-url https://your.server:8080/
auth/realms/your-realm --client-id XXX --username-claim username
```

使用–username-claim username 允许 Proxmox VE 使用 username 作为用户名，像这样 example.user@myrealm2.

注意！您需要确保不允许用户自己编辑用户名设置（在密钥保护服务器上）。

14.6 14.6. 二次验证

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。

14.6.1 14.6.1 可用的二次验证

您可以同时设置多种二次验证，以避免丢失智能手机或安全密钥将而无法登录的情况。

除了领域强制的 TOTP 和 YubiKey OTP 之外，还提供了以下双因素身份验证方法：

- 用户配置的 TOTP（基于时间的一次性密码）。从共享密钥和当前时间派生的短代码，它每 30 秒更改一次。

- WebAuthn（Web Authentication）。身份验证的一般标准。它由各种安全设备实现，例如来自计算机或智能手机的硬件密钥或受信任的平台模块（TPM）。
- 还原密钥。应打印出来并锁定在安全位置或以数字方式保存在电子保险库中的密钥列表。每个密钥只能使用一次。这些非常适合确保您不会被锁定，即使您所有其他第二因素都丢失或损坏。

在支持 WebAuthn 之前，U2F 可以由用户设置。现有的 U2F 验证仍然可以使用，但建议在服务器上配置后切换到 WebAuthn。

14.6.2 14.6.2. 领域强制双因素身份验证

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。

14.6.3 14.6.3 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*密钥*栏是 key 信息，可以通过*随机化*按钮自动生成新的 key。*发行者名称*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*发行者名称*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*验证码*栏输入当前 OTP 值，最后点击*添加*按钮。

14.6.4 14.6.4. TOTP

无需设置服务器。只需在智能手机上安装 TOTP 应用程序（例如，FreeOTP），然后使用 Proxmox 备份服务器 Web 界面添加 TOTP 因子。

14.6.5 14.6.5. WebAuthn

要启动 WebAuthn, 需要做下面 2 个步骤：

- 受信任的 HTTPS 证书（例如，通过使用 Let’s Encrypt）。如果环境中没有可信任的根证书，某些浏览器可能会警告或拒绝 WebAuthn 操作（如果它不受信任）。
- 在 GUI 网页上配置 WebAuthn （数据中心-> 选项-> WebAuthn Settings），这可以在大多数设置中自动完成配置。

满足这两个要求后，可以在“数据中心”下的“二次验证”面板中添加 WebAuthn （配置 → “权限” → “二次验证”）

14.6.6 14.6.6 还原密钥

还原密钥代码不需要任何准备; 您只需在“数据中心”下的“二次验证”面板中创建一组还原密钥，（“权限” → “二次验证”）。

提示: 在任何时候，每个用户只能有一组一次性还原密钥。

14.6.7 14.6.7. WebAuthn

要允许用户使用 WebAuthn 身份验证，必须使用具有有效 SSL 证书的有效域名，否则某些浏览器可能会发出警告或拒绝身份验证。

提示: 修改 WebAuthn 配置可能导致现有的 WebAuthn 注册不可用！

这是通过 /etc/pve/datacenter.cfg 完成配置的。例如：

```
webauthn:
rp=mypve.example.com,origin=https://mypve.example.com:8006,id=mypve.example.com
```

14.6.8 14.6.8 服务器端 U2F 配置

提示：建议改用 WebAuthn。

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。

14.6.9 14.6.9 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。

14.7 14.7 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、组和令牌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）、（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。

14.7.1 14.7.1. 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。

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权限 → 角色选项卡，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"
```

14.7.2 14.7.2. 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
- Pool.Audit: 查看资源池
- 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

14.7.3 14.7.3. 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。

此外，当一个用户不拥有某个路径的权限时，其特权分离的令牌也不会拥有此路径的权限。

14.7.4 14.7.4. 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。

14.7.5 14.7.5 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/, 拥有 Realm.AllocateUser 访问权。其中是用户通过 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 @ 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。

14.8 14.8 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"
```

14.9 14.9 实际应用示例

14.9.1 14.8.1 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
```

14.9.2 14.8.2 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
```

14.9.3 14.8.3 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。

14.9.4 14.8.4 只用于监控的 API 权限

给定在所有虚拟机上具有 PVEVMAdmin 角色的用户 Joe@pve：

```
pveum aclmod /vms -user joe@pve -role PVEVMAdmin
```

添加具有单独权限的新 API token，该令牌仅允许查看 VM 信息 (例如，用于监视目的)：

```
pveum user token add joe@pve monitoring -privsep 1
pveum aclmod /vms -token ' joe@pve!monitoring' -role PVEAuditor
```

验证用户和 token 的权限：

```
pveum user permissions joe@pve
pveum user token permissions joe@pve monitoring
```

14.9.5 14.8.5 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。

CHAPTER 15

第十五章 HA 高可用

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%。

15.1 15.1. 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

15.2 15.2 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。

15.3 15.3 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。

15.4 15.4 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。

15.4.1 15.4.1 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）。

- recovery

  等待服务恢复。HA 管理器将搜寻可用的节点。此搜索不仅取决于在线的节点和仲裁节点，还要看此服务是否为组成员，以及这个组是否有限制。一旦新的可用节点被发现，服务将移动到此处，并开始置于停止状态。如果被配置为运行在新节点，则会执行这个操作。

- 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。

15.4.2 15.4.2 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：最大并发调整提示

- 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//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。

15.4.3 15.4.3 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

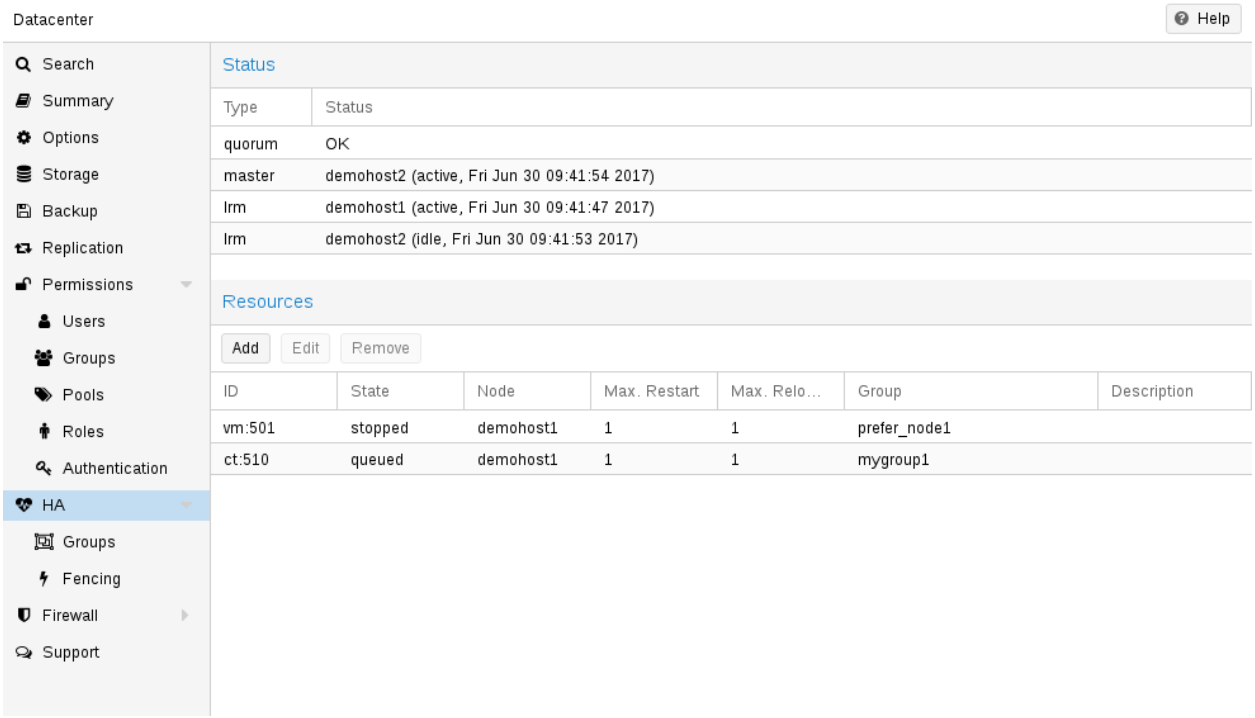
  表示 CRM 失去了独占锁，一般意味着有错误发生，并且节点失去了多数票。

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。

15.5 HA 模拟器



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。

15.6 15.6. 配置

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。

15.6.1 15.6.1. 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
```

15.6.2 15.6.2. 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。

15.7 15.7. 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。

15.7.1 15.7.1 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。

15.7.2 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。

15.7.2 15.7.3 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。

15.8 15.8 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。

15.9 15.9 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。

15.10 15.10 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。

15.11 15.11 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。使用 HA 堆栈时也是如此。可以配置 HA 堆栈在关闭期间的行为。

15.11.1 15.11.1 关闭策略

在下面，您可以找到有关节点关闭的不同 HA 策略的说明。由于向后兼容，当前条件为默认设置。一些用户可能会发现迁移的行为更符合预期。

- 迁移

  一旦本地资源管理器 (LRM) 收到关闭请求并且启用了此策略，它会将其自身标记为对当前 HA 管理器不可用。这将触发当前位于此节点上的所有 HA 服务的迁移。在所有正在运行的服务移走之前，LRM 将尝试延迟关闭过程。但是，这需要将正在运行的服务迁移到另一个节点。换句话说，服务不能本地绑定，例如通过使用硬件通道。由于如果没有可用的组成员，则非组成员节点被视为可运行的目标，因此在仅选择了一些节点的情况下使用 HA 组时，仍可以使用此策略。但是，将组标记为受限会告诉 HA 管理器服务不能在所选节点集之外运行，如果所有这些节点都不可用，则关闭将挂起，直到您手动干预。一旦关闭的节点重新联机，如果之前替换的服务没有在中间手动迁移，它们将被移回。

注意: 在关闭时的迁移过程中，监视程序仍处于活动状态。如果节点失去仲裁，它将被隔离，并且服务将恢复。

如果在当前正在维护的节点上启动 (先前停止的) 服务，则需要隔离该节点，以确保可以在另一个可用节点上移动和启动该服务。

- 故障切换

  此模式可确保停止所有服务，但如果当前节点未立即联机，则也会恢复这些服务。在集群规模上执行维护时可能会很有用，因为如果一次关闭多个节点，则可能无法实时迁移虚拟机，但您仍希望确保 HA 服务尽快恢复并重新启动。

- 冻结

  此模式可确保停止并冻结所有服务，以便在当前节点再次联机之前不会恢复这些服务。

- 有条件的

  有条件关闭策略自动检测是否请求关闭或重新启动，并相应地更改行为。

- 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。

- 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。

- 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。

CHAPTER 16

第十六章备份和恢复

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/pve/jobs.cfg‘文件中，该文件会被‘pvescheduler‘守护程序读取并执行。备份作业由日历事件来定义计划

16.1 16.1 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：

16.1.1 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。

16.1.2 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。

16.2 16.2. 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。

16.3 16.3. 备份文件压缩

可以使用以下算法之一压缩备份文件：lzo、gzip 或 zstd。

目前，ZStandard(zstd) 是这三种算法中最快的，多线程是 zstd 相对于 lzo 和 gzip 的另一个优势。Lzo 和 gzip 使用更广泛，而且默认已经安装相关软件.

您可以安装 Pigz 作为 gzip 的临时替代品，以通过多线程提供更好的性能。对于 Pigz&zstd，线程/核心的数量可以调整。请参阅下面的第 16.5 节配置选项。

备份文件名的扩展名通常可用于确定已使用哪种压缩算法创建备份。

如果备份文件名不是以上述文件扩展名之一结尾，那么它不是由 vzdump 压缩的。

16.4 16.4. 备份加密

针对于 Proxmox 备份服务器，可以选择设置备份加密，请参阅加密部分。

16.5 16.5. 备份保留

使用备份保留选项，可以灵活的指定要保留的备份。

备份保留有以下可用选项:

- 保留所有备份

  勾选即保留所有的历史备份，且其他选项将不可选。

- 保留上次

  保留最近 N 次备份。

- 保留每小时

  以小时为计算度量，保留最近 4 次的小时备份。

  若计划，1：00，2：00，12：00 备份，小时设置为 4，则保留当天，12：00，2：00，1：00。

- 保留每天

  以天为计算度量，保留最近 N 天的备份。一天若有多个备份，保留最新备份。

  在上面的基础上，若天设置为 3，则会保留，昨天 12：00，前天 12：00，大前天 12:00 数据

- 保留每周

  以周为计算度量，保留最近 N 周的备份。如果一周内有多个备份，则仅保留最新的备份

  在上面的基础上，若周设置为 3，则会保存，之前三周，每周日 12：00 的备份

- 保留每月

  以月为计算度量，保留最近 N 月的备份

  在上面的基础上，若月设置为 3，则会保留最近 3 个月 30 或 31 日的 12：00 的备份。

- 保留每年

  保留过去几年的备份，如果一年有多个备份，则会保留最新的备份。

备份保留功能会按照上面的顺序执行。每个选项只在其时间段涵盖备份，下一个选项不处理已覆盖的备份。它只会处理比较旧的备份。意味着，如果周和天，有重复的备份，那么周备份不会去处理天的备份，而是会处理下一份旧备份。

此功能，建议直接使用 PBS 的调度模拟器查看备份计划!

可以使用逗号作为分隔符，指定需要的保留选项，如:

```
# vzdump 777 --prune-backups keep-last=3,keep-daily=13,keep-yearly=9
```

这样可以直接将备份保留选项传递给 vzdump，但在 web 面板上配置基于存储的备份计划更为明智。

提示：

旧的 maxfiles 选项已弃用，应替换为 keep-last，或者如果 maxfiles 为 0，则表示无限保留，则应替换为 keep-all。

16.5.1 16.5.1. 调度模拟器

您可以使用 Proxmox 备份服务器文档的调度模拟器来查看具有不同备份时间表的不同保留选项的效果。

16.5.2 16.5.2. 保留设置示例

备份频率和旧备份保留应该根据数据更改频率，以及在特定工作负载下，旧备份的重要性来进行设置。当备份作为公司文档时，可能法律还会要求存档应保留多久。

例如，正在执行每日备份，需要保留 10 年，且备份与备份之间的保留间隔逐渐增大。

keep-last=3 - 即使只执行每日备份，管理员也可能希望在大升级之前或之后创建一个额外的备份。设置“保留最后”可确保这一点。

未设置每小时保持一次 - 对于每日备份，这与每日备份无关。您已经通过 Keep-Last 覆盖了额外的手动备份。

keep-daily=13 - 与至少涵盖一天的 keep-last 一起，这可确保您至少有两周的备份。

keep-weekly=8 - 确保您至少有两个完整月的每周备份。

keep-monthly=11 - 与之前的 keep 设置一起，这可确保您至少有一年的每月备份。

keep-year=9 - 这是针对长期存档的。当您使用以前的选项覆盖当年时，您将将其设置为 9，以便将其余选项设置为 9，从而为您提供至少 10 年的保险。

我们建议您使用比您的环境最低要求的保留期更高的保留期。如果您发现它不必要地高，则始终可以减少它，但是一旦删除备份，就无法重新创建备份。

16.6 16.6. 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

`qm restore` 虚拟机恢复命令

详情可查看相应的手册。

16.6.1 16.4.1 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
```

16.6.2 16.6.2. 实时还原

恢复一个大型的备份，会占用很大的时间，且在此期间无法访问虚拟机。存储在备份服务器上的备份可以通过实时还原选项来减少等待时间。

在 GUI 中勾选实时还原或者使用 `qm restore --live-restore` 会时虚拟机在还原开始后，立即启动。并在后台复制数据，优先处理 VM 正在主动访问的区块。

注意，这有两个警告：

- 在实时还原期间，VM 将以有限的磁盘读取速度运行，因为数据必须从备份服务器加载（加载后，它立即在目标存储上可用，因此访问数据两次只会在第一次产生性能损失)。写入速度基本上不受影响。
- 只要实时还原失败，VM 将处于未定义状态————也就是说，数据可能没有从备份中完整复制过来，并且很可能丢失在还原期间写入的任何数据。

这种还原模式对于大型 VM 特别有用，其中初始操作只需要少量数据，例如 Web 服务器————一旦操作系统和必要的服务启动，VM 即可运行，其他数据会在后台继续复制。

16.6.3 16.6.3. 文件还原

在 GUI 的备份选项中，点击文件还原按钮可直接浏览备份包中的文件。此功能只在 Proxmox 备份服务器后端有效。

对于容器，第一层是 pxar 压缩存档，可以自由打开和浏览。

对于虚拟机，第一层展现的是可打开的磁盘映像列表。在最基本的情况下，这将是一个名为 `part` 的条目，表示一个分区表，其中包含在磁盘映像上上找到的每个分区列表。注意，并非所有数据都可以访问（如，不支持文件系统，存储技术等）

可以点击下载按钮下载目录或者文件，随后会被压缩成一个 zip 文档。

若要对包含不安全数据的 VM 映像进行安全访问，将启动临时 VM（不作为来宾可见）。这并不意味着从此类存档下载的数据本质上是安全的，但它避免了将虚拟机管理程序系统暴露在危险之中。VM 将在超时后自行停止。从用户的角度来看，整个过程都是透明的。

16.7 16.7 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。

prune-backups: [keep-all=<1|0>] [,keep-daily=<N>] [,keep-hourly=<N>] [,keep-last=<N>] [,keep-monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>] (default = keep-all=1)

使用这些保留选项，而不是存储配置中的保留选项。

此功能，建议直接使用 PBS 的调度模拟器查看备份计划!

- keep-all=<boolean>

  勾选即保留所有的历史备份，且其他选项将不可选。

- keep-daily=<N>

  以天为计算度量，保留最近 N 天的备份。一天若有多个备份，保留最新备份。

  在上面的基础上，若天设置为 3，则会保留，昨天 12：00，前天 12：00，大前天 12:00 数据

- keep-hourly=<N>

  以小时为计算度量，保留最近 4 次的小时备份。

  若计划，1：00，2：00，12：00 备份，小时设置为 4，则保留当天，12：00，2：00，1：00。

- keep-last=<N>

  保留最近 N 次备份。

- keep-monthly=<N>

  以月为计算度量，保留最近 N 月的备份

  在上面的基础上，若月设置为 3，则会保留最近 3 个月 30 或 31 日的 12：00 的备份。

- keep-weekly=<N>

  以周为计算度量，保留最近 N 周的备份。如果一周内有多个备份，则仅保留最新的备份

  在上面的基础上，若周设置为 3，则会保存，之前三周，每周日 12：00 的备份

- keep-yearly=<N>

  保留过去几年的备份，如果一年有多个备份，则会保留最新的备份。

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。

zstd: <integer> (default = 1)

Zstd 线程数量。N=0 使用一半的可用内核，N>0 使用 N 作为线程计数。

vzdump.conf 配置示例

```
tmpdir: /mnt/fast_local_disk
storage: my_backup_storage
mode: snapshot
bwlimit: 10000
```

16.8 16.8. 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）。

16.9 16.9. 排除文件

注意: 此选项仅适用于容器备份。

vzdump 指令默认排除以下文件（可通过参数--stdexcludes 0 禁用默认排除）

```
/tmp/?*
/var/tmp/?*
/var/run/?*pid
```

可以手动指定排除路径，如

```
vzdump 777 --exclude-path /tmp/ --exclude-path '/var/foo*'
```

上面命令将排除/tmp/文件夹和在/var/下的任何 foo 开头的文件，如/var/foo, /var/foobar。

如果路径不以/开头，vzdump 会匹配任何的子目录，不会匹配根目录，如

```
vzdump 777 --exclude-path bar
```

上面命令命令将排除任何 bar 文件夹，如/var/bar,/bar,/var/foo/bar 等，但不会排除/bar2。

备份文件存储在备份包（/etc/vzdump/）中，并将正确还原。

16.10 16.10. 示例

简单备份 777 的客户机，没有快照，只是简单的备份磁盘文件和配置信息并储存到默认的备份文件夹中 (通常是 /var/lib/vz/dump/)。

```
vzdump 777
```

使用暂停模式备份，会创建一个快照，随后再备份。(停机时间最少)

```
# vzdump 777 --mode suspend
```

备份所有的客户机，并在备份完成之后，发送邮件给 root 和 admin

```
# vzdump --all --mode suspend --mailto root --mailto admin
```

使用快照模式备份，并指定备份文件夹。(没有停机时间)

```
# vzdump 777 --dumpdir /mnt/backup --mode snapshot
```

备份多个客户机，并且发送邮件。

```
# vzdump 101 102 103 --mailto root
```

备份除 101，102 以外的所有客户机。

```
# vzdump --mode suspend --exclude 101,102
```

还原一个容器 777 备份到新容器 600。

```
# pct restore 600 /mnt/backup/vzdump-lxc-777.tar
```

还原一个虚拟机。

```
# qmrestore /mnt/backup/vzdump-qemu-888.vma 601
```

利用 pipe 管道，克隆一个 101 容器，并还原成 300，顺便把 rootfs 设置成 4G。

```
# vzdump 101 --stdout | pct restore --rootfs 4 300 -
```

CHAPTER 17

第十七章重要服务

17.1 17.1 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。

17.2 17.2 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。

17.2.1 17.2.1 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。

17.2.2 17.2.2 监听的 IP 地址

pveproxy 和 spcieproxy 使用通配符监听所有地址，包括 ipv4 和 ipv6。

修改/etc/default/pveproxy 文件，可以指定监听的 ip，此 ip 必须在此节点上已配置好。

配置了 sysctl net.ipv6.bindv6only=1 将会使守护进程只监听 ipv6，通常会出现其他问题。建议删掉此配置，或者将 LISTEN_IP 设置成 0.0.0.0（只监听 ipv4）。

17.2.3 17.2.3 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

17.2.4 17.2.4 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。

17.2.5 17.2.5 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。

17.2.6 17.2.6 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
```

17.3 17.3 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。

17.4 17.4 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。

17.4.1 17.4.1 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。

17.4.2 17.5. pvescheduler Proxmox VE 调度守护进程

该守护进程负责根据计划启动作业，例如复制和备份。

对于备份任务，它从文件 /etc/pve/jobs.cfg 中获取配置信息。

CHAPTER 18

第十八章命令行工具

18.1 18.1. pvesubscription –订阅管理工具

Promxox VE 订阅管理工具。

18.2 18.2 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。

18.3 18.3 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。

18.3.1 18.3.1 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
```

CHAPTER 19

第十九章常见问题

 注意更新的常见问题将在追加在本节最后。

19.1 1.Proxmox VE 基于哪个发行版？

Proxmox VE 基于 Debian GNU/Linux。

19.2 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。

19.3 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。

19.4 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
```

19.5 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）

19.6 6. 支持的 AMD CPU

支持 AMD 虚拟化技术（AMD-V）的 64 位 CPU。

19.7 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。

19.8 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。

19.9 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。

19.10 10. 各版本的 Proxmox VE 最终支持期限是？

Proxmox VE 的支持周期和 debian 对oldstable的支持相同。Proxmox VE 使用滚动发布模型，并且始终建议使用最新的稳定版本。

19.11 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 6.x 升级到 7.0
- 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

19.12 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。

CHAPTER 20

第二十章书目

20.1 关于 Proxmox VE 的书籍

[Ahmed16] Wasim Ahmed. Mastering Proxmox - Third Edition. Packt Publishing, 2017. ISBN 978-1788397605

[Ahmed15] Wasim Ahmed. Proxmox Cookbook. Packt Publishing, 2015. ISBN 978-1783980901

[Cheng14] Simon M.C. Cheng. Proxmox High Availability. Packt Publishing, 2014. ISBN 978-1783980888

[Goldman16] Rik Goldman. Learning Proxmox VE. Packt Publishing, 2016. ISBN 978-1783981786

[Surber16]] Lee R. Surber. Virtualization Complete: Business Basic Edition. Linux Solutions (LRS-TEK), 2016. ASIN B01BBVQZT6

20.2 和技术相关的书籍

[Hertzog13] Raphaël Hertzog & Roland Mas. The Debian Administrator's Handbook: Debian Jessie from Discovery to Mastery, Freexian, 2013. ISBN 979-1091414050

[Bir96] Kenneth P. Birman. Building Secure and Reliable Network Applications. Manning Publications Co, 1996. ISBN 978-1884777295

[Walsh10] Norman Walsh. DocBook 5: The Definitive Guide. O'Reilly & Associates, 2010. ISBN 978-0596805029

[Richardson07] Leonard Richardson & Sam Ruby. RESTful Web Services. O'Reilly Media, 2007. ISBN 978-0596529260

[Singh15] Karan Singh. Learning Ceph. Packt Publishing, 2015. ISBN 978-1783985623

[Singh16] Karan Signh. Ceph Cookbook Packt Publishing, 2016. ISBN 978-1784393502

[Mauerer08] Wolfgang Mauerer. Professional Linux Kernel Architecture. John Wiley & Sons, 2008. ISBN 978-0470343432

[Loshin03] Pete Loshin, IPv6: Theory, Protocol, and Practice, 2nd Edition. Morgan Kaufmann, 2003. ISBN 978-1558608108

[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

[Kreibich10] Jay A. Kreibich. Using SQLite, O' Reilly and Associates, 2010. ISBN 978-0596521189

20.3 和主题相关的书籍

[Bessen09] James Bessen & Michael J. Meurer, Patent Failure: How Judges, Bureaucrats, and Lawyers Put Innovators at Risk. Princeton Univ Press, 2009. ISBN 978-0691143217