Pxvirt 管理指南

Release

July 17, 2026

Lierfang
www.lierfang.com

Copyright © 2026 梨儿方

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled ”GNU Free Documentation License”.

Contents

1 简介 1

1.1 集中管理 . . . 2
1.2 灵活存储 . . . 3
1.3 集成备份与还原 . . . 4
1.4 高可用集群 . . . 4
1.5 灵活网络 . . . 4
1.6 集成防火墙 . . . 4
1.7 超融合基础设施 . . . 4
1.7.1 使用 Proxmox VE 构建超融合基础设施（HCI）的优势 . . . 4
1.7.2 超融合基础设施：存储 . . . 5
1.8 为什么选择开源 . . . 5
1.9 使用 Proxmox VE 的优势 . . . 5
1.10 获取帮助 . . . 6
1.10.1 Pxvirt Wiki . . . 6
1.10.2 商业支持 . . . 6
1.10.3 缺陷跟踪器 . . . 6
1.11 项目历史 . . . 6
1.12 Pxvirt 项目历史 . . . 7
1.13 改进 Proxmox VE 文档 . . . 7
1.14 翻译 Proxmox VE . . . 7
1.14.1 使用 git 翻译 . . . 7
1.14.2 不使用 git 翻译 . . . 8
1.14.3 测试翻译 . . . 8
1.14.4 发送翻译 . . . 8
1.14.5 常用命令示例 . . . 8

2 安装 Proxmox VE **9**
2.1 系统要求 . . . 9
2.1.1 最低要求，仅用于评估 . . . 9
2.1.2 推荐系统要求 . . . 10
2.1.3 简单性能概览 . . . 10
2.1.4 常用命令示例 . . . 10
2.1.5 访问 Web 界面支持的浏览器 . . . 10
2.2 准备安装介质 . . . 11
2.2.1 将 USB 闪存盘准备为安装介质 . . . 11
2.2.2 GNU/Linux 操作说明 . . . 11
2.2.3 macOS 操作说明 . . . 12
2.2.4 Windows 操作说明 . . . 12
2.3 使用 Proxmox VE 安装程序 . . . 13
2.3.1 安装后访问管理界面 . . . 15
2.3.2 高级 LVM 配置选项 . . . 16
2.3.3 高级 ZFS 配置选项 . . . 17
2.3.4 高级 BTRFS 配置选项 . . . 17
2.3.5 ZFS 性能提示 . . . 17
2.3.6 添加 nomodeset 内核参数 . . . 18
2.4 无人值守安装 . . . 18
2.5 在 OpenEuler 上安装 Proxmox VE . . . 18
2.6 另请参阅 . . . 18

3 主机系统管理 **20**
3.1 软件包仓库 . . . 20
3.1.1 Proxmox VE 中的软件仓库 . . . 20
3.1.2 PXVIRT 仓库 . . . 21
3.1.3 Ceph 软件包仓库 . . . 21
3.1.4 软件包签名验证 . . . 21
3.2 系统软件更新 . . . 22
3.3 固件更新 . . . 22
3.3.1 持久化固件 . . . 22
3.3.2 运行时固件文件 . . . 23
3.3.3 CPU 微码更新 . . . 23
3.3.4 常用命令示例 . . . 25
3.4 网络配置 . . . 25

3.4.1 应用网络更改 . . . 26
3.4.2 命名约定 . . . 26
3.4.3 选择网络配置 . . . 29
3.4.4 使用 Bridge 的默认配置 . . . 30
3.4.5 路由配置 . . . 30
3.4.6 使用 iptables 的 Masquerading (NAT) . . . 31
3.4.7 Linux Bond . . . 32
3.4.8 VLAN 802.1Q . . . 33
3.4.9 在节点上禁用 IPv6 . . . 35
3.4.10在 Bridge 上禁用 MAC Learning . . . 36
3.5 时间同步 . . . 36
3.5.1 使用自定义 NTP 服务器 . . . 36
3.6 外部指标服务器 . . . 38
3.6.1 Graphite 服务器配置 . . . 38
3.6.2 InfluxDB 插件配置 . . . 38
3.6.3 常用命令示例 . . . 39
3.7 磁盘健康监控 . . . 39
3.7.1 常用命令示例 . . . 39
3.8 逻辑卷管理器 (LVM) . . . 40
3.8.1 硬件 . . . 40
3.8.2 引导加载器 . . . 40
3.8.3 创建卷组 . . . 41
3.8.4 为 /var/lib/vz 创建额外 LV . . . 41
3.8.5 调整 thin 池大小 . . . 42
3.8.6 创建 LVM-thin 池 . . . 42
3.8.7 常用命令示例 . . . 42
3.9 Linux 上的 ZFS . . . 42
3.9.1 硬件 . . . 43
3.9.2 作为根文件系统安装 . . . 43
3.9.3 ZFS RAID 级别考量 . . . 44
3.9.4 ZFS dRAID . . . 46
3.9.5 Bootloader . . . 46
3.9.6 ZFS 管理 . . . 47
3.9.7 配置电子邮件通知 . . . 50
3.9.8 限制 ZFS 内存使用 . . . 51

3.9.9 ZFS 上的 SWAP . . . 51
3.9.10加密的 ZFS Dataset . . . 52
3.9.11ZFS 中的压缩 . . . 53
3.9.12ZFS Special Device . . . 54
3.9.13ZFS Pool 特性 . . . 55
3.9.14常用命令示例 . . . 56
3.10BTRFS . . . 56
3.10.1作为根文件系统安装 . . . 56
3.10.2BTRFS 管理 . . . 57
3.11Proxmox 节点管理 . . . 59
3.11.1常用命令示例 . . . 59
3.11.2Wake-on-LAN . . . 60
3.11.3任务历史 . . . 60
3.11.4批量客户机电源管理 . . . 60
3.11.5首个客户机启动延迟 . . . 61
3.11.6批量客户机迁移 . . . 61
3.11.7Ballooning 的 RAM 使用率目标 . . . 61
3.12证书管理 . . . 61
3.12.1集群内部通信证书 . . . 61
3.12.2API 和 Web GUI 证书 . . . 62
3.12.3上传自定义证书 . . . 62
3.12.4通过 Let's Encrypt (ACME) 获取可信证书 . . . 62
3.12.5ACME HTTP 挑战插件 . . . 63
3.12.6ACME DNS API 挑战插件 . . . 64
3.12.7ACME 证书自动续期 . . . 65
3.12.8使用 pvenode 的 ACME 示例 . . . 65
3.13主机引导加载器 . . . 69
3.13.1安装程序使用的分区方案 . . . 69
3.13.2使用 proxmox-boot-tool 同步 ESP 内容 . . . 69
3.13.3确定正在使用的引导加载器 . . . 72
3.13.4GRUB . . . 72
3.13.5Systemd-boot . . . 72
3.13.6编辑内核命令行 . . . 73
3.13.7为下次启动覆盖内核版本 . . . 73
3.13.8Secure Boot . . . 74
3.14Kernel Samepage Merging (KSM) . . . 77
3.14.1KSM 的影响 . . . 77
3.14.2禁用 KSM . . . 77

4 图形用户界面 78
4.1 功能 . . . 78
4.2 登录 . . . 78
4.3 GUI 概览 . . . 79
4.3.1 页眉 . . . 79
4.3.2 我的设置 . . . 79
4.3.3 资源树 . . . 80
4.3.4 日志面板 . . . 80
4.4 内容面板 . . . 81
4.4.1 数据中心 . . . 81
4.4.2 节点 . . . 82
4.4.3 客户机 . . . 82
4.4.4 存储 . . . 83
4.4.5 池 . . . 83
4.5 标签 . . . 83
4.5.1 样式配置 . . . 84
4.5.2 权限 . . . 84
4.6 同意横幅 . . . 84

5 集群管理器 85
5.1 要求 . . . 85
5.2 准备节点 . . . 86
5.3 创建集群 . . . 86
5.3.1 通过 Web GUI 创建 . . . 86
5.3.2 通过命令行创建 . . . 87
5.3.3 同一网络中的多个集群 . . . 87
5.4 向集群添加节点 . . . 87
5.4.1 通过 GUI 将节点加入集群 . . . 87
5.4.2 通过命令行将节点加入集群 . . . 88
5.4.3 使用独立集群网络添加节点 . . . 89
5.5 移除集群节点 . . . 89
5.5.1 不重装而分离节点 . . . 91
5.6 仲裁 . . . 92
5.7 集群网络 . . . 93
5.7.1 网络要求 . . . 93
5.7.2 独立集群网络 . . . 93

5.7.3 Corosync 地址 96
5.8 Corosync 冗余 97
5.8.1 向现有集群添加冗余链路 97
5.9 SSH 在 Proxmox VE 集群中的作用 99
5.9.1 SSH 设置 99
5.9.2 自动执行 .bashrc 及类似文件带来的陷阱 100
5.10 Corosync 外部投票支持 100
5.10.1 QDevice 技术概览 100
5.10.2 支持的配置 101
5.10.3 QDevice-Net 设置 101
5.10.4 常见问题 102
5.11 Corosync 配置 103
5.11.1 编辑 corosync.conf 103
5.11.2 故障排查 104
5.11.3 Corosync 配置术语表 104
5.12 集群冷启动 105
5.13 客户机 VMID 自动选择 105
5.14 客户机迁移 105
5.14.1 迁移类型 105
5.14.2 迁移网络 106
5.14.3 常用命令示例 107

6 Proxmox Cluster File System (pmxcfs) 108

6.1 POSIX 兼容性 108
6.2 文件访问权限 109
6.3 技术 109
6.4 文件系统布局 109
6.4.1 文件 109
6.4.2 符号链接 110
6.4.3 用于调试的特殊状态文件（JSON） 110
6.4.4 启用/禁用调试 111
6.5 恢复 111
6.5.1 移除集群配置 111
6.5.2 从故障节点恢复/移动客户机 111

7 Proxmox VE 存储 113
7.1 存储类型 . . . 113
7.1.1 精简配置 . . . 114
7.2 存储配置 . . . 114
7.2.1 存储池 . . . 114
7.2.2 通用存储属性 . . . 115
7.3 卷 . . . 116
7.3.1 卷所有权 . . . 117
7.4 使用命令行界面 . . . 117
7.4.1 示例 . . . 117
7.5 目录后端 . . . 119
7.5.1 配置 . . . 119
7.5.2 文件命名约定 . . . 120
7.5.3 存储特性 . . . 120
7.5.4 示例 . . . 121
7.6 NFS 后端 . . . 121
7.6.1 配置 . . . 122
7.6.2 eNFS 多路径 . . . 122
7.6.3 存储特性 . . . 124
7.6.4 示例 . . . 125
7.7 CIFS 后端 . . . 125
7.7.1 配置 . . . 125
7.7.2 存储特性 . . . 126
7.7.3 示例 . . . 127
7.8 Proxmox Backup Server . . . 127
7.8.1 配置 . . . 127
7.8.2 存储特性 . . . 128
7.8.3 加密 . . . 129
7.8.4 示例：通过 CLI 添加存储 . . . 129
7.9 GlusterFS 后端 . . . 130
7.9.1 配置 . . . 130
7.9.2 文件命名约定 . . . 130
7.9.3 存储特性 . . . 130
7.9.4 示例 . . . 131
7.10 本地 ZFS Pool 后端 . . . 131

7.10.1配置 . . . 131
7.10.2文件命名约定 . . . 132
7.10.3存储特性 . . . 132
7.10.4示例 . . . 132
7.10.5常用命令示例 . . . 133
7.11 LVM 后端 . . . 133
7.11.1配置 . . . 133
7.11.2文件命名约定 . . . 133
7.11.3存储特性 . . . 134
7.11.4示例 . . . 134
7.11.5常用命令示例 . . . 134
7.12 LVM thin 后端 . . . 134
7.12.1配置 . . . 135
7.12.2文件命名约定 . . . 135
7.12.3存储特性 . . . 135
7.12.4示例 . . . 135
7.12.5常用命令示例 . . . 136
7.13 Open-iSCSI initiator . . . 136
7.13.1配置 . . . 136
7.13.2文件命名约定 . . . 137
7.13.3存储特性 . . . 137
7.13.4常用命令示例 . . . 137
7.14 用户态 iSCSI 后端 . . . 137
7.14.1配置 . . . 137
7.14.2存储特性 . . . 138
7.14.3常用命令示例 . . . 138
7.15 Ceph RADOS 块设备（RBD） . . . 138
7.15.1配置 . . . 139
7.15.2认证 . . . 139
7.15.3Ceph 客户端配置（可选） . . . 140
7.15.4存储特性 . . . 140
7.15.5常用命令示例 . . . 141
7.16 Ceph 文件系统（CephFS） . . . 141
7.16.1配置 . . . 141
7.16.2认证 . . . 142

7.16.3存储特性 . . . 142
7.16.4常用命令示例 . . . 143
7.17 BTRFS 后端 . . . 143
7.17.1配置 . . . 143
7.17.2快照 . . . 144
7.18 ZFS over iSCSI 后端 . . . 144
7.18.1配置 . . . 144
7.18.2存储特性 . . . 146

8 部署超融合 Ceph 集群 147
8.1 简介 . . . 147
8.2 术语 . . . 148
8.3 健康 Ceph 集群的建议 . . . 148
8.4 Ceph 初始安装与配置 . . . 150
8.4.1 使用基于 Web 的向导 . . . 150
8.4.2 通过 CLI 安装 Ceph 软件包 . . . 151
8.4.3 通过 CLI 进行 Ceph 初始配置 . . . 151
8.4.4 常用命令示例 . . . 151
8.5 Ceph Monitor . . . 152
8.5.1 创建 Monitor . . . 152
8.5.2 销毁 Monitor . . . 152
8.6 Ceph Manager . . . 152
8.6.1 创建 Manager . . . 152
8.6.2 销毁 Manager . . . 153
8.7 Ceph OSDs . . . 153
8.7.1 创建 OSD . . . 153
8.7.2 销毁 OSD . . . 154
8.8 Ceph Pools . . . 155
8.8.1 创建和编辑 Pool . . . 155
8.8.2 纠删码 Pool . . . 157
8.8.3 销毁 Pool . . . 158
8.8.4 PG Autoscaler . . . 158
8.9 Ceph CRUSH 与设备类别 . . . 159
8.10 Ceph Client . . . 160
8.11 CephFS . . . 161
8.11.1 Metadata Server (MDS) . . . 161

8.11.2 创建 CephFS . . . 161
8.11.3 销毁 CephFS . . . 162
8.12 Ceph 维护 . . . 163
8.12.1 替换 OSD . . . 163
8.12.2 Trim/Discard . . . 163
8.12.3 Scrub & Deep Scrub . . . 163
8.12.4 关闭 Proxmox VE + Ceph HCI 集群 . . . 163
8.13 Ceph 监控与故障排查 . . . 164
8.13.1 故障排查 . . . 164

9 存储复制 166
9.1 支持的存储类型 . . . 166
9.2 调度格式 . . . 167
9.3 错误处理 . . . 167
9.3.1 可能的问题 . . . 167
9.3.2 发生错误时迁移来宾 . . . 167
9.3.3 示例 . . . 167
9.4 管理作业 . . . 168
9.5 网络 . . . 168
9.6 命令行接口示例 . . . 168

10 QEMU/KVM 虚拟机 170
10.1 模拟设备和半虚拟化设备 . . . 170
10.2 虚拟机设置 . . . 171
10.2.1 常规设置 . . . 171
10.2.2 操作系统设置 . . . 171
10.2.3 系统设置 . . . 171
10.2.4 硬盘 . . . 172
10.2.5 CPU . . . 174
10.2.6 内存 . . . 179
10.2.7 内存加密 . . . 180
10.2.8 网络设备 . . . 182
10.2.9 显示 . . . 183
10.2.10 USB 直通 . . . 184
10.2.11 BIOS 和 UEFI . . . 184
10.2.12 可信平台模块 (TPM) . . . 185

10.2.13 VM 间共享内存 . . . 186
10.2.14 音频设备 . . . 186
10.2.15 VirtIO RNG . . . 186
10.2.16 Virtiofs . . . 187
10.2.17 设备启动顺序 . . . 188
10.2.18 虚拟机自动启动和关闭 . . . 189
10.2.19 QEMU Guest Agent . . . 189
10.2.20 SPICE 增强 . . . 191
10.3 迁移 . . . 192
10.3.1 在线迁移 . . . 192
10.3.2 离线迁移 . . . 193
10.4 复制和克隆 . . . 193
10.5 虚拟机模板 . . . 194
10.6 VM Generation ID . . . 194
10.7 导入虚拟机 . . . 194
10.7.1 导入向导 . . . 195
10.7.2 通过 CLI 导入 OVF/OVA . . . 195
10.8 Cloud-Init 支持 . . . 197
10.8.1 准备 Cloud-Init 模板 . . . 197
10.8.2 部署 Cloud-Init 模板 . . . 198
10.8.3 自定义 Cloud-Init 配置 . . . 198
10.8.4 Windows 上的 Cloud-Init . . . 199
10.8.5 准备 Cloudbase-Init 模板 . . . 199
10.8.6 Cloudbase-Init 与 Sysprep . . . 200
10.8.7 Cloud-Init 专用选项 . . . 201
10.9 PCI(e) 直通 . . . 202
10.9.1 通用要求 . . . 202
10.9.2 主机设备直通 . . . 205
10.9.3 SR-IOV . . . 208
10.9.4 中介设备（vGPU、GVT-g） . . . 208
10.9.5 在集群中使用 . . . 209
10.9.6 vIOMMU（模拟 IOMMU） . . . 209
10.10 Hook 脚本 . . . 210
10.11 休眠 . . . 210
10.12 资源映射 . . . 211

10.13 使用 qm 管理虚拟机 212
10.13.1 CLI 使用示例 212
10.14 配置 213
10.14.1 文件格式 213
10.14.2 快照 214
10.14.3 选项 214
10.15 锁定 239

11 Proxmox 容器工具包 240
11.1 技术概览 240
11.2 支持的发行版 241
11.2.1 Alpine Linux 241
11.2.2 Arch Linux 241
11.2.3 CentOS, Almalinux, Rocky Linux 241
11.2.4 Debian 242
11.2.5 Devuan 242
11.2.6 Fedora 242
11.2.7 Gentoo 243
11.2.8 OpenSUSE 243
11.2.9 Ubuntu 243
11.3 容器镜像 243
11.4 容器设置 244
11.4.1 常规设置 244
11.4.2 CPU 245
11.4.3 内存 246
11.4.4 挂载点 246
11.4.5 网络 249
11.4.6 容器的自动启动和关闭 250
11.4.7 Hook 脚本 250
11.5 安全注意事项 250
11.5.1 AppArmor 250
11.5.2 Control Groups (*cgroup*) 251
11.6 客户机操作系统配置 252
11.7 容器存储 254
11.7.1 FUSE 挂载 254
11.7.2 在容器内使用 Quota 254

11.7.3在容器内使用 ACL . . . 255
11.7.4容器挂载点备份 . . . 255
11.7.5容器挂载点复制 . . . 255
11.8备份和还原 . . . 255
11.8.1容器备份 . . . 255
11.8.2还原容器备份 . . . 255
11.9使用 pct 管理容器 . . . 256
11.9.1CLI 使用示例 . . . 256
11.9.2常用命令示例 . . . 257
11.9.3获取调试日志 . . . 257
11.10迁移 . . . 258
11.11配置 . . . 258
11.11.1文件格式 . . . 259
11.11.2快照 . . . 259
11.11.3选项 . . . 259
11.12锁 . . . 264

12软件定义网络 265
12.1简介 . . . 265
12.2支持状态 . . . 265
12.2.1历史 . . . 265
12.2.2当前状态 . . . 265
12.3安装 . . . 266
12.3.1SDN 核心 . . . 266
12.3.2DHCP IPAM . . . 266
12.3.3FRRouting . . . 266
12.4配置概览 . . . 267
12.5技术与配置 . . . 267
12.6区域 . . . 267
12.6.1通用选项 . . . 268
12.6.2Simple 区域 . . . 268
12.6.3VLAN 区域 . . . 268
12.6.4QinQ 区域 . . . 269
12.6.5VXLAN 区域 . . . 269
12.6.6EVPN 区域 . . . 270
12.7VNet . . . 270

12.8 子网 . . . 271
12.9 控制器 . . . 272
12.9.1 EVPN Controller . . . 272
12.9.2 BGP Controller . . . 272
12.9.3 ISIS Controller . . . 273
12.10 IPAM . . . 273
12.10.1 PVE IPAM Plugin . . . 274
12.10.2 NetBox IPAM Plugin . . . 274
12.10.3 phpIPAM Plugin . . . 274
12.11 DNS . . . 274
12.11.1 PowerDNS Plugin . . . 275
12.12 DHCP . . . 275
12.12.1 配置 . . . 275
12.12.2 Plugins . . . 276
12.13 防火墙集成 . . . 277
12.13.1 VNet 和子网 . . . 277
12.13.2 IPAM . . . 278
12.14 示例 . . . 278
12.14.1 Simple 区域示例 . . . 278
12.14.2 源 NAT 示例 . . . 279
12.14.3 VLAN 设置示例 . . . 279
12.14.4 QinQ 设置示例 . . . 280
12.14.5 VXLAN 设置示例 . . . 281
12.14.6 EVPN 设置示例 . . . 281
12.15 说明 . . . 283
12.15.1 多个 EVPN 出口节点 . . . 283
12.15.2 VXLAN IPSEC 加密 . . . 283

13 Proxmox VE 防火墙 285
13.1 方向与区域 . . . 285
13.1.1 方向 . . . 285
13.1.2 区域 . . . 286
13.2 配置文件 . . . 286
13.2.1 集群范围设置 . . . 286
13.2.2 主机专用配置 . . . 288
13.2.3 VM/容器配置 . . . 289

13.2.4 VNet 配置 . . . 290
13.3 防火墙规则 . . . 291
13.4 安全组 . . . 293
13.5 IP 别名 . . . 293
13.5.1 标准 IP 别名 local_network . . . 293
13.6 IP 集 . . . 294
13.6.1 标准 IP 集 management . . . 294
13.6.2 标准 IP 集 blacklist . . . 294
13.6.3 标准 IP 集 ipfilter-net* . . . 294
13.7 服务和命令 . . . 295
13.8 默认防火墙规则 . . . 295
13.8.1 数据中心入站/出站 DROP/REJECT . . . 295
13.8.2 VM/CT 入站/出站 DROP/REJECT . . . 296
13.9 防火墙规则日志 . . . 296
13.9.1 用户定义防火墙规则的日志记录 . . . 297
13.10 提示与技巧 . . . 297
13.10.1 如何允许 FTP . . . 297
13.10.2 Suricata IPS 集成 . . . 298
13.11 IPv6 说明 . . . 298
13.12 Proxmox VE 使用的端口 . . . 298
13.13 nftables . . . 299
13.13.1 安装和使用 . . . 299
13.13.2 使用 . . . 300
13.13.3 常用命令 . . . 300

14 用户管理 302
14.1 用户 . . . 302
14.1.1 系统管理员 . . . 303
14.2 组 . . . 303
14.3 API token . . . 303
14.4 资源池 . . . 303
14.5 认证域 . . . 303
14.5.1 Linux PAM 标准认证 . . . 304
14.5.2 Proxmox VE Authentication Server . . . 304
14.5.3 LDAP . . . 304
14.5.4 Microsoft Active Directory (AD) . . . 305

14.5.5同步基于 LDAP 的域 . . . 306
14.5.6OpenID Connect . . . 308
14.6双因素认证 . . . 310
14.6.1可用的第二因素 . . . 310
14.6.2认证域强制双因素认证 . . . 310
14.6.3双因素认证限制与锁定 . . . 311
14.6.4用户配置的 TOTP 认证 . . . 311
14.6.5TOTP . . . 311
14.6.6WebAuthn . . . 311
14.6.7Recovery Keys . . . 312
14.6.8服务器端 Webauthn 配置 . . . 312
14.6.9服务器端 U2F 配置 . . . 312
14.6.10用户启用 U2F . . . 313
14.7权限管理 . . . 313
14.7.1角色 . . . 313
14.7.2权限 . . . 314
14.7.3对象和路径 . . . 316
14.7.4池 . . . 317
14.7.5我需要哪些权限？ . . . 317
14.8命令行工具 . . . 318
14.9真实场景示例 . . . 318
14.9.1管理员组 . . . 318
14.9.2审计员 . . . 319
14.9.3委派用户管理 . . . 319
14.9.4用于监控的受限 API token . . . 319
14.9.5资源池 . . . 320

15高可用性 321
15.1要求 . . . 322
15.2资源 . . . 322
15.3管理任务 . . . 323
15.4工作原理 . . . 324
15.4.1服务状态 . . . 324
15.4.2本地资源管理器 . . . 325
15.4.3集群资源管理器 . . . 326
15.5HA 模拟器 . . . 327

15.6 配置 . . . 328
15.6.1 资源 . . . 328
15.6.2 组 . . . 329
15.7 Fencing . . . 330
15.7.1 Proxmox VE 如何执行 Fencing . . . 330
15.7.2 配置硬件 Watchdog . . . 331
15.7.3 恢复已 Fenced 的服务 . . . 331
15.8 启动失败策略 . . . 331
15.9 错误恢复 . . . 332
15.10 软件包更新 . . . 332
15.11 节点维护 . . . 332
15.11.1 维护模式 . . . 332
15.11.2 关机策略 . . . 333
15.12 集群资源调度 . . . 335
15.12.1 Basic 调度器 . . . 335
15.12.2 Static-Load 调度器 . . . 335
15.12.3 CRS 调度点 . . . 335

16 备份与还原 337
16.1 备份模式 . . . 337
16.1.1 虚拟机备份 Fleecing . . . 339
16.1.2 CT 变更检测模式 . . . 339
16.2 备份文件名 . . . 340
16.3 备份文件压缩 . . . 340
16.4 备份加密 . . . 340
16.5 备份作业 . . . 340
16.6 备份保留 . . . 341
16.6.1 Prune 模拟器 . . . 342
16.6.2 保留设置示例 . . . 342
16.7 备份保护 . . . 342
16.8 备份备注 . . . 342
16.9 还原 . . . 343
16.9.1 带宽限制 . . . 343
16.9.2 实时还原（Live-Restore） . . . 344
16.9.3 单文件还原 . . . 344
16.10 配置 . . . 344
16.11 钩子脚本 . . . 347
16.12 文件排除 . . . 347
16.13 示例 . . . 348

17 通知 350
17.1 概述 . . . 350
17.2 通知目标 . . . 350
17.2.1 Sendmail . . . 350
17.2.2 SMTP . . . 351
17.2.3 Gotify . . . 352
17.2.4 Webhook . . . 353
17.3 通知匹配器 . . . 355
17.3.1 匹配器选项 . . . 355
17.3.2 日历匹配规则 . . . 355
17.3.3 字段匹配规则 . . . 355
17.3.4 严重级别匹配规则 . . . 356
17.3.5 示例 . . . 356
17.4 通知事件 . . . 357
17.5 系统邮件转发 . . . 357
17.6 权限 . . . 357
17.7 通知模式 . . . 357
17.8 覆盖通知模板 . . . 358
17.9 常用命令示例 . . . 358

18 重要服务守护进程 359
18.1 pvedaemon - Proxmox VE API 守护进程 . . . 359
18.2 pveproxy - Proxmox VE API 代理守护进程 . . . 359
18.2.1 基于主机的访问控制 . . . 359
18.2.2 监听 IP 地址 . . . 360
18.2.3 SSL 加密套件 . . . 361
18.2.4 支持的 TLS 版本 . . . 361
18.2.5 Diffie-Hellman 参数 . . . 361
18.2.6 替代 HTTPS 证书 . . . 362
18.2.7 响应压缩 . . . 362
18.2.8 真实客户端 IP 日志记录 . . . 362
18.3 pvestatd - Proxmox VE 状态守护进程 . . . 362
18.3.1 常用命令示例 . . . 363
18.4 spiceproxy - SPICE 代理服务 . . . 363
18.4.1 基于主机的访问控制 . . . 363
18.5 pvescheduler - Proxmox VE 调度器守护进程 . . . 363
18.5.1 常用命令示例 . . . 363

19 实用命令行工具 **364**
19.1 pvesubscription - 订阅管理 . . . 364
19.1.1 常用命令示例 . . . 364
19.2 pveperf - Proxmox VE 基准测试脚本 . . . 364
19.3 pvebcache - bcache 管理工具 . . . 365
19.3.1 后端设备管理 . . . 365
19.3.2 缓存设备管理 . . . 366
19.3.3 缓存策略 . . . 367
19.3.4 常用命令示例 . . . 367
19.4 Proxmox VE API 的 Shell 接口 . . . 367
19.4.1 示例 . . . 368

20 常见问题 **369**

21 参考文献 **372**
21.1 关于 Proxmox VE 的书籍 . . . 372
21.2 关于相关技术的书籍 . . . 372
21.3 关于相关主题的书籍 . . . 373

A 命令行界面 **374**
A.1 常规 . . . 374
A.2 输出格式选项 [FORMAT_OPTIONS] . . . 374
A.3 **pvesm** - Proxmox VE 存储管理器 . . . 375
A.4 **pvesubscription** - Proxmox VE 订阅管理器 . . . 388
A.5 **pveperf** - Proxmox VE 基准测试脚本 . . . 388
A.6 **pvebcache** - PXVIRT bcache 管理工具 . . . 389
A.7 **pveceph** - 管理 Proxmox VE 节点上的 CEPH 服务 . . . 390
A.8 **pvenode** - Proxmox VE 节点管理 . . . 397
A.9 **pvesh** - Proxmox VE API 的 Shell 接口 . . . 404
A.10 **qm** - QEMU/KVM 虚拟机管理器 . . . 406
A.11 **qmrestore** - 还原 QemuServer vzdump 备份 . . . 452
A.12 **pct** - Proxmox 容器工具包 . . . 452
A.13 **pveam** - Proxmox VE Appliance 管理器 . . . 475
A.14 **pvecm** - Proxmox VE 集群管理器 . . . 476
A.15 **pvesr** - Proxmox VE 存储复制 . . . 479
A.16 **pveum** - Proxmox VE 用户管理器 . . . 483
A.17 **vzdump** - 虚拟机和容器备份工具 . . . 497
A.17.1 常用命令示例 . . . 500
A.18 **ha-manager** - Proxmox VE HA 管理器 . . . 500
A.18.1 常用命令示例 . . . 504

B 服务守护进程 505
B.1 **pve-firewall** - Proxmox VE 防火墙守护进程 . . . 505
B.1.1 常用命令示例 . . . 506
B.2 **pvedaemon** - Proxmox VE API 守护进程 . . . 506
B.3 **pveproxy** - Proxmox VE API 代理守护进程 . . . 507
B.4 **pvestatd** - Proxmox VE 状态守护进程 . . . 508
B.5 **spiceproxy** - SPICE 代理服务 . . . 508
B.5.1 常用命令示例 . . . 509
B.6 **pmxcfs** - Proxmox 集群文件系统 . . . 509
B.7 **pve-ha-crm** - 集群资源管理器守护进程 . . . 510
B.7.1 常用命令示例 . . . 510
B.8 **pve-ha-lrm** - 本地资源管理器守护进程 . . . 511
B.9 **pvescheduler** - Proxmox VE 调度器守护进程 . . . 511
B.9.1 常用命令示例 . . . 512

C 配置文件 513
C.1 常规 . . . 513
C.1.1 属性名称的大小写形式 . . . 513
C.2 数据中心配置 . . . 513
C.2.1 文件格式 . . . 513
C.2.2 选项 . . . 514

D 日历事件 518
D.1 调度格式 . . . 518
D.2 详细规范 . . . 518
D.2.1 示例： . . . 519

E QEMU vCPU 列表 520
E.1 简介 . . . 520
E.2 Intel CPU 类型 . . . 520
E.3 AMD CPU 类型 . . . 521
E.4 ARM64 CPU 类型 . . . 522
E.5 riscv64 CPU 类型 . . . 522
E.6 loongarch64 CPU 类型 . . . 522

F 防火墙宏定义 523

G Markdown 入门 **538**

G.1 Markdown 基础 538

G.1.1 标题 538

G.1.2 强调 538

G.1.3 链接 539

G.1.4 列表 539

G.1.5 表格 540

G.1.6 块引用 540

G.1.7 代码和片段 540

H GNU 自由文档许可证 **542**

Chapter 1

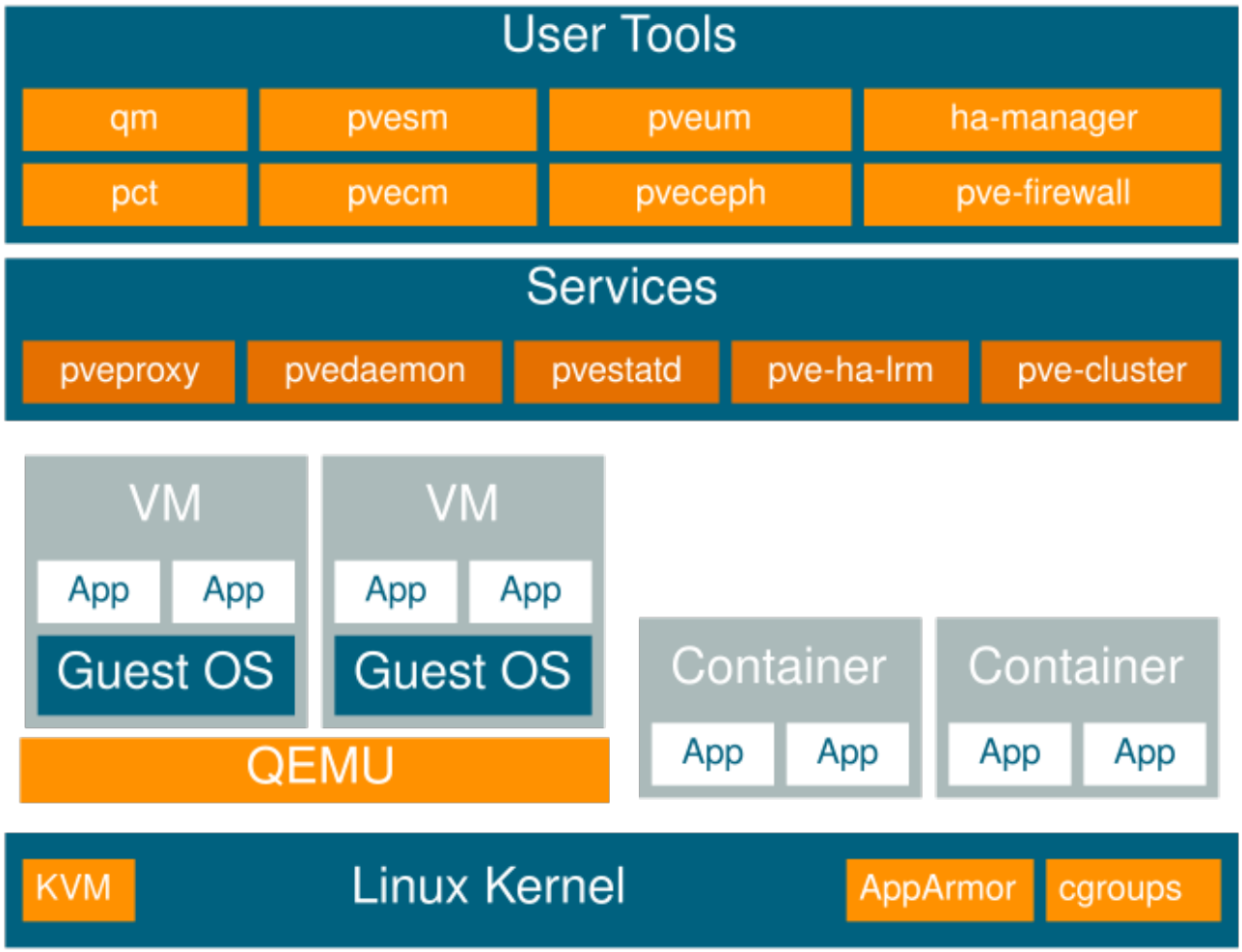
简介

Proxmox VE 是用于运行虚拟机和容器的平台。它基于 OpenEuler Linux，并且完全开源。为了获得最大灵活性，我们实现了两种虚拟化技术：基于内核的虚拟机（KVM）和基于容器 的虚拟化（LXC

一个主要设计目标是尽可能简化管理。你可以在单个节点上使用 Proxmox VE，也可以组建由 多个节点 Web 的管理界面完成，即使是新手 用户也能在数分钟内完成 Proxmox VE 的设置和安装。

Important

PXVIRT 是基于 Proxmox VE 开源代码和相关生态组件适配、独立维护的虚拟化平台，面向 openEuler 和多架构环境提供打包、集成与功能扩展。由于部分上游组件名称、命令、API、配置路径及历史术语沿用 PVE/Proxmox 命名，本文档中可能仍出现 PVE、pve、Proxmox 等字样。这些名称仅用于说明技术来源、兼容接口或组件名称。PXVIRT 不是 Proxmox Server Solutions GmbH 的官方产品，也不隶属于 Proxmox 项目。 Proxmox 和 Proxmox VE 是 Proxmox Server Solutions GmbH 的商标。



1.1 集中管理

虽然许多人从单节点开始使用，但 Proxmox VE 可以横向扩展为包含大量节点的集群。集群 软件栈已完

独特的多主设计

集成的基于 Web 的管理界面可以清晰展示所有 KVM 客户机和 Linux 容器，甚至可以 展示整个集 GUI 轻松管理虚拟机、容器、存储或集群，无需安装单独、 复杂且昂贵的管理服务器。

Proxmox Cluster File System (pmxcfs)

Proxmox VE 使用独特的 Proxmox Cluster file system（pmxcfs），这是一个用于保存配置 文件、由数据库驱动的文件系统。它使你能够保存数千台虚拟机的配置。通过使用 corosync，这 的持久化数据库中，同时在 RAM 中保留一份数据副本，最大存储容量为 30MB；这对于 数千台虚

Proxmox VE 是唯一使用这种独特集群文件系统的虚拟化平台。

基于 **Web** 的管理界面

Proxmox VE 易于使用。管理任务可以通过随附的基于 Web 的管理界面完成，无需安装单独的 管理工具，也不需要带有庞大数据库的额外管理节点。多主工具允许你从集群中的任意 节点管理整 JavaScript Framework（ExtJS）的集中式 Web 管理界面，使 你可以通过 GUI 控制所有功能，并 syslog。这包括运行 备份或还原作业、在线迁移或 HA 触发的活动。

命令行
: 对于习惯 Unix shell 或 Windows Powershell 便利性的高级用户，Proxmox VE 提供了命令行界面，用于管理虚拟环境的所有组件。该命令行界面具备智能 Tab 补全，并以 UNIX man page 的形式提供完整文档。

REST API
: Proxmox VE 使用 RESTful API。我们选择 JSON 作为主要数据格式，并使用 JSON Schema 对 整个 API 进行正式定义。这使第三方管理工具能够快速、轻松地集成，例如定制化托管 环境。

基于角色的管理
: 你可以使用基于角色的用户和权限管理，为所有对象（如虚拟机、存储、节点等）定义 细粒度访问控制列表：每条权限都会在特定路径上指定一个主体（用户或组）和一个角色（权限 集合）。

认证领域
: Proxmox VE 支持多种认证来源，例如 Microsoft Active Directory、LDAP、Linux PAM 标准 认证，或内置的 Proxmox VE 认证服务器。

1.2 灵活存储

Proxmox VE 的存储模型非常灵活。虚拟机镜像既可以存储在一个或多个本地存储上，也可以 存储在 NFS、SAN 等共享存储上。配置数量没有限制，你可以按需配置任意数量的存储 定义。所有 OpenEuler Linux 可用的存储技术都可以使用。

将虚拟机存储在共享存储上的一个主要优势，是可以在不停机的情况下在线迁移正在 运行的机器，因为

当前支持以下网络存储类型：

- LVM Group（通过 iSCSI target 提供网络后端）
- iSCSI target
- NFS 共享
- CIFS 共享
- Ceph RBD
- 直接使用 iSCSI LUN
- GlusterFS

支持的本地存储类型包括：

- LVM Group（块设备、FC 设备、DRBD 等本地后端设备）
- Directory（现有文件系统上的存储）
- ZFS

1.3 集成备份与还原

集成备份工具（vzdump）可以为正在运行的容器和 KVM 客户机创建一致性快照。它 本质上会创建包含

KVM 在线备份适用于所有存储类型，包括位于 NFS、CIFS、iSCSI LUN、Ceph RBD 上的 虚拟机镜像。
乱序数据、最小化 I/O）。

1.4 高可用集群

多节点 Proxmox VE HA 集群支持定义高可用虚拟服务器。Proxmox VE HA 集群基于经过验证的
Linux HA 技术，可提供稳定可靠的 HA 服务。

1.5 灵活网络

Proxmox VE 使用桥接网络模型。所有虚拟机可以共享同一个网桥，就像来自每个客户机的 虚拟网线都
并分配 TCP/IP 配置。

为了获得更高灵活性，还可以使用 VLAN（IEEE 802.1q）和网络绑定/聚合。通过这种 方式，可以为
Proxmox VE 主机构建复杂、灵活的虚拟网络，充分发挥 Linux 网络栈的能力。

1.6 集成防火墙

集成防火墙允许你在任意虚拟机或容器接口上过滤网络数据包。常用的防火墙规则集合 可以归组为
“security groups”。

1.7 超融合基础设施

Proxmox VE 是一个虚拟化平台，紧密集成计算、存储和网络资源，并管理高可用集群、 备份/还原以及

因此，可以通过集中式 Web 管理界面像管理单一系统一样管理这些组件。这些能力使 Proxmox
VE 成为部署和管理开源 hyper-converged infrastructure 的理想选择。

1.7.1 使用 **Proxmox VE** 构建超融合基础设施（**HCI**）的优势

超融合基础设施（HCI）尤其适用于基础设施需求较高但管理预算有限的部署场景， 也适用于远程和分

HCI 提供以下优势：

- 可扩展性：可以无缝扩展计算、网络和存储设备（即快速且相互独立地扩展 服务器和存储）。
- 低成本：Proxmox VE 是开源软件，并集成计算、存储、网络、备份和管理中心等所需 组件。它可以
- 数据保护和效率：集成了备份和灾难恢复等服务。

- 简洁性：配置简单，并支持集中式管理。
- 开源：无厂商锁定。

1.7.2 超融合基础设施：存储

Proxmox VE 对部署超融合存储基础设施提供了紧密集成的支持。例如，可以仅通过 Web 界面部署并管

- **Ceph**：一种具备自修复和自管理能力的共享、可靠且高度可扩展的存储系统。 请参见 如何在 Proxmox VE 节点上管理 Ceph 服务
- **ZFS**：结合文件系统和逻辑卷管理器，提供广泛的数据损坏防护、多种 RAID 模式、快速且成本低的 如何在 Proxmox VE 节点上发挥 ZFS 的能力。

除上述技术外，Proxmox VE 还支持集成多种其他存储技术。可以在 存储管理器章节中了解相关内容。

1.8 为什么选择开源

Proxmox VE 使用 Linux 内核，并基于 Debian GNU/Linux 发行版。Proxmox VE 的源代码按照 GNU Affero General Public License, version 3 发布。这意味着你可以随时自由查看源代码，也可以 贡献代码。

在 Proxmox，我们承诺尽可能使用开源软件。使用开源软件可确保完整访问所有功能，同时获得高安全 基于其继续构建，或向项目提交变更。我们鼓励所有人参与贡献，同时 Proxmox 会确保 产品始终满足

开源软件还有助于降低成本，并使核心基础架构不依赖单一厂商。

Pxvirt 基于 Openeuler 发行版。将Proxmox VE原生软件包移植到国产的OS中，这意味着即便不用商

1.9 使用 **Proxmox VE** 的优势

- 开源软件
- 无厂商锁定
- Linux 内核
- 安装快速且易于使用
- 基于 Web 的管理界面
- REST API
- 活跃且庞大的社区
- 管理成本低，部署简单

1.10 获取帮助

1.10.1 Pxvirt Wiki

主要信息来源是 Pxvirt Wiki。它将参考文档与用户贡献的内容结合在一起。

1.10.2 商业支持

梨儿方为Pxvirt 提供企业级支持，为政务事业单位提供信创专供版本，符合信息安全要求。可前往我们

如需批量折扣或更多一般信息，请联系 Service@lierfang.com.

1.10.3 缺陷跟踪器

梨儿方 在 https://github.com/jiangcuo/pxvirt 运行公开缺陷跟踪器。如果发现问题，请在 那里提交
持续跟踪问题，并会在问题解决后发送通知。

1.11 项目历史

该项目始于 2007 年，并在 2008 年发布了第一个稳定版本。当时我们使用 OpenVZ 运行 容器，使用
KVM 的 QEMU 运行虚拟机。那时集群功能较为有限，用户界面也很简单（由服务器生成的网页）。

但我们很快基于 Corosync 集群栈开发了新 功能，并引入了新的 Proxmox cluster file system（pmx
因为它完全向用户隐藏了集群复杂性。管理一个 16 节点集群就像管理单个节点一样 简单。

我们引入了新的 REST API，并使用 JSON-Schema 编写了完整的声明式规范，使其他人 能够将
Proxmox VE 集成到自己的基础架构中，也让提供附加服务变得更加容易。

此外，新的 REST API 使我们能够用基于 JavaScript 的现代客户端单页应用替换原始 用户界面。我们还
noVNC 替换了旧的基于 Java 的 VNC 控制台代码。因此，你只需要一个 Web 浏览器即可管理虚拟机。

支持多种存储类型是另一项重要工作。值得注意的是，Proxmox VE 是 2014 年第一个默认 随附
ZFS on Linux 的发行版。另一个里程碑是在 hypervisor 节点上运行和管理 Ceph 存储的能力。这类部
具有极高的成本效益。

项目初期，我们是最早为 KVM 提供商业支持的公司之一。KVM 项目本身持续演进，如今 已成为广泛使
hypervisor。每个版本都会带来新功能。我们开发了 KVM 在线备份 功能，使在任意存储类型上创建快

4.0 版本最显著的变化是从 OpenVZ 迁移到 LXC。容器 如今已深度集成，并且可以使用与虚拟机相同的
易用的 高可用（HA）管理器，简化了高可用 部署的配置和管理。

在 Proxmox VE 5 的开发期间，我们引入了异步 存储复制， 以及使用 ACME/Let's Encrypt
的自动化 证书管理，同时还加入了许多 其他功能。

Software Defined Network (SDN) 栈是与社区合作开发的。它在 6.2 版本中作为实验性功能集成到
Web 界面中，简化了复杂网络配置的管理。自 8.1 版本起，SDN 集成已获得完整支持，并默认安装。

2020 年发布了一个新项目： Proxmox Backup Server。Proxmox Backup Server 与 Prox-
mox VE 深度集成，通过实现增量备份、 重复数据删除等能力，显著提升了备份能力。

另一个新工具 Proxmox Offline Mirror 于 2022 年发布， 使无法连接公共互联网的系统也可以使用订阅

呼声很高的 Web 界面深色主题于 2023 年引入。同年晚些时候，8.0 版本集成了对 Ceph enterprise repository 的访问。现在，任何 Proxmox VE 订阅都包含对最稳定 Ceph 仓库的 访问权限。

8.2 版本为官方 ISO installer 引入了自动化和无人值守 安装，显著简化了 Proxmox VE 的大规模部署。

同样在 8.2 版本中引入的 导入向导，使用户 能够轻松高效地直接从 VMware ESXi 等其他 hypervisor 迁移客户机 [1]。 此外，现在还可以在 Web 界面中从基于文件的存储直接导入 Open Virtualization Format（OVF/OVA）归档。

1.12 Pxvirt 项目历史

2022 年，我们发行了基于pve 7.2的arm版本 2023 年，我们发行了基于pve 8.0的loongarch64版本
2024 年，我们发行了飞腾专用版本 2025 年，我们发行了基于openeuler的版本 目前，Pxvirt已经支

1.13 改进 Proxmox VE 文档

始终欢迎对 Proxmox VE 文档进行贡献和改进。可以通过多种方式参与贡献。

如果你在本文档中发现错误或其他可改进之处，请在 Proxmox 缺陷跟踪器 提交 bug，以提出修正建议

如果要提出新的内容，请选择以下选项之一：

- Wiki：对于特定部署、操作指南或教程，wiki 是合适的贡献位置。
- 参考文档：对于对所有用户都有帮助的通用内容，请向参考文档提交贡献建议。 这包括有关安装、配 Proxmox VE 功能问题的所有信息。参考文档使用 AsciiDoc 格式 编写。要编辑该文档， 需要克隆位 `git://git.proxmox.com/git/pve-docs.git` 的 git 仓库， 然后按照 README.adoc 文档操作。

Note
如果你有兴趣参与 Proxmox VE 代码库开发， 开发者文档 wiki 文章会说明从何处开始。

1.14 翻译 Proxmox VE

Proxmox VE 用户界面默认使用英语。不过，得益于社区贡献，也提供了其他语言的翻译。 我们欢迎任

我们使用 gettext 管理翻译文件。 Poedit 等工具提供了便于编辑翻译文件的用户界面， 但你也可以使

1.14.1 使用 git 翻译

语言文件以 git 仓库形式提供。 如果你熟悉 git，请按照我们的 开发者文档进行贡献。

可以通过以下步骤创建新的翻译（将 <LANG> 替换为语言 ID）：

[1] Migrate to Proxmox VE https://pve.proxmox.com/wiki/Migrate_to_Proxmox_VE

```
# git clone git://git.proxmox.com/git/proxmox-i18n.git
# cd proxmox-i18n
# make init-<LANG>.po
```

也可以使用你选择的编辑器编辑已有翻译：

```
# poedit <LANG>.po
```

1.14.2 不使用 git 翻译

即使你不熟悉 git，也可以帮助翻译 Proxmox VE。首先，可以从 这里下载语言文件。 找到要改进的语
”raw” 链接并选择 *Save Link As...*。修改文件后，将最终翻译连同已签署的 贡献者许可协议 直接发送
office(at)proxmox.com。

1.14.3 测试翻译

为了让 Proxmox VE 使用该翻译，必须先将 `.po` 文件转换为 `.js` 文件。可以调用同一仓库中的以下脚本

```
# ./po2js.pl -t pve xx.po >pve-lang-xx.js
```

随后可将生成的文件 `pve-lang-xx.js` 复制到 Proxmox 服务器上的 `/usr/share/pve-i18n` 目录中进行测试。

也可以在仓库根目录运行以下命令构建 deb 包：

```
# make deb
```

Important
无论使用哪种方法，都需要在系统中安装以下 perl 包。对于 Debian/Ubuntu：

```
# apt-get install perl liblocale-po-perl libjson-perl
```

1.14.4 发送翻译

可以将完成的翻译（`.po` 文件）连同已签署的贡献者许可协议发送到 Proxmox 团队的 office(at)proxm
Proxmox VE 开发邮件列表。参见 开发者文档。

1.14.5 常用命令示例

生成并测试某个语言的 JavaScript 翻译文件：

```
# ./po2js.pl -t pve zh_CN.po >pve-lang-zh_CN.js
# cp pve-lang-zh_CN.js /usr/share/pve-i18n/
```

Chapter 2

安装 Proxmox VE

Proxmox VE 基于 OpenEuler。因此，梨儿方 提供的安装磁盘镜像（ISO 文件）中包含完整的 OpenEuler 系统，以及所有必需的 Proxmox VE 软件包。

Tip
请参阅 FAQ 中的支持表，了解 Proxmox VE 版本与 OpenEuler 版本之间的对应关系。

安装程序会引导你完成设置，包括对本地磁盘进行分区、应用基础系统配置（例如时区、 语言、网络）用户，推荐使用提供的 ISO 进行安装。

也可以在现有 OpenEuler 系统之上安装 Proxmox VE。该选项仅推荐高级用户使用，因为需要对 Proxmox VE 有较深入的了解。

2.1 系统要求

在生产环境中运行 Proxmox VE 时，建议使用高质量服务器硬件。为进一步降低主机故障的影响，可以将 Proxmox VE 作为集群运行，并为虚拟机和容器启用高可用（HA）。

Proxmox VE 可以使用本地存储（DAS）、SAN、NAS，以及 Ceph RBD 等分布式存储。详情请参见存储章节。

2.1.1 最低要求，仅用于评估

这些最低要求仅适用于评估用途，不应在生产环境中使用。

- CPU: X86_64 \ AARCH64 \ LOONGARCH64 \Riscv64
- RAM：4 GB RAM，另需为客户机预留额外 RAM
- 硬盘
- 一块网卡（NIC）

2.1.2 推荐系统要求

- 内存：操作系统和 Proxmox VE 服务至少需要 2 GB，此外还需为客户机分配内存。Ceph 和 ZFS 需要额外内存；每 TB 已用存储大约需要 1 GB 内存。
- 快速且具备冗余的存储；使用 SSD 可获得最佳效果。
- OS 存储：使用带电池保护写缓存（“BBU”）的硬件 RAID，或使用非 RAID 配合 ZFS （可选 SSD 用于 ZIL）。
- 虚拟机存储：
  - 对于本地存储，使用带电池保护写缓存（BBU）的硬件 RAID，或为 ZFS 和 Ceph 使用非 RAID。Z 和 Ceph 都不兼容硬件 RAID 控制器。
  - 可以使用共享存储和分布式存储。
  - 为获得良好性能，建议使用具备断电保护（PLP）的 SSD。不建议使用消费级 SSD。
- 冗余的（多）千兆网卡，并根据所选存储技术和集群设置增加额外网卡。
- 对于 PCI(e) 直通，CPU 需要支持 VT-d/AMD-d / Smmu 标志。

2.1.3 简单性能概览

要快速了解已安装 Proxmox VE 系统的 CPU 和硬盘性能，可运行随附的 `pveperf` 工具。

Note
这只是一个非常快速且概括性的基准测试。建议执行更详细的测试，尤其是针对系统 I/O 性能的测试。

2.1.4 常用命令示例

在 Proxmox VE 节点 shell 中运行以下命令，可查看基本性能指标：

```
pveperf
```

2.1.5 访问 **Web** 界面支持的浏览器

要访问基于 Web 的用户界面，建议使用以下浏览器之一：

- Firefox：当年发布版本，或最新的 Extended Support Release
- Chrome：当年发布版本
- Microsoft 当前支持的 Edge 版本
- Safari：当年发布版本

从移动设备访问时，Proxmox VE 会显示轻量级、基于触控的界面。

2.2 准备安装介质

从以下地址下载安装程序 ISO 镜像：https://www.lierfang.com
Proxmox VE 安装介质是一个混合 ISO 镜像，可通过两种方式使用：

- 可刻录到 CD 或 DVD 的 ISO 镜像文件。
- 可复制到 USB 闪存盘（U 盘）的原始扇区（IMG）镜像文件。

建议使用 USB 闪存盘安装 Proxmox VE，因为这是速度更快的方式。

2.2.1 将 USB 闪存盘准备为安装介质

闪存盘至少需要有 2 GB 可用存储空间。

Note
不要使用 UNetbootin。它不能配合 Proxmox VE 安装镜像工作。

Important
请确保 USB 闪存盘未被挂载，并且其中不包含任何重要数据。

2.2.2 GNU/Linux 操作说明

在类 Unix 操作系统上，使用 dd 命令将 ISO 镜像复制到 USB 闪存盘。首先找到 USB 闪存盘的正确设备
dd 命令。

```
# dd bs=1M conv=fdatasync if=./proxmox-ve_*.iso of=/dev/XYZ
```

Note
请务必将 /dev/XYZ 替换为正确的设备名称，并相应调整输入文件名（if）路径。

Caution
务必非常谨慎，不要覆盖错误的磁盘。

查找正确的 **USB** 设备名称

有两种方式可以找出 USB 闪存盘的名称。第一种是在插入闪存盘前后比较 dmesg 命令输出的最后几行。第二种是比较 `lsblk` 命令的输出。打开终端并运行：

```
# lsblk
```

然后插入 USB 闪存盘，并再次运行该命令：

```
# lsblk
```

将会出现一个新设备。这就是要使用的设备。为进一步确保安全，请检查报告的大小是否 与 USB 闪存盘相符。

2.2.3 macOS 操作说明

打开终端（可在 Spotlight 中搜索 Terminal）。

使用 `hdiutil` 的 convert 选项将 `.iso` 文件转换为 `.dmg` 格式，例如：

```
# hdiutil convert pxvirt_*.iso -format UDRW -o pxvirt_*.dmg
```

> **Tip**
> macOS 通常会自动在输出文件名后添加 `.dmg`。

运行以下命令获取当前设备列表：

```
# diskutil list
```

现在插入 USB 闪存盘，并再次运行该命令，以确定系统为其分配了哪个设备节点（例如 /dev/diskX）

```
# diskutil list
# diskutil unmountDisk /dev/diskX
```

> **Note**
> 请将 X 替换为上一条命令显示的磁盘编号。

```
# sudo dd if=pxvirt_*.dmg bs=1M of=/dev/rdiskX
```

> **Note**
> 上一条命令中有意使用 `rdiskX` 而不是 `diskX`，这样可以提升写入速度。

2.2.4 Windows 操作说明

使用 **Etcher**

Etcher 可直接使用。请从 https://etcher.io 下载 Etcher。它会引导你完成选择 ISO 和 USB 闪存盘的过程。

使用 **Rufus**

Rufus 是更轻量的替代工具，但需要使用 **DD mode** 才能正常工作。请从 https://rufus.ie/ 下载 Rufus。可以安装它，也可以使用便携版。选择目标驱动器和 Proxmox VE ISO 文件。

Important
点击 Start 后，如果对话框询问是否下载其他版本的 GRUB，必须点击 No。在下一个对话框中选择 DD 模式。

2.3 使用 Proxmox VE 安装程序

安装程序 ISO 镜像包含以下内容：

- 完整操作系统（OpenEuler Linux，64 位）
- Proxmox VE 安装程序，可使用 ext4、XFS、BTRFS（技术预览）或 ZFS 对本地磁盘进行 分区，并
- 支持 KVM 和 LXC 的 Proxmox VE Linux 内核
- 用于管理虚拟机、容器、主机系统、集群和所有必要资源的完整工具集
- 基于 Web 的管理界面

Note
安装过程中会移除所选驱动器上的所有现有数据。安装程序不会为其他操作系统添加启动菜单项。

请插入 准备好的安装介质（例如 USB 闪存盘或 CD-ROM），并从该介质启动。

Tip
请确保服务器固件设置中已启用从安装介质（例如 USB）启动。启动 Proxmox VE 8.1 之前版本的安装程序时，需要禁用 Secure Boot。

选择正确的启动项后（例如 *Boot from USB*），会显示 Proxmox VE 菜单，并可选择以下 选项之一：

Install Proxmox VE (Graphical)
启动普通图形安装。

Tip
可以只使用键盘操作安装向导。按下 ALT 键并结合相应按钮中带下划线的字符，即可点击按钮。例如，按 ALT + N 可点击 Next 按钮。

Install Proxmox VE (Terminal UI)
启动终端模式安装向导。它提供与图形安装程序总体相同的安装体验，但通常对非常旧 或非常新的

Install Proxmox VE (Terminal UI, Serial Console)
启动终端模式安装向导，并额外设置 Linux 内核使用机器的（第一个）串口进行输入和 输出。该选

两种模式在实际安装流程中使用同一套代码库，从而受益于十余年的问题修复，并确保 功能一致。

Tip
如果图形安装程序因驱动等问题无法正常工作，可以使用 *Terminal UI* 选项。 另请参阅 添加 nomodeset 内核参数。

Advanced Options: Install Proxmox VE (Graphical, Debug Mode)
以调试模式启动安装。在若干安装步骤中会打开控制台。如果出现问题，这有助于调试 现场。要退 CTRL-D。该选项可用于启动一个包含所有基础工具的 live 系统。例如，可用于 修复降级的 ZFS rpool，或修复现有 Proxmox VE 设置的 引导加载器。

Advanced Options: Install Proxmox VE (Terminal UI, Debug Mode)
与图形调试模式相同，但会准备系统以运行基于终端的安装程序。

Advanced Options: Install Proxmox VE (Serial Console Debug Mode)
与基于终端的调试模式相同，但会额外设置 Linux 内核使用机器的（第一个）串口进行 输入和输出

Advanced Options: Install Proxmox VE (Automated)
以无人值守模式启动安装程序，即使 ISO 尚未为自动化安装正确准备。该选项可用于 收集硬件详情 无人值守安装。

Advanced Options: Rescue Boot
通过该选项可以启动现有安装。它会搜索所有连接的硬盘。如果找到现有安装，则使用 ISO 中的 Linux 内核直接启动到该磁盘。当引导加载器（GRUB/`systemd-boot`）存在 问题，或 BIOS/UEFI 无法从磁盘读取引导块时，该选项会很有用。

Advanced Options: Test Memory (memtest86+)
运行 `memtest86+`。这有助于检查内存是否可用且没有错误。要运行该选项，必须在 UEFI 固件设置工具中关闭 Secure Boot。

通常选择 **Install Proxmox VE (Graphical)** 来开始安装。

第一步是阅读 EULA（End User License Agreement）。随后，可以选择安装目标硬盘。

Caution
默认情况下会使用整台服务器，并移除所有现有数据。继续安装前，请确保 服务器上没有重要数据。

`Options` 按钮可用于选择目标文件系统，默认为 `ext4`。如果选择 `ext4` 或 `xfs` 作为文件系统，安装程序 LVM，并提供额外选项来限制 LVM 空间（参见 下文）。

Proxmox VE 也可以安装在 ZFS 上。由于 ZFS 提供多种软件 RAID 级别，对于没有硬件 RAID 控制器的系统，这是一种可选方案。必须在 `Options` 对话框中选择目标磁盘。更多 ZFS 专用设置可在 Advanced Options 中更改。

Warning
不支持在任何硬件 RAID 之上使用 ZFS，这可能导致数据丢失。

下一页会要求填写基础配置选项，例如位置、时区和键盘布局。位置信息用于选择邻近 下载服务器，以
检测失败，或需要使用所在国家不常见的键盘布局时，才需要手动修改。

接下来需要指定超级用户（`root`）的密码和电子邮件地址。密码至少需要包含 8 个 字符。强烈建议使用

- 密码长度至少 12 个字符。
- 包含小写字母、大写字母、数字和符号。
- 避免字符重复、键盘模式、常见词典词汇、字母或数字序列、用户名、亲属或宠物 名称、恋爱关系相
  姓名或日期）。

电子邮件地址用于向系统管理员发送通知。例如：

- 可用软件包更新的信息。
- 周期性 *cron* 任务产生的错误消息。

所有这些通知邮件都会发送到指定电子邮件地址。

最后一步是网络配置。在下拉菜单中，状态为 *UP* 的网络接口名称前会显示实心圆。 请注意，安装期间
IPv4 或 IPv6 地址之一，不能同时指定二者。要配置双栈 节点，请在安装完成后添加额外 IP 地址。

下一步会显示此前所选选项的摘要。请重新检查每项设置；如果需要更改某项设置， 请使用 `Previous`
按钮。

点击 `Install` 后，安装程序会开始格式化磁盘，并将软件包复制到目标磁盘。请等待 该步骤完成；随
复制软件包通常需要几分钟，主要取决于安装介质速度和目标磁盘性能。

复制并设置软件包完成后，可以重启服务器。默认情况下，系统会在几秒后自动完成该 操作。

安装失败

如果安装失败，请在第二个 TTY（*CTRL* + *ALT* + *F2*）上查看具体错误，并确保系统 满足 最低要求。

如果安装仍无法正常进行，请查看 如何获取帮助章节。

2.3.1 安装后访问管理界面

成功安装并重启系统后，可以使用 Proxmox VE Web 界面进行进一步配置。

1. 在浏览器中访问安装期间指定的 IP 地址和端口 8006，例如： https://youripaddress:8006
2. 使用 `root`（realm *PAM*）用户名和安装期间选择的密码登录。
3. 检查 IP 配置和主机名。
4. 检查时区。
5. 检查 防火墙设置。

常用命令示例

```
# pveversion -v
# systemctl status pveproxy.service
# ip address
```

2.3.2 高级 LVM 配置选项

如果使用 ext4 或 xfs，安装程序会创建名为 pve 的卷组（VG），以及名为 root、data 和 swap 的逻辑卷（LV）。要控制这些卷的大小，请使用：

hdsize
定义要使用的硬盘总大小。这样可以在硬盘上保留空闲空间用于进一步分区（例如在 同一块硬盘上
PV 和 VG，用作 LVM 存储）。

swapsize
定义 swap 卷大小。默认值为已安装内存大小，最小 4 GB，最大 8 GB。最终值不能 大于 hdsize/8。

> **Note**
> 如果设置为 0，则不会创建 swap 卷。

maxroot
定义 root 卷的最大大小，该卷用于存储操作系统。root 卷大小上限为 hdsize/4。

maxvz
定义 data 卷的最大大小。data 卷的实际大小为：

```
datasize = hdsize - rootsize - swapsize - minfree
```

其中 datasize 不能大于 maxvz。

> **Note**
> 对于 LVM thin，仅当 datasize 大于 4GB 时才会创建 data 池。

> **Note**
> 如果设置为 0，则不会创建 data 卷，存储配置也会相应调整。

minfree
定义应在 LVM 卷组 pve 中保留的空闲空间量。当可用存储超过 128GB 时，默认值为 16GB；否则
hdsize/8。

> **Note**
> LVM 需要 VG 中有空闲空间才能创建快照（lvmthin 快照不需要）。

2.3.3 高级 ZFS 配置选项

如果使用 ZFS，安装程序会创建 ZFS 池 rpool。不会创建 swap 空间，但可以在安装 磁盘上为 swap 保留一些未分区空间。也可以在安装后创建 swap zvol，不过这可能会 带来问题（参见 ZFS swap notes）。

ashift
: 定义所创建池的 ashift 值。ashift 至少需要设置为底层磁盘的扇区大小（2 的 ashift 次方即扇区大小），或未来可能加入该池的任何磁盘的扇区大小（例如替换 故障磁盘时使用的磁盘

compress
: 定义是否为 rpool 启用压缩。

checksum
: 定义 rpool 应使用哪种校验和算法。

copies
: 定义 rpool 的 copies 参数。请查看 zfs(8) 手册页了解其语义，以及为什么它 不能替代磁盘级

ARC max size
: 定义 ARC 可增长到的最大大小，从而限制 ZFS 可使用的内存量。更多细节另请参阅 如何限制 ZFS 内存使用章节。

hdsize
: 定义要使用的硬盘总大小。这有助于在硬盘上保留空闲空间用于进一步分区（例如创建 swap 分区）。hdsize 仅对可启动磁盘生效，也就是 RAID0、RAID1 或 RAID10 中的 第一块磁盘或镜 RAID-Z[123] 中的所有磁盘。

2.3.4 高级 BTRFS 配置选项

使用 BTRFS 时不会创建 swap 空间，但可以在安装磁盘上为 swap 保留一些未分区空间。 可以创建独立 子卷，或使用 `btrfs filesystem mkswapfile` 命令创建 swapfile。

compress
: 定义是否为 BTRFS 子卷启用压缩。支持不同压缩算法：*on*（等同于 *zlib*）、 *zlib*、*lzo* 和 *zstd*。默认为 *off*。

hdsize
: 定义要使用的硬盘总大小。这有助于在硬盘上保留空闲空间用于进一步分区（例如创建 swap 分区）。

2.3.5 ZFS 性能提示

ZFS 在内存充足时效果最佳。如果计划使用 ZFS，请确保有足够 RAM 可用。一个较好 的估算方式是 4GB 基础内存，加上每 TB RAW 磁盘空间 1GB RAM。

ZFS 可以使用专用驱动器作为写缓存，称为 ZFS Intent Log（ZIL）。应使用快速驱动器（SSD）承载

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

2.3.6 添加 nomodeset 内核参数

在非常旧或非常新的硬件上，图形驱动可能导致问题。如果安装在启动期间卡住，可以 尝试添加 nomod 参数。该参数会阻止 Linux 内核加载任何图形驱动，并强制其 继续使用 BIOS/UEFI 提供的 frame-buffer。

在 Proxmox VE 引导加载器菜单中，导航到 *Install Proxmox VE (Terminal UI)*，按 e 编辑该条目。使用方向键导航到以 `linux` 开头的行，将光标移到该行末尾，并添加 `nomodeset` 参数，需与原

然后按 `Ctrl-X` 或 `F10` 使用该配置启动。

2.4 无人值守安装

自动化安装方法允许以无人值守方式安装 Proxmox VE。这使你可以在裸机上完全自动化设置 流程。安 Ansible 等自动化工具进一步配置该安装。

必须在应答文件中提供安装程序所需选项。该文件允许使用过滤规则来确定应使用哪些 磁盘和网卡。

要使用自动化安装，首先需要选择应答文件的获取来源，然后根据该选择准备安装 ISO。

ISO 准备完成后，其初始启动菜单会显示一个名为 *Automated Installation* 的新启动 项，并在 10 秒超时后自动选择该项。

有关无人值守安装的更多细节和信息，请 访问我们的 wiki。

2.5 在 OpenEuler 上安装 Proxmox VE

Proxmox VE 以一组 OpenEuler 软件包的形式发布，可以安装在标准 OpenEuler 安装之上。配置仓库后需要运行以下命令：

```
# dnf makecache
# dnf install proxmox-ve
```

在现有 OpenEuler 安装之上安装看似简单，但前提是基础系统已经正确安装，并且你清楚 如何配置和

一般来说，这并不简单，尤其是在使用 LVM 或 ZFS 时。

详细的逐步操作指南可在

https://docs.pxvirt.lierfang.com/zh/installfromopeneuler.html

2.6 另请参阅

- Prepare Installation Media
- Install Proxmox VE on OpenEuler 12 Bookworm
- System Requirements
- Package Repositories

- Host System Administration
- Network Configuration
- Installation: Tips and Tricks

Chapter 3

主机系统管理

以下章节将重点介绍常见虚拟化任务，并说明 Proxmox VE 在主机管理和运维方面的特定内容。

Pxvirt 基于 OpenEuler GNU/Linux，并通过额外仓库提供 Proxmox VE 相关软件包。这意味着可以使
OpenEuler 软件包体系，包括安全更新和错误修复。

Pxvirt提供基于 OpenEuler kernel 的自有 Linux kernel，并启用了所有必要的虚拟化和容器功能，
同时包含 ZFS 以及若干额外硬件驱动。

对于以下章节未涵盖的其他主题，请参考 OpenEuler 文档。在线版

https://docs.openeuler.org/zh/

3.1 软件包仓库

Proxmox VE 基于 openEuler，使用 RPM 软件包格式，并通过 `dnf` 管理软件包和仓库。

Proxmox VE 每天自动检查软件包更新。`root@pam` 用户会通过电子邮件收到可用更新通知。 在
GUI 中，可以使用 *Changelog* 按钮查看所选更新的更多详细信息。

3.1.1 Proxmox VE 中的软件仓库

仓库是一组软件包集合，可用于安装新软件，也是获取安全更新、错误修复和新功能的重要来源。

Note
需要配置有效的 openEuler 基础仓库和 PXVIRT 仓库，才能获得完整的软件包依赖和更新。

DNF 仓库定义在 `/etc/yum.repos.d/` 目录下的 `.repo` 文件中。每个仓库通常包含 `baseurl`、`enab`
等字段。

仓库管理

在 Web 界面中，可以通过节点的更新相关面板查看当前仓库状态和可用更新。 也可以使用命令行工具
`dnf` 管理仓库和执行系统更新。

3.1.2 PXVIRT 仓库

PXVIRT 软件包通过梨儿方 PXVIRT 仓库发布。请在每个 Proxmox VE 节点上创建 /etc/yum.repos.

文件 **/etc/yum.repos.d/pxvirt.repo**

```
[pxvirt]
name=Lierfang PxVirt Repo
baseurl=https://mirrors.lierfang.com/pxcloud/pxvirt/repo/openeuler/24.03/ ↩
   $basearch
enabled=1
gpgcheck=1
gpgkey=https://mirrors.lierfang.com/pxcloud/lierfang.gpg
```

也可以使用编辑器创建该文件：

```
# editor /etc/yum.repos.d/pxvirt.repo
[pxvirt]
name=Lierfang PxVirt Repo
baseurl=https://mirrors.lierfang.com/pxcloud/pxvirt/repo/openeuler/24.03/ ↩
   $basearch
enabled=1
gpgcheck=1
gpgkey=https://mirrors.lierfang.com/pxcloud/lierfang.gpg
```

配置完成后，刷新软件包元数据：

```
# chmod 0600 /etc/yum.repos.d/pxvirt.repo
# dnf makecache
```

如果需要确认仓库是否已经启用，可以运行：

```
# dnf repolist
```

3.1.3 Ceph 软件包仓库

PXVIRT 的 openEuler 版本不使用 Proxmox 的独立 Ceph 仓库。 不要沿用 Proxmox 文档中的 enterprise、no-subscription 或 test Ceph 仓库地址。

如需安装或更新 Ceph 相关软件包，请使用已配置的 openEuler 基础仓库和 PXVIRT 仓库。 如果使用外 Ceph 集群，只需要按存储章节配置客户端连接信息。

3.1.4 软件包签名验证

PXVIRT 仓库启用了 gpgcheck=1，DNF 会使用 gpgkey 指定的 GPG 公钥验证软件包签名。 如果密钥尚未导入，首次安装或更新软件包时，DNF 会提示导入该密钥。

也可以手动导入仓库密钥：

```
# rpm --import https://mirrors.lierfang.com/pxcloud/lierfang.gpg
```

ßß ==== 常用命令示例 ß 以下命令可用于刷新软件包索引、查看可升级软件包，并检查 Proxmox VE 软件包版本：

```
# dnf makecache
# dnf check-update
# pveversion -v
```

安装所有可用更新：

```
# dnf update
```

3.2 系统软件更新

Proxmox 会定期为所有软件仓库提供更新。要安装更新，可以使用基于 Web 的 GUI，或使用以下 CLI 命令：

```
# dnf makecache
# dnf update
```

Note
DNF 包管理系统非常灵活，并提供许多功能；更多信息请参见 `man dnf`。

Tip
定期更新对于获取最新补丁和安全相关修复至关重要。重大系统升级会在 Pxvirt Community Forum 中公告。

3.3 固件更新

在裸金属服务器上运行 Proxmox VE 时，应应用本章所述的固件更新。是否适合在客户机内部配置固件（例如使用设备直通时）高度依赖具体部署，因此不在本章讨论范围内。

除了常规软件更新，固件更新对于可靠、安全的运行同样重要。

在获取并应用固件更新时，建议结合使用可用的多种方法，以便尽早获得更新，或确保能够获得更新。

从术语上看，firmware 通常分为微码（用于 CPU）和固件（用于其他设备）。

3.3.1 持久化固件

本节适用于所有设备。更新后的微码通常包含在 BIOS/UEFI 更新中，并存储在主板上；其他固件则存储
CPU 尤其重要，因为它允许在启动时尽早按常规流程加载更新后的微码。

Caution
某些更新（例如 BIOS/UEFI 或存储控制器更新）可能会重置设备配置。请仔细遵循厂商说明，

请向厂商确认可用的更新方法。

- 服务器的便捷更新方法可能包括 Dell 的 Lifecycle Manager 或 HPE 的 Service Packs。
- 有时也可以使用 Linux 实用工具。例如，NVIDIA ConnectX 可使用 *mlxup*，Broadcom 网卡可使用 *bnxtnvm/niccli*。
- 如果正在使用的硬件厂商与 LVFS 合作，并且硬件属于 受支持硬件，那么 LVFS 也是一种选择。其技术要求是系统制造于 2014 年之后，并通过 UEFI 启动。

在 openEuler 中，可以通过 `fwupd` 软件包使用 LVFS 提供的固件更新。 如果系统未能自动识别 EFI 分区位置，可以在 `/etc/fwupd/daemon.conf` 中显式配置正确的挂载点，例如：

文件 **/etc/fwupd/daemon.conf**

```
# Override the location used for the EFI system partition (ESP) path.
EspLocation=/boot/efi
```

Tip
如果更新说明要求重启主机，请确保可以安全执行。另请参见 节点维护。

3.3.2 运行时固件文件

此方法将固件存储在 Proxmox VE 操作系统中；如果设备的 持久化固件版本较旧，则会将该固件传递给
该方法受网络卡、显卡等设备支持，但不适用于依赖持久化固件的设备，例如主板和硬盘。

在 Proxmox VE 中，常见硬件所需的运行时固件由 openEuler 的 `linux-firmware` 软件包提供。因此
系统更新， 常见硬件所包含的固件会自动保持最新。

如需额外固件，请确认已经启用相应的 openEuler 软件源。

如果尝试安装额外固件软件包但发生冲突，DNF 将中止安装。特定固件也许可以通过其他方式获取。

3.3.3 CPU 微码更新

微码更新用于修复已发现的安全漏洞和其他严重 CPU 缺陷。虽然 CPU 性能可能受到影响，但已打补丁
根据 CPU 类型不同，如果不有意让 CPU 运行在不安全状态，可能无法再达到存在缺陷的出厂状态下的

要查看当前 CPU 漏洞及其缓解措施概览，请运行 `lscpu`。只有当 Proxmox VE 主机 保持最新、版本尚
结束生命周期，并且自上次内核更新以来至少重启过一次时， 当前真实世界中已知的漏洞才会显示出来

除了推荐通过 持久化 BIOS/UEFI 更新来更新微码之外， 还可以使用一种独立方式：早期 **OS** 微码更新。
在主板厂商不再提供 BIOS/UEFI 更新时也很有帮助。无论使用哪种方法，应用微码更新都始终需要重启

设置早期 **OS** 微码更新

要设置由 Linux 内核在启动早期应用的微码更新，需要：

1. 确认已经启用相应的 openEuler 软件源
2. 获取最新可用软件包：`dnf makecache`（也可以使用 Web 界面中的 Node → Updates）
3. 安装 CPU 微码软件包：`dnf install microcode_ctl`
4. 重启 Proxmox VE 主机

之后的任何微码更新也都需要重启后才能加载。

微码版本

要获取当前正在运行的微码修订版以便比较或调试：

```
# grep microcode /proc/cpuinfo | uniq
microcode       : 0xf0
```

一个微码软件包包含许多不同 CPU 的更新。但专门适用于你的 CPU 的更新可能并不频繁。 因此，仅查
CPU 发布了更新。

如果已安装新的微码软件包并重启 Proxmox VE 主机，且这个新微码同时比 CPU 内置版本和主板固件
系统日志中会出现 ”microcode updated early” 消息。

```
# dmesg | grep microcode
[    0.000000] microcode: microcode updated early to revision 0xf0, date =  ↩
   2021-11-12
[    0.896580] microcode: Microcode Update Driver: v2.2.
```

故障排查

出于调试目的，可以按如下方式临时禁用系统启动时常规应用的早期 OS 微码更新：

1. 确保主机可以 安全重启
2. 重启主机以进入 GRUB 菜单（如果菜单隐藏，请按住 SHIFT）
3. 在所需的 Proxmox VE 启动项上按 E
4. 转到以 `linux` 开头的行，并以空格分隔追加 **`dis_ucode_ldr`**
5. 按 CTRL-X，本次启动将不使用早期 OS 微码更新

如果怀疑问题与最近的微码更新有关，应考虑软件包降级，而不是移除软件包（`dnf remove microc
持久化微码，即使较新的微码本可以正常运行。

如果 openEuler 软件源中存在较早版本的微码软件包，则可以降级，如以下示例所示：

```
# dnf --showduplicates list microcode_ctl
Installed Packages
microcode_ctl.x86_64 4:2.1-53.oe2203sp4 @OS
Available Packages
microcode_ctl.x86_64 4:2.1-51.oe2203sp4 OS
```

```
# dnf downgrade microcode_ctl
...
Downgrading:
 microcode_ctl x86_64 4:2.1-51.oe2203sp4 OS
...
Complete!
...
```

再次确认主机可以 安全重启。要应用该微码软件包中可能包含的、 适用于你 CPU 类型的较旧微码，请

Tip
将降级后的软件包保持一段时间，然后稍后再尝试更新版本，是合理的做法。即使未来软件包版本相
期间的系统更新也可能已经修复曾遇到的问题。

```
# dnf versionlock add microcode_ctl
```

```
# dnf versionlock delete microcode_ctl
# dnf makecache
# dnf update
```

3.3.4 常用命令示例

查看 CPU 漏洞和缓解状态概览：

```
# lscpu
```

查看当前正在运行的微码修订版：

```
# grep microcode /proc/cpuinfo | uniq
```

3.4 网络配置

Proxmox VE 使用 Linux 网络栈。这使 Proxmox VE 节点上的网络设置具备很高的灵活性。可以通过 GUI 完成配置，也可以手动编辑包含完整网络配置的 `/etc/network/interfaces` 文件。`interfac` 手册页包含完整的格式说明。所有 Proxmox VE 工具都会尽量保留用户的直接修改，但仍然更建议使用 GUI，因为它可以帮助避免错误。

需要使用 Linux bridge 接口（通常称为 *vmbrX*）将客户机连接到底层物理网络。可以将其理解为一个 bond 实现冗余、使用 vlans，或采用 routed 与 NAT 配置。

Software Defined Network 可用于 Proxmox VE 集群中更复杂的虚拟网络。

Warning
如果不确定其影响，不建议使用传统 Debian 工具 ifup 和 ifdown，因为它们存在一些容易踩到的问题。例如执行 ifdown vmbrX 会中断所有客户机流量，但之后对同一 bridge 执行 ifup 时不会重新连接这些客户机。

3.4.1 应用网络更改

Proxmox VE 不会将更改直接写入 /etc/network/interfaces。相反，系统会写入名为 /etc/net 的临时文件，这样可以一次完成多项相关更改。这也允许在应用前确认更改是否正确，因为错误的网络

使用 **ifupdown2** 实时重新加载网络

使用推荐的 *ifupdown2* 软件包（自 Proxmox VE 7.0 起为新安装的默认选择），可以在不重启的情况下 GUI 修改网络配置，可以点击 *Apply Configuration* 按钮。该操作会将更改从暂存的 interfaces.ne 文件移动到 /etc/network/interfaces，并实时应用。

如果直接手动修改了 /etc/network/interfaces 文件，可以运行 ifreload -a 来应用这些更改。

常用命令示例

查看当前网络接口配置：

```
ip address show
```

在使用 ifupdown2 的系统上实时重新加载网络配置：

```
ifreload -a
```

重启节点以应用

应用新网络配置的另一种方式是重启节点。在这种情况下，systemd 服务 pvenetcommit 会在 networking 服务应用配置前，先激活暂存的 interfaces.new 文件。

3.4.2 命名约定

当前设备名称使用以下命名约定：

- Ethernet 设备：en*，即 systemd 网络接口名称。自 5.0 版本起，新的 Proxmox VE 安装使用此命
- Ethernet 设备：eth[N]，其中 0 ≤ N（eth0、eth1 等）。该命名方案用于 5.0 发布前安装的 Proxmox VE 主机。升级到 5.0 时，名称会保持不变。
- Bridge 名称：通常为 vmbr[N]，其中 0 ≤ N ≤ 4094（vmbr0 - vmbr4094），但也可以使用任意以 10 个字符的字母数字字符串。
- Bonds：bond[N]，其中 0 ≤ N（bond0、bond1 等）。

- VLAN：直接在设备名称后追加 VLAN 编号，并用句点分隔（`eno1.50`、`bond1.30`）。

这样更容易调试网络问题，因为设备名称能够体现设备类型。

Systemd 网络接口名称

Systemd 为网络设备名称定义了带版本的命名方案。该方案对 Ethernet 网络设备使用两个字符的前缀 en。后续字符取决于设备驱动、设备位置和其他属性。可能的模式包括：

- `o<index>[n<phys_port_name>|d<dev_port>]` — 板载设备
- `s<slot>[f<function>][n<phys_port_name>|d<dev_port>]` — 按热插拔 ID 标识的设备
- `[P<domain>]p<bus>s<slot>[f<function>][n<phys_port_name>|d<dev_port>]` — 按总线 ID 标识的设备
- `x<MAC>` — 按 MAC 地址标识的设备

以下是最常见模式的一些示例：

- `eno1` — 第一块板载 NIC
- `enp3s0f1` — PCI 总线 3、插槽 0 上 NIC 的功能 1

有关所有可能设备名称模式的完整列表，请参见 systemd.net-naming-scheme(7) 手册页。

新版本的 systemd 可能会定义新版网络设备命名方案，并默认使用该方案。因此，更新到较新的 systemd 版本时，例如进行 Proxmox VE 大版本升级期间，网络设备名称可能发生变化，并需要调整网 下文）。

但是，即使命名方案版本已固定，网络设备名称仍可能因内核或驱动更新而变化。要彻底避免特定网络 link 文件手动覆盖其名称（见 下文）。

有关网络接口名称的更多信息，请参见 Predictable Network Interface Names。

固定特定命名方案版本

可以通过向 内核命令行 添加 `net.naming-scheme=<version>` 参数，固定网络设备命名方案的特定 systemd.net-naming-scheme(7) 手册页。

例如，要固定版本 v252（这是全新安装 Proxmox VE 8.0 时使用的最新命名方案版本），请添加以下内

```
net.naming-scheme=v252
```

另请参见 本节，了解如何编辑内核命令行。需要重启后更改才会生效。

覆盖网络设备名称

Proxmox VE 提供了一个工具，可自动生成用于覆盖网络设备名称的 .link 文件。它还会自动替换以下文

- /etc/network/interfaces
- /etc/pve/nodes/<nodename>/host.fw
- /etc/pve/sdn/controllers.cfg

Note
由于生成的映射只属于生成它的本地节点，Firewall Datacenter 配置（/etc/pve/firewall/cluster.fw）中包含的接口名称不会自动更新。

生成的 link 文件存放在 /usr/local/lib/systemd/network 中。对于配置文件，会在相同位置生成 .new 后缀的新文件。这样可以使用 diff（或你选择的其他 diff 查看器）检查对配置所做的更改：

```
diff -y /etc/network/interfaces /etc/network/interfaces.new
```

如果发现任何有问题的更改，或希望在重启前撤销 pinning 工具所做的更改，只需删除所有 .new 文件以及 /usr/local/lib/systemd/network 中对应的 link 文件。

以下命令会为尚未拥有 .link 文件的所有物理网络接口生成 .link 文件，并更新选定的 Proxmox VE 配置文件（见上文）。生成的名称将使用默认前缀 nic，因此得到的接口名称会是 nic1、nic2 等。

```
pve-network-interface-pinning generate
```

可以使用 --prefix 标志覆盖默认前缀：

```
pve-network-interface-pinning generate --prefix myprefix
```

也可以只固定特定接口：

```
pve-network-interface-pinning generate --interface enp1s0
```

固定特定接口时，可以指定该接口应固定为的精确名称：

```
pve-network-interface-pinning generate --interface enp1s0 --target-name ↩
   if42
```

要将 pve-network-interface-pinning 所做更改应用到网络配置，需要重启节点。

可以使用自定义 systemd.link 文件为特定网络设备手动分配名称。这会覆盖按最新网络设备命名方案应

自定义 link 文件应放在 /etc/systemd/network/ 中，并命名为 <n>-<id>.link，其中 n 是小于 99 的优先级，id 是某个标识符。link 文件包含两个小节：[Match] 决定该文件应用到哪些接口，决定这些接口应如何配置，包括其命名。

要为特定网络设备分配名称，需要在 [Match] 小节中以唯一且持久的方式标识该设备。一种做法是使用 MACAddress 选项匹配设备的 MAC 地址，因为它通常不会变化。

[Match] 小节还应包含 Type 选项，以确保只匹配预期的物理接口，而不是具有相同 MAC 地址的 bridge/bond/VLAN 接口。在大多数配置中，Type 应设置为 ether，以便只匹配 Ethernet 设备，但其 systemd.link(5) 手册页。

然后，可以在 [Link] 小节中使用 Name 选项分配名称。

link 文件会被复制到 initramfs，因此建议在添加、修改或移除 link 文件后刷新 initramfs：

```
# update-initramfs -u -k all
```

例如，要将名称 enwan0 分配给 MAC 地址为 aa:bb:cc:dd:ee:ff 的 Ethernet 设备，请创建文件 /etc/systemd/network/10-enwan0.link，内容如下：

```
[Match]
MACAddress=aa:bb:cc:dd:ee:ff
Type=ether

[Link]
Name=enwan0
```

不要忘记调整 /etc/network/interfaces 以使用新名称，并按上文所述刷新 initramfs。需要重

Note
建议分配以 en 或 eth 开头的名称，使 Proxmox VE 能够将该接口识别为物理网络设备，并通过 GUI 进行配置。同时，应确保该名称未来不会与其他接口名称冲突。一种做法是分配一个不匹配 systemd 网络接口任何命名模式的名称（见上文），例如上例中的 enwan0。

有关 link 文件的更多信息，请参见 systemd.link(5) 手册页。

3.4.3 选择网络配置

可以根据当前网络组织方式和可用资源，选择 bridged、routed 或 masquerading 网络配置。

位于私有 **LAN** 中、通过外部网关访问互联网的 **Proxmox VE** 服务器

在这种情况下，**Bridged** 模型最合适，这也是新安装 Proxmox VE 的默认模式。每个客户机系统都会有 Proxmox VE bridge。其效果类似于将客户机网卡直接连接到 LAN 中的一台新交换机，而 Proxmox VE 主机承担该交换机的角色。

位于托管服务提供商处、为客户机提供公网 **IP** 段的 **Proxmox VE** 服务器

对于这种配置，可以根据提供商允许的方式使用 **Bridged** 或 **Routed** 模型。

位于托管服务提供商处、只有单个公网 **IP** 地址的 **Proxmox VE** 服务器

在这种情况下，要让客户机系统访问外部网络，唯一方式是使用 **Masquerading**。如果需要外部访问 **Port Forwarding**。

为了获得更高灵活性，可以配置 VLAN（IEEE 802.1q）和网络 bonding，也称为 "link aggregation"。这样可以构建复杂且灵活的虚拟网络。

3.4.4 使用 **Bridge** 的默认配置

Bridge 类似于用软件实现的物理网络交换机。所有虚拟客户机可以共享一个 bridge，也可以创建多个 bridge 来隔离网络域。每台主机最多可以拥有 4094 个 bridge。

安装程序会创建一个名为 vmbr0 的 bridge，并将其连接到第一块 Ethernet 网卡。/etc/network/i 中的对应配置可能如下：

```
auto lo
iface lo inet loopback

iface eno1 inet manual

auto vmbr0
iface vmbr0 inet static
        address 192.168.10.2/24
        gateway 192.168.10.1
        bridge-ports eno1
        bridge-stp off
        bridge-fd 0
```

虚拟机的行为就像直接连接到物理网络一样。反过来，网络会将每台虚拟机视为拥有自己的 MAC，即使

3.4.5 路由配置

大多数托管服务提供商不支持上述配置。出于安全原因，一旦检测到单个接口上存在多个 MAC 地址，它

Tip
某些提供商允许通过其管理界面注册额外 MAC。这可以避免该问题，但配置起来可能比较繁琐，因为 MAC。

可以通过将所有流量经由单个接口“路由”来避免该问题。这可以确保所有网络数据包都使用同一个 MAC 地址。

常见场景是拥有一个公网 IP（本例假设为 198.51.100.5），并为虚拟机拥有一个额外 IP 段（203.0

```
auto lo
iface lo inet loopback

auto eno0
iface eno0 inet static
        address  198.51.100.5/29
        gateway  198.51.100.1
        post-up echo 1 > /proc/sys/net/ipv4/ip_forward
        post-up echo 1 > /proc/sys/net/ipv4/conf/eno0/proxy_arp


auto vmbr0
iface vmbr0 inet static
        address  203.0.113.17/28
        bridge-ports none
```

```
        bridge-stp off
        bridge-fd 0
```

3.4.6 使用 iptables 的 Masquerading（NAT）

Masquerading 允许只有私有 IP 地址的客户机通过主机 IP 地址作为出站流量来源来访问网络。每个出
iptables 重写，使其看起来像是源自主机；响应也会相应重写，以便路由回原始发送方。

```
auto lo
iface lo inet loopback

auto eno1
#real IP address
iface eno1 inet static
        address  198.51.100.5/24
        gateway  198.51.100.1

auto vmbr0
#private sub network
iface vmbr0 inet static
        address  10.10.10.1/24
        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
```

> **Note**
> 在某些启用了防火墙的 masquerade 配置中，出站连接可能需要 conntrack zones。否则防火墙可能阻止出站连接，因为它们会优先匹配虚拟机 bridge 的 POSTROUTING，而不是 MASQUERADE。

在 /etc/network/interfaces 中添加以下行可以修复此问题：

```
post-up   iptables -t raw -I PREROUTING -i fwbr+ -j CT --zone 1
post-down iptables -t raw -D PREROUTING -i fwbr+ -j CT --zone 1
```

有关此问题的更多信息，请参考以下链接：

Netfilter Packet Flow

Patch on netdev-list introducing conntrack zones

Blog post with a good explanation by using TRACE in the raw table

3.4.7 Linux Bond

Bonding（也称为 NIC teaming 或 Link Aggregation）是一种将多个 NIC 绑定为单个网络设备的技术

Fibre Channel 等高速硬件及其配套交换硬件可能非常昂贵。通过链路聚合，两块 NIC 可以呈现为一个
Linux 内核原生功能，并受大多数交换机支持。如果节点有多个 Ethernet 端口，可以将网络线缆连接到
连接会故障转移到其中一条线缆。

聚合链路可以降低在线迁移延迟，并提高 Proxmox VE 集群节点之间的数据复制速度。

Bonding 有 7 种模式：

- **Round-robin (balance-rr):** 按顺序从第一个可用的从属网络接口（NIC）到最后一个接口传输网
- **Active-backup (active-backup):** bond 中只有一个从属 NIC 处于活动状态。仅当活动从属接口
  bonded 接口的 MAC 地址在外部只会出现在一个 NIC（端口）上，以避免网络交换机产生混乱。该模
- **XOR (balance-xor):** 根据 [(源 MAC 地址 XOR 目标 MAC 地址) modulo 从属 NIC 数量]
  传输网络数据包。该模式会为每个目标 MAC 地址选择相同的从属 NIC，提供负载均衡和容错能力。
- **Broadcast (broadcast):** 在所有从属网络接口上传输网络数据包。该模式提供容错能力。
- **IEEE 802.3ad Dynamic link aggregation (802.3ad)(LACP):** 创建共享相同速率和双工设置
  802.3ad 规范使用活动聚合组中的所有从属网络接口。
- **Adaptive transmit load balancing (balance-tlb):** Linux bonding 驱动模式，不需要网络交
  MAC 地址。
- **Adaptive load balancing (balance-alb):** 包含 balance-tlb，并为 IPV4 流量提供接收负载均
  ARP 协商实现。bonding 驱动会拦截本地系统发出的 ARP Replies，并将源硬件地址重写为单个逻辑
  bonded 接口中某个从属 NIC 的唯一硬件地址，使不同网络对端使用不同 MAC 地址发送其网络数据包

如果交换机支持 LACP（IEEE 802.3ad）协议，建议使用对应的 bonding 模式（802.3ad）。否则，通
active-backup 模式。

对于集群网络（Corosync），建议配置多个网络。Corosync 不需要通过 bond 实现网络冗余，因为当

以下 bond 配置可用作分布式/共享存储网络。其优势是可以获得更高速度，并让网络具备容错能力。

示例：使用具有固定 **IP** 地址的 **bond**

```
auto lo
iface lo inet loopback

iface eno1 inet manual

iface eno2 inet manual

iface eno3 inet manual

auto bond0
iface bond0 inet static
      bond-slaves eno1 eno2
      address  192.168.1.2/24
```

```
      bond-miimon 100
      bond-mode 802.3ad
      bond-xmit-hash-policy layer2+3

auto vmbr0
iface vmbr0 inet static
        address  10.10.10.2/24
        gateway  10.10.10.1
        bridge-ports eno3
        bridge-stp off
        bridge-fd 0
```

另一种做法是直接将 bond 用作 bridge port。这可用于让客户机网络具备容错能力。

示例：将 **bond** 用作 **bridge port**

```
auto lo
iface lo inet loopback

iface eno1 inet manual

iface eno2 inet manual

auto bond0
iface bond0 inet manual
      bond-slaves eno1 eno2
      bond-miimon 100
      bond-mode 802.3ad
      bond-xmit-hash-policy layer2+3

auto vmbr0
iface vmbr0 inet static
        address  10.10.10.2/24
        gateway  10.10.10.1
        bridge-ports bond0
        bridge-stp off
        bridge-fd 0
```

3.4.8 VLAN 802.1Q

虚拟 LAN（VLAN）是在网络二层进行划分和隔离的广播域。因此，可以在一个物理网络中拥有多个彼
个）。

每个 VLAN 网络由一个通常称为 *tag* 的数字标识。随后网络数据包会被“打标签”，用于标识其所属的虚

客户机网络中的 **VLAN**

Proxmox VE 原生支持此配置。创建虚拟机时可以指定 VLAN tag。VLAN tag 是客户机网络配置的一部
bridge 配置不同，网络层支持不同模式来实现 VLAN：

- **Linux bridge** 上的 **VLAN awareness:** 在这种情况下，每个客户机的虚拟网卡都会分配一个 VLAN tag，并由 Linux bridge 透明支持。也可以使用 Trunk 模式，但这需要在客户机内进行配置。
- **Linux bridge** 上的**“传统” VLAN:** 与 VLAN awareness 方法不同，此方法不是透明的，并会为每 VLAN 创建一个 VLAN 设备及其关联 bridge。也就是说，例如在 VLAN 5 上创建客户机时，会创建 eno1.5 和 vmbr0v5 两个接口，并一直保留到发生重启。
- **Open vSwitch VLAN:** 此模式使用 OVS VLAN 功能。
- 客户机内配置的 **VLAN:** VLAN 在客户机内部分配。在这种情况下，配置完全在客户机内完成，无法 NIC 上使用多个 VLAN。

主机上的 **VLAN**

为了允许主机与隔离网络通信，可以将 VLAN tag 应用到任意网络设备（NIC、Bond、Bridge）。一般 NIC 之间抽象层最少的接口上配置 VLAN。

例如，在默认配置中，如果希望将主机管理地址放在单独的 VLAN 上。

示例：使用传统 **Linux bridge**，将 **VLAN 5** 用于 **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  10.10.10.2/24
        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：使用 **VLAN aware Linux bridge**，将 **VLAN 5** 用于 **Proxmox VE** 管理 **IP**

```
auto lo
iface lo inet loopback

iface eno1 inet manual
```

```
auto vmbr0.5
iface vmbr0.5 inet static
        address  10.10.10.2/24
        gateway  10.10.10.1

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

下一个示例使用相同配置，但通过 bond 使该网络具备故障保护能力。

示例：使用传统 **Linux bridge**，通过 **bond0** 上的 **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
      bond-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  10.10.10.2/24
        gateway  10.10.10.1
        bridge-ports bond0.5
        bridge-stp off
        bridge-fd 0

auto vmbr0
iface vmbr0 inet manual
        bridge-ports bond0
        bridge-stp off
        bridge-fd 0
```

3.4.9 在节点上禁用 IPv6

无论是否部署 IPv6，Proxmox VE 都能在所有环境中正常工作。建议保留所有设置的默认值。

如果仍需在节点上禁用 IPv6 支持，请创建适当的 sysctl.conf (5) 片段文件，并设置正确的 sysctls, 例如添加内容如下的 /etc/sysctl.d/disable-ipv6.conf：

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

相比在 内核命令行 中禁用 IPv6 模块加载，更推荐使用此方法。

3.4.10 在 Bridge 上禁用 MAC Learning

默认情况下，bridge 上会启用 MAC learning，以确保虚拟客户机及其网络获得平稳体验。

但在某些环境中，这可能并非期望行为。自 Proxmox VE 7.3 起，可以在 /etc/network/interfac 中为 bridge 设置 bridge-disable-mac-learning 1 配置，从而禁用该 bridge 上的 MAC learning，例如：

```
# ...

auto vmbr0
iface vmbr0 inet static
        address  10.10.10.2/24
        gateway  10.10.10.1
        bridge-ports ens18
        bridge-stp off
        bridge-fd 0
        bridge-disable-mac-learning 1
```

启用后，Proxmox VE 会手动将虚拟机和容器中配置的 MAC 地址添加到 bridge 的 forwarding database，以确保客户机仍可使用网络，但前提是它们使用自身实际的 MAC 地址。

3.5 时间同步

Proxmox VE 集群栈本身高度依赖所有节点之间精确同步的时间。其他一些组件（如 Ceph）也要求所

节点之间的时间同步可以通过“网络时间协议”（NTP）实现。从 Proxmox VE 7 开始，默认 NTP 守护进程为 chrony，而 Proxmox VE 6 使用 systemd-timesyncd。两者都预配置为使用一组公共服

Important
如果将系统升级到 Proxmox VE 7，建议手动安装 chrony、ntp 或 openntpd 之一。

3.5.1 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 Proxmox VE 节点由于受限的 防火墙规则 NTP 服务器，并指示 NTP 守护进程使用它们。

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
```

重启 chrony：

```
# systemctl restart chronyd
```

检查 journal，确认新配置的 NTP 服务器正在被使用：

```
# journalctl --since -1h -u chrony
```

```
...
Aug 26 13:00:09 node1 systemd[1]: Started chrony, an NTP client/server.
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
```

然后重启同步服务（systemctl restart systemd-timesyncd），并通过检查 journal（journa
--since -1h -u systemd-timesyncd）确认新配置的 NTP 服务器正在使用：

```
...
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
...
```

常用命令示例：

```
timedatectl status
chronyc tracking
systemctl status chrony
```

3.6 外部指标服务器

在 Proxmox VE 中，可以定义外部指标服务器，用于周期性接收主机、虚拟客户机和存储的各类统计信

当前支持：

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

外部指标服务器定义保存在 `/etc/pve/status.cfg` 中，也可以通过 Web 界面编辑。

3.6.1 Graphite 服务器配置

默认端口为 **2003**，默认 Graphite 路径为 **proxmox**。

默认情况下，Proxmox VE 通过 UDP 发送数据，因此 Graphite 服务器必须配置为接受 UDP 数据。在这里也可以为未使用标准 **1500** MTU 的环境配置最大传输单元（MTU）。

也可以将插件配置为使用 TCP。为了避免阻塞重要的 `pvestatd` 统计采集守护进程，需要设置超时时间

3.6.2 InfluxDB 插件配置

Proxmox VE 通过 UDP 发送数据，因此 InfluxDB 服务器也必须为此进行配置。必要时，也可以在这里 MTU。

以下是在 InfluxDB 服务器上的 InfluxDB 配置示例：

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

使用该配置时，服务器会在所有 IP 地址的 8089 端口监听，并将数据写入 **proxmox** 数据库。

也可以将插件配置为使用 InfluxDB 2.x 的 http(s) API。InfluxDB 1.8.x 也包含与该 v2 API 向前兼容的 API endpoint。

要使用该方式，请根据配置将 `influxdbproto` 设置为 `http` 或 `https`。默认情况下，Proxmox VE 使用组织 `proxmox` 和 bucket/db `proxmox`（可分别通过 `organization` 和 `bucket` 配置项设置）

由于 InfluxDB 的 v2 API 只能在认证后使用，因此必须生成一个能够写入正确 bucket 的 token 并进行设置。

在 1.8.x 的 v2 兼容 API 中，如有需要，可以使用 `user:password` 作为 token；由于 `organizatio` 在 InfluxDB 1.x 中没有意义，因此可以省略。

还可以通过 `timeout` 设置 HTTP 超时时间（默认 1s），并通过 `max-body-size` 设置最大批量大小（ 25000000 字节）。后者对应 InfluxDB 中同名设置。

3.6.3 常用命令示例

查看外部指标服务器配置文件：

```
# cat /etc/pve/status.cfg
```

确认统计采集守护进程正在运行：

```
# systemctl status pvestatd
```

3.7 磁盘健康监控

虽然建议使用健壮且具备冗余能力的存储，但监控本地磁盘的健康状况仍然非常有用。

从 Proxmox VE 4.3 开始，系统会安装并依赖 smartmontools [1] 软件包。这是一组用于监控和控制本地 S.M.A.R.T. 系统的工具。

可以通过执行以下命令获取磁盘状态：

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

其中 /dev/sdX 是某块本地磁盘的路径。

如果输出显示：

```
SMART support is: Disabled
```

可以使用以下命令启用：

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

有关 smartctl 用法的更多信息，请参见 man smartctl。

默认情况下，smartmontools 守护进程 smartd 处于启用和运行状态，并每 30 分钟扫描 /dev/sdX 和 /dev/hdX 下的磁盘以检查错误和警告；如果检测到问题，会向 root 发送电子邮件。

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

3.7.1 常用命令示例

查看某块磁盘的 SMART 摘要信息：

```
# smartctl -H /dev/sdX
```

查看 smartd 服务状态：

```
# systemctl status smartmontools
```

如果硬盘通过硬件 RAID 控制器使用，通常需要使用控制器厂商提供的工具来监控 RAID 阵列中的磁盘以及 RAID 控制器厂商的文档。

[1] smartmontools homepage https://www.smartmontools.org

3.8 逻辑卷管理器 (LVM)

多数用户会将 Proxmox VE 直接安装到本地磁盘上。Proxmox VE 安装 CD 为本地磁盘管理提供了多种
当前默认配置使用 LVM。安装程序允许你为这种配置选择一块单独的磁盘，并将该磁盘作为 卷组（**V**ol
Group，VG）pve 的物理卷。以下输出来自一套使用小型 8GB 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
```

安装程序会在该 VG 中分配三个逻辑卷（**L**ogical **V**olumes，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，并包含操作系统。

swap
: 交换分区。

data
: 该卷使用 LVM-thin，用于存储虚拟机镜像。LVM-thin 更适合这类用途，因为它能高效支持快照和

对于 Proxmox VE 4.1 及更早版本，安装程序会创建名为“data”的标准逻辑卷，并挂载到 /var/lib/

从 4.2 版本开始，逻辑卷“data”是一个 LVM-thin 池，用于存储基于块的客户机镜像，而 /var/lib/
只是根文件系统上的一个目录。

3.8.1 硬件

强烈建议在此类配置中使用硬件 RAID 控制器（带 BBU）。这可以提升性能、提供冗余，并让磁盘更换

LVM 本身不需要任何特殊硬件，内存需求也很低。

3.8.2 引导加载器

默认会安装两个引导加载器。第一个分区包含标准 GRUB 引导加载器。第二个分区是 **E**FI **S**ystem
Partition（ESP），用于在 EFI 系统上启动，并允许从用户空间应用 持久化固件更新。

3.8.3 创建卷组

假设有一块空磁盘 /dev/sdb，我们希望在其上创建名为 “vmdata” 的卷组。

Caution
请注意，以下命令会销毁 /dev/sdb 上的所有现有数据。

首先创建一个分区。

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

创建一个物理卷（**P**hysical **V**olume，PV），不要求确认，并使用 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
```

3.8.4 为 /var/lib/vz 创建额外 LV

可以通过创建新的 thin LV 轻松完成。

```
# lvcreate -n <Name> -V <Size[M,G,T]> <VG>/<LVThin_pool>
```

实际示例：

```
# lvcreate -n vz -V 10G pve/data
```

现在必须在该 LV 上创建文件系统。

```
# mkfs.ext4 /dev/pve/vz
```

最后需要将其挂载。

Warning
请确保 /var/lib/vz 为空。在默认安装中，它并不是空目录。

要使其始终可访问，请将以下行添加到 /etc/fstab。

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

3.8.5 调整 thin 池大小

使用以下命令调整 LV 和元数据池大小：

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

Note
扩展数据池时，也必须同时扩展元数据池。

3.8.6 创建 LVM-thin 池

thin 池必须创建在卷组之上。关于如何创建卷组，请参见 LVM 章节。

```
# lvcreate -L 80G -T -n vmstore vmdata
```

3.8.7 常用命令示例

以下命令可用于查看本地 LVM 状态和 thin 池使用情况：

```
# pvs
# vgs
# lvs
# lvs -a
```

3.9 Linux 上的 ZFS

ZFS 是由 Sun Microsystems 设计的文件系统与逻辑卷管理器组合。从 Proxmox VE 3.4 开始，ZFS 文件系统的原生 Linux 内核移植版本作为可选文件系统引入，同时也可作为根文件系统的额外选择。无 ZFS 模块，所有软件包均已包含。

通过使用 ZFS，可以在低预算硬件上获得丰富的企业级功能，也可以借助 SSD 缓存甚至纯 SSD 部署构建高性能系统。ZFS 可以用适度的 CPU 和内存负载以及易于管理的方式，替代成本较高的硬件 RAID 卡。

ZFS 的一般优势

- 可通过 Proxmox VE GUI 和 CLI 轻松配置和管理。
- 可靠。
- 防止数据损坏。
- 文件系统级数据压缩。
- 快照。

- 写时复制克隆。
- 多种 RAID 级别：RAID0、RAID1、RAID10、RAIDZ-1、RAIDZ-2、RAIDZ-3、 dRAID, dRAID2, dRAID3
- 可使用 SSD 作为缓存。
- 自愈能力。
- 持续完整性检查。
- 面向大容量存储设计。
- 通过网络进行异步复制。
- 开源。
- 加密。
- …

3.9.1 硬件

ZFS 对内存依赖较高，因此起步至少需要 8GB。实践中，应在硬件和预算允许的范围内尽量配置更多内
ECC RAM。

如果使用专用缓存盘和/或日志盘，应使用企业级 SSD。这可以显著提升整体性能。

Important
不要在带有自身缓存管理的硬件 RAID 控制器之上使用 ZFS。ZFS 需要直接与磁盘通信。HBA 适配器，或刷入 “IT” 模式的 LSI 控制器一类设备更合适。

如果是在虚拟机中实验安装 Proxmox VE（嵌套虚拟化），不要为该虚拟机的磁盘使用 `virtio`，因为 ZFS 不支持这种磁盘。请改用 IDE 或 SCSI（也可使用 `virtio` SCSI 控制器类型）。

3.9.2 作为根文件系统安装

使用 Proxmox VE 安装程序安装时，可以为根文件系统选择 ZFS。安装时需要选择 RAID 类型：

RAID0	也称为 “striping”。此类卷的容量是所有磁盘容量之和。但 RAID0 不提供任何冗余，因此单块磁盘故障就会导致该卷不可用。
RAID1	也称为 “mirroring”。数据会以相同方式写入所有磁盘。此模式至少需要 2 块同等大小的磁盘，最终容量等同于单块磁盘。
RAID10	RAID0 与 RAID1 的组合。至少需要 4 块磁盘。
RAIDZ-1	RAID-5 的变体，单校验。至少需要 3 块磁盘。

RAIDZ-2 RAID-5 的变体，双校验。至少需要 4 块磁盘。

RAIDZ-3 RAID-5 的变体，三重校验。至少需要 5 块磁盘。

安装程序会自动对磁盘分区，创建名为 rpool 的 ZFS pool，并将根文件系统安装到 ZFS 子卷 rpool/ROOT/pve-1 上。

安装程序还会创建名为 rpool/data 的子卷用于存储虚拟机镜像。为了让 Proxmox VE 工具使用该子卷 /etc/pve/storage.cfg 中创建以下配置项：

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

安装完成后，可以使用 zpool 命令查看 ZFS pool 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。以下命令会列出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  -
```

3.9.3 ZFS RAID 级别考量

选择 ZFS pool 布局时，需要考虑几个因素。ZFS pool 的基本构建块是虚拟设备，即 vdev。pool 中的所有 vdev 都会被均衡使用，数据会在它们之间条带化（RAID0）。有关 vdev 的更多详情，请查看 zpoolconcepts(7) manpage。

性能

每种 vdev 类型都有不同的性能行为。主要关注的两个参数是 IOPS（每秒输入/输出操作数）以及数据

写入数据时，*mirror* vdev（RAID1）在这两个参数上的表现大致类似单块磁盘。读取数据时，性能会
mirror 中磁盘数量线性扩展。

一种常见场景是有 4 块磁盘。将其设置为 2 个 mirror vdev（RAID10）时，从 IOPS 和带宽角度看，该
pool 的写入特性相当于两块单盘。读取操作则接近 4 块单盘。

任意冗余级别的 *RAIDZ* 在 IOPS 方面大致类似单块磁盘，但具有较高带宽。具体带宽取决于 RAIDZ
vdev 的大小和冗余级别。

dRAID pool 的性能应与等价的 *RAIDZ* pool 相当。

对于运行虚拟机而言，在多数情况下 IOPS 是更重要的指标。

大小、空间使用和冗余

由 *mirror* vdev 组成的 pool 具有最佳性能特性，但可用空间只有可用磁盘容量的 50%。如果一个
mirror vdev 由超过 2 块磁盘组成，例如 3-way mirror，则可用空间更少。每个 mirror 至少需要一块
才能保持可用。

由 N 块磁盘组成的 *RAIDZ* 类型 vdev 可用空间大致为 N-P，其中 P 为 RAIDZ 级别。RAIDZ
级别表示在不丢失数据的情况下可任意故障的磁盘数量。4 块磁盘的 RAIDZ2 pool 是一个特殊场景；这
2 个 mirror vdev，因为可用空间相同但性能更好。

使用任意 RAIDZ 级别时，另一个重要因素是用于虚拟机磁盘的 ZVOL dataset 的行为。对于每个数据块
都需要校验数据，其大小至少为 pool 的 `ashift` 值所定义的最小块大小。`ashift` 为 12 时，pool
块大小为 4k。ZVOL 的默认块大小为 8k。因此，在 RAIDZ2 中，每写入一个 8k 块，就会额外写入两个
4k 校验块，即 8k + 4k + 4k = 16k。当然这是简化说明，实际情况会因元数据、压缩等因素而略有不

检查 ZVOL 的以下属性时，可以观察到这种行为：

- `volsize`
- `refreservation`（如果 pool 未使用 thin provisioning）
- `used`（如果 pool 使用 thin provisioning 且不存在快照）

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

`volsize` 是呈现给虚拟机的磁盘大小，而 `refreservation` 显示 pool 上的预留空间，其中包含校验
pool 使用 thin provisioning，则 `refreservation` 会设置为 0。另一种观察方式是比较虚拟机内部已
`used` 属性。请注意，快照会影响该值。

有几种方式可以抵消空间使用增加的问题：

- 增大 `volblocksize` 以改善数据与校验的比例
- 使用 *mirror* vdev 替代 *RAIDZ*
- 使用 `ashift=9`（块大小为 512 字节）

volblocksize 属性只能在创建 ZVOL 时设置。默认值可以在存储配置中修改。这样做时，需要对客户
ZFS 层转移到了客户机内部。

创建 pool 时使用 ashift=9 可能会导致性能较差，具体取决于底层磁盘，并且之后无法更改。

Mirror vdev（RAID1、RAID10）对虚拟机工作负载更友好。除非环境有特定需求和特性，且 RAIDZ
性能特征可以接受，否则应优先使用 mirror vdev。

3.9.4 ZFS dRAID

在 ZFS dRAID（declustered RAID）中，热备盘会参与 RAID。其备用容量会被保留，并在某块磁盘故
RAIDZ 提供更快的重建速度。更多信息请参见官方 OpenZFS 文档。 [2]

Note
dRAID 适用于一个 dRAID 中包含超过 10-15 块磁盘的场景。在大多数使用场景中，磁盘数量较少时
RAIDZ 配置通常更合适。

Note
GUI 要求的磁盘数量比最低要求多一块（例如 dRAID1 需要 3
块）。它会假定同时添加一块备用磁盘。

- dRAID1 或 dRAID：至少需要 2 块磁盘，可在丢失数据前容忍 1 块磁盘故障
- dRAID2：至少需要 3 块磁盘，可在丢失数据前容忍 2 块磁盘故障
- dRAID3：至少需要 4 块磁盘，可在丢失数据前容忍 3 块磁盘故障

更多信息可在 manual page 中找到：

```
# man zpoolconcepts
```

Spares 和 Data

spares 的数量告诉系统在磁盘故障时应预留多少块磁盘。默认值为 0 spares。没有 spare 时，重建不

data 定义冗余组中的设备数量。默认值为 8。除非 disks - parity - spares 小于 8，此时会使用
data 设备会带来更高 IOPS、更好的压缩率和更快的 resilvering，但定义较少 data 设备会降低
pool 的可用存储容量。

3.9.5 Bootloader

Proxmox VE 使用 proxmox-boot-tool 管理 bootloader 配置。详情请参见 Proxmox VE 主机
bootloader 章节。

[2] OpenZFS dRAID https://openzfs.github.io/openzfs-docs/Basic%20Concepts/-dRAID%20Howto.html

3.9.6 ZFS 管理

本节给出一些常见任务的使用示例。ZFS 本身功能非常强大，并提供大量选项。管理 ZFS 的主要命令是 `zfs` 和 `zpool`。这两个命令都带有完善的 manual page，可以通过以下命令阅读：

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

创建新的 **zpool**

创建新 pool 至少需要一块磁盘。`ashift` 应与底层磁盘的扇区大小（2 的 `ashift` 次方）相同或更大。

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

Tip
Pool 名称必须遵循以下规则：

- 以字母开头（a-z 或 A-Z）
- 仅包含字母数字、-、_、.、: 或``（空格）字符
- 不得以 `mirror`、`raidz`、`draid` 或 `spare` 开头
- 不得为 `log`

要启用压缩（参见 ZFS 中的压缩 章节）：

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

创建带 **RAID-0** 的新 **pool**

至少 1 块磁盘

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

创建带 **RAID-1** 的新 **pool**

至少 2 块磁盘

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

创建带 **RAID-10** 的新 **pool**

至少 4 块磁盘

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

创建带 **RAIDZ-1** 的新 **pool**

至少 3 块磁盘

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

创建带 **RAIDZ-2** 的新 **pool**

至少 4 块磁盘

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

设置 pool 前，请阅读 ZFS RAID 级别考量 章节，以粗略估算 IOPS 和带宽预期，特别是在计划使用 RAID-Z 模式时。

创建带缓存（**L2ARC**）的新 **pool**

可以使用专用设备或分区作为二级缓存以提升性能。此类缓存设备尤其有助于大部分为静态数据的随机 ARC 之间的额外缓存层，如果由于内存约束必须降低 ARC，也能提供帮助。

创建带磁盘缓存的 **ZFS pool**

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

这里仅使用了单个 `<device>` 和单个 `<cache-device>`，但也可以使用更多设备，如 创建带 RAID 的新 pool 中所示。

请注意，缓存设备不存在 mirror 或 RAID 模式，它们只是简单累加。

如果任何缓存设备在读取时产生错误，ZFS 会透明地将该请求转向底层存储层。

创建带日志（**ZIL**）的新 **pool**

可以为 ZFS Intent Log（ZIL）使用专用磁盘或分区。它主要用于提供安全的同步事务，因此常用于数据 `fsync` 操作的程序。

pool 是默认的 ZIL 位置。将 ZIL I/O 负载转移到独立设备，可以在减轻主 pool 负载的同时降低事务延迟

对于直接作为日志设备或通过分区作为日志设备的磁盘，建议：

- 使用具备掉电保护的快速 SSD，因为这类设备的提交延迟更低。
- 为分区（或整个设备）至少使用几 GB 空间，但超过已安装内存的一半不会带来实际优势。

创建带独立日志设备的 **ZFS pool**

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

上例使用了单个 <device> 和单个 <log-device>，但也可以结合其他 RAID 变体使用，如 创建带 RAID 的新 pool 章节所述。

也可以将日志设备 mirror 到多个设备。这主要用于确保单个日志设备故障时，性能不会立即下降。

如果所有日志设备均故障，ZFS 会重新使用主 pool 本身，直到日志设备被替换。

向现有 **pool** 添加缓存和日志

如果某个 pool 没有缓存和日志，也可以在任何时候添加二者或其中之一。

例如，假设有一块具备掉电保护的优质企业级 SSD，并希望用它提升 pool 的整体性能。

由于日志设备最大大小应约为已安装物理内存的一半，这意味着 ZIL 很可能只占用 SSD 的一小部分，剩

首先需要使用 parted 或 gdisk 在 SSD 上创建两个 GPT 分区。

然后即可将它们添加到 pool：

向现有 **pool** 同时添加独立日志设备和二级缓存

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

只需将 <pool>、<device-part1> 和 <device-part2> 替换为 pool 名称以及两个分区的 /dev/disk/by-id/ 路径。

也可以分别添加 ZIL 和缓存。

向现有 **ZFS pool** 添加日志设备

```
# zpool add <pool> log <log-device>
```

更换故障设备

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

更换故障的可启动设备

根据 Proxmox VE 的安装方式，它会通过 proxmox-boot-tool [3] 使用 systemd-boot 或 GRUB，或者使用普通 GRUB 作为 bootloader（参见 主机 Bootloader）。可以通过运行以下命令检查

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

[3] Systems installed with Proxmox VE 6.4 or later, EFI systems installed with Proxmox VE 5.4 or later

复制分区表、重新生成 GUID 并替换 ZFS 分区的前几个步骤相同。为了让系统能够从新磁盘启动，还需 bootloader 执行不同步骤。

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

Note
使用 `zpool status -v` 命令监控新磁盘 resilvering 过程的进度。

使用 **proxmox-boot-tool**：

```
# proxmox-boot-tool format <new disk's ESP>
# proxmox-boot-tool init <new disk's ESP> [grub]
```

Note
ESP 表示 EFI System Partition。从版本 5.4 起，使用 Proxmox VE 安装程序时，它会在可启动磁盘上设置为第 2 个分区。详情请参见 设置新分区以用作同步 ESP。

Note
如果 `proxmox-boot-tool status` 表明当前磁盘使用 GRUB，请确保将 *grub* 作为模式传递给 `proxmox-boot-tool init`，尤其是在启用 Secure Boot 时。

使用普通 **GRUB**：

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

Note
普通 GRUB 仅用于使用 Proxmox VE 6.3 或更早版本安装且尚未手动迁移到 `proxmox-boot-tool` 的系统。

3.9.7 配置电子邮件通知

ZFS 带有事件守护进程 ZED，用于监控 ZFS 内核模块生成的事件。该守护进程也可以在发生 pool 错误等 ZFS 事件时发送电子邮件。较新的 ZFS 软件包将该守护进程放在独立的 `zfs-zed` 软件包中，该 Proxmox VE 中默认安装。

可以使用喜欢的编辑器通过 `/etc/zfs/zed.d/zed.rc` 文件配置该守护进程。电子邮件通知所需设置 `ZED_EMAIL_ADDR`，默认设置为 `root`。

```
ZED_EMAIL_ADDR=”root”
```

请注意，Proxmox VE 会将发往 `root` 的邮件转发到为 root 用户配置的电子邮件地址。

3.9.8 限制 ZFS 内存使用

默认情况下，ZFS 使用主机内存的 *50* % 作为 **A**daptive **R**eplacement **C**ache （ARC）。从 Proxmox VE 8.1 开始的新安装中，ARC 使用上限会设置为已安装物理内存的 *10* %，最大限制为 16 GiB。该值写入 /etc/modprobe.d/zfs.conf。

为 ARC 分配足够内存对 I/O 性能至关重要，因此降低该值时应谨慎。一般经验是至少分配 2 GiB Base + 1 GiB/TiB-Storage。例如，如果 pool 有 8 TiB 可用存储空间，则应为 ARC 使用 10 GiB 内存。

ZFS 还会强制执行 64 MiB 的最小值。

可以通过直接写入 zfs_arc_max 模块参数，为当前启动周期修改 ARC 使用上限（重启会重置该更改）

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

要*永久更改* ARC 限制，请在 /etc/modprobe.d/zfs.conf 中添加（或修改已有的）以下行：

```
options zfs zfs_arc_max=8589934592
```

此示例设置会将使用量限制为 8 GiB（$8 * 2^{30}$）。

Important
如果期望的 zfs_arc_max 值小于或等于 zfs_arc_min（默认为系统内存的 1/32），则 zfs_arc_max 会被忽略，除非同时将 zfs_arc_min 设置为至多 zfs_arc_max - 1。

```
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
```

在总内存超过 256 GiB 的系统上，此示例设置会将使用量（临时）限制为 8 GiB（$8 * 2^{30}$）。在这种情 zfs_arc_max 不会生效。

Important
如果根文件系统是 ZFS，每次该值变化后都必须更新 initramfs：

```
# update-initramfs -u -k all
```

必须*重启*才能激活这些更改。

3.9.9 ZFS 上的 SWAP

在 zvol 上创建的 swap 空间可能会产生一些问题，例如阻塞服务器或造成高 I/O 负载，这种情况常见于

强烈建议使用足够内存，使系统通常不会遇到低内存情况。如果需要或希望添加 swap，首选是在物理硬 swap 设备。可以在安装程序高级选项中为此预留一些空间。此外，还可以降低 "swappiness" 值。服 10：

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

要使 swappiness 持久化，请使用喜欢的编辑器打开 /etc/sysctl.conf 并添加以下行：

```
vm.swappiness = 10
```

Table 3.1: Linux 内核 swappiness 参数值

值	策略
vm.swappiness = 0	内核只会为了避免 *out of memory* 情况而使用 swap
vm.swappiness = 1	在不完全禁用 swap 的情况下，尽量减少 swap 使用量。
vm.swappiness = 10	当系统内存充足时，有时建议使用该值以改善性能。
vm.swappiness = 60	默认值。
vm.swappiness = 100	内核会积极使用 swap。

3.9.10 加密的 ZFS Dataset

Warning
Proxmox VE 中的原生 ZFS 加密仍处于实验阶段。已知限制和问题包括加密 dataset 的复制 [a]，以及使用快照或 ZVOL 时的 checksum 错误。[b]

[a] Bugzilla #2350
[b] OpenZFS issue #11688

ZFS on Linux 0.8.0 版本引入了对 dataset 原生加密的支持。从较早 ZFS on Linux 版本升级后，可以 pool 启用加密特性：

```
# 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
```

Warning
目前不支持使用 GRUB 从包含加密 dataset 的 pool 启动，并且对启动时自动解锁加密 dataset 仅提供有限支持。不支持加密的旧版 ZFS 将无法解密已存储数据。

Note
建议在启动后手动解锁存储 dataset，或编写自定义 unit，在启动时将解锁所需 key material 传递给 `zfs load-key`。

Warning
在为生产数据启用加密前，请建立并测试备份流程。如果相关 key material/passphrase/keyfile 丢失，将无法再访问加密数据。

加密需要在创建 dataset/zvol 时设置，并默认由子 dataset 继承。例如，要创建加密 dataset 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
```

在该存储上创建的所有客户机卷/磁盘都会使用父 dataset 的共享 key material 加密。

要实际使用该存储，需要加载关联的 key material 并挂载 dataset。可以通过以下命令一步完成：

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

也可以通过设置 keylocation 和 keyformat 属性，使用（随机）keyfile 替代交互式输入 passphra
zfs change-key 对现有 dataset 设置：

```
# dd if=/dev/urandom of=/path/to/keyfile bs=32 count=1

# zfs change-key -o keyformat=raw -o keylocation=file:///path/to/keyfile ↩
  tank/encrypted_data
```

Warning
使用 keyfile 时，必须特别注意保护 keyfile，避免未授权访问或意外丢失。没有 keyfile，就无法访问明文数据。

在加密 dataset 下创建的客户机卷会相应设置其 encryptionroot 属性。每个 encryptionroot 只需加载一次 key material，其下所有加密 dataset 即可使用。

更多详情和高级用法，请参见 encryptionroot、encryption、keylocation、keyformat 和 keystatus 属性，zfs load-key、zfs unload-key、zfs change-key 命令，以及 man zfs 中的 Encryption 章节。

3.9.11 ZFS 中的压缩

在 dataset 上启用压缩后，ZFS 会尝试在写入所有*新*块之前压缩它们，并在读取时解压缩。已存在的

可以使用以下命令启用压缩：

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

建议使用 lz4 算法，因为它只会带来很小的 CPU 开销。也可以使用其他算法，例如 lzjb 和 gzip-N，N 是从 1（最快）到 9（最佳压缩率）的整数。根据算法和数据可压缩性不同，启用压缩甚至可能提升 I/O 性能。

可以随时使用以下命令禁用压缩：

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

同样，只有新块会受到该更改影响。

3.9.12 ZFS Special Device

自 0.8.0 版本起，ZFS 支持 special 设备。pool 中的 special 设备用于存储元数据、去重表，并可选

对于由慢速机械硬盘组成且存在大量元数据变更的 pool，special 设备可以提升速度。例如，涉及创建 special 设备中受益。还可以配置 ZFS dataset，将整个小文件存储在 special 设备上，从而进一步 设备应使用快速 SSD。

Important
special 设备的冗余级别应与 pool 保持一致，因为 special 设备是整个 pool 的故障点。

Warning
向 pool 添加 special 设备后无法撤销。

创建带 **special** 设备和 **RAID-1** 的 **pool**：

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

向现有 **RAID-1 pool** 添加 **special** 设备：

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

ZFS dataset 暴露 special_small_blocks=<size> 属性。size 可以为 0，表示禁用在 special 设备上存储小文件块；也可以是 512B 到 1M 范围内的 2 的幂。设置该属性后，小于 size 的新文件块会 special 设备上。

Important
如果 special_small_blocks 的值大于或等于 dataset 的 recordsize（默认 128K），则*所有*数据都会写入 special 设备，因此请谨慎设置。

在 pool 上设置 special_small_blocks 属性会改变所有子 ZFS dataset 该属性的默认值（例如该 pool 中的所有容器都会选择使用小文件块）。

在整个 **pool** 范围内选择将小于 **4K-blocks** 的所有文件写入 **special device**：

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

为单个 **dataset** 选择使用小文件块：

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

为单个 **dataset** 退出使用小文件块：

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

3.9.13 ZFS Pool 特性

ZFS 磁盘格式的变更只会在主版本变更之间进行，并通过 **features** 指定。所有 feature 以及通用机制
`zpool-features(5)` manpage 中有详细记录。

由于启用新 feature 可能导致旧版 ZFS 无法导入 pool，因此必须由管理员主动在 pool 上运行 `zpool upgrade` 完成（参见 `zpool-upgrade(8)` manpage）。

除非需要使用某个新 feature，否则启用它们没有额外收益。

事实上，启用新 feature 存在一些缺点：

- 如果 root on ZFS 系统仍使用 GRUB 启动，而 rpool 上激活了新 feature，由于 GRUB 中的 ZFS 实现不兼容，系统将无法启动。
- 使用仍随附旧 ZFS 模块的旧内核启动时，系统将无法导入任何已升级的 pool。
- 启动较旧的 Proxmox VE ISO 来修复无法启动的系统也同样不可行。

Important
如果系统仍使用 GRUB 启动，*不要*升级 rpool，因为这会导致系统无法启动。这包括在 Proxmox VE 5.4 之前安装的系统，以及使用 legacy BIOS boot 启动的系统（参见 如何判断所使用的 bootloader）。

为 **ZFS pool** 启用新 **feature**：

```
# zpool upgrade <pool>
```

3.9.14 常用命令示例

查看所有 ZFS pool 的状态：

```
# zpool status
```

列出 ZFS dataset：

```
# zfs list
```

3.10 BTRFS

Warning
BTRFS 集成目前在 Proxmox VE 中仍是 **technology preview**。

BTRFS 是 Linux 内核原生支持的现代写时复制文件系统，实现了快照、内置 RAID，以及通过数据和元
Proxmox VE 7.0 开始，BTRFS 作为根文件系统的可选项引入。

BTRFS 的一般优势

- 主系统设置与传统基于 ext4 的设置几乎相同
- 快照
- 文件系统级数据压缩
- 写时复制克隆
- RAID0, RAID1 and RAID10
- 防止数据损坏
- 自修复
- Linux 内核原生支持

注意事项

- RAID 级别 5/6 仍处于实验阶段且存在风险，请参见 BTRFS Status

3.10.1 作为根文件系统安装

使用 Proxmox VE 安装程序安装时，可以为根文件系统选择 BTRFS。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之和。但 RAID0 不提供任何冗余，因此单个驱动器故障就会使该卷不可用。
RAID1	也称为`‘镜像’’。数据会以相同方式写入所有磁盘。此模式至少需要 2 块相同容量的磁盘。最终容量等同于单块磁盘容量。
RAID10	RAID0 和 RAID1 的组合。至少需要 4 块磁盘。

安装程序会自动对磁盘分区，并在 /var/lib/pve/local-btrfs 创建一个额外子卷。 为了让 Proxmox VE 工具使用该子卷，安装程序会在 /etc/pve/storage.cfg 中创建 以下配置条目：

```
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
```

这会显式禁用默认的 local 存储，改用额外子卷上的 BTRFS 专用存储条目。

btrfs 命令用于配置和管理 BTRFS 文件系统。安装完成后，以下命令会列出所有 额外子卷：

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

3.10.2 BTRFS 管理

本节给出一些常见任务的使用示例。

创建 **BTRFS** 文件系统

创建 BTRFS 文件系统时使用 mkfs.btrfs。-d 和 -m 参数分别用于设置数据和 元数据的 profile。可选的 -L 参数可用于设置标签。

通常支持以下模式：single、raid0、raid1、raid10。

在单块磁盘 /dev/sdb 上创建标签为 My-Storage 的 BTRFS 文件系统：

```
 # 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
```

挂载 **BTRFS** 文件系统

随后可以手动挂载新的文件系统，例如：

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

BTRFS 也可以像其他挂载点一样添加到 /etc/fstab，从而在启动时自动挂载。 建议避免使用块设备路
mkfs.btrfs 命令输出的 UUID 值， 尤其是在 BTRFS 设置中包含多块磁盘时。

例如：

文件 **/etc/fstab**

```
# ... b''为b''b''简b''b''洁b''b''起b''b''见b''b'',b''b''省b''b''略b''b''其b'' ↩
   b''他b''b''挂b''b''载b''b''点b''

# b''强b''b''烈b''b''建b''b''议b''b''使b''b''用b'' mkfs.btrfs b''输b''b' ↩
   '出b''b''中b''b''的b'' UUID
UUID=e2c0c3ff-2114-4f54-b767-3a203e49f6f3 /my-storage btrfs defaults 0 0
```

Tip
如果不再持有 UUID，可以使用 blkid 工具列出块设备的所有属性。

之后可以通过执行以下命令触发首次挂载：

```
mount /my-storage
```

下一次重启后，系统会在启动时自动执行该挂载。

将 **BTRFS** 文件系统添加到 **Proxmox VE**

可以通过 Web 界面将现有 BTRFS 文件系统添加到 Proxmox VE，也可以使用 CLI，例如：

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

创建子卷

创建子卷会将其关联到 BTRFS 文件系统中的某个路径，在该路径下它会表现为普通目录。

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

之后 /some/path 会像普通目录一样工作。

删除子卷

与通过 rmdir 删除目录不同，使用 btrfs 命令删除子卷时，子卷不需要为空。

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

创建子卷快照

BTRFS 实际上并不区分快照和普通子卷，因此创建快照也可以理解为创建一个子卷的 任意副本。按照约
VE 在为客户机磁盘或子卷创建快照时会使用只读标志， 但该标志之后也可以更改。

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

这会在 /a/new/path 创建 /some/path 上该子卷的只读“克隆”。以后对 /some/path 的任何修改都
如果省略只读（ -r ）选项，则两个子卷都将可写。

启用压缩

默认情况下，BTRFS 不会压缩数据。要启用压缩，可以添加 compress 挂载选项。 注意，已经写入的数
默认情况下，rootfs 会在 /etc/fstab 中按如下方式列出：

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

可以直接在上面的 defaults 后追加 compress=zstd、compress=lzo 或 compress=zlib，如下所

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

该更改会在重启后生效。

检查空间使用情况

传统的 df 工具在某些 BTRFS 设置中可能会输出令人困惑的数值。要获得更好的 估算结果，请使用 btrfs filesystem usage /PATH 命令，例如：

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

常用命令示例：

```
# btrfs subvolume list /
# btrfs filesystem usage /my-storage
# pvesm status
```

3.11 Proxmox 节点管理

Proxmox VE 节点管理工具（ pvenode ）允许你控制节点特定的设置和资源。
目前，pvenode 可用于设置节点描述、对节点上的客户机执行各种批量操作、查看节点 任务历史，以及
SSL 证书；这些证书会通过 pveproxy 用于 API 和 Web GUI。

3.11.1 常用命令示例

以下命令可用于查看节点任务、唤醒集群节点或批量启动已配置自启动的客户机：

```
pvenode task list
pvenode wakeonlan <node>
pvenode startall
```

3.11.2 Wake-on-LAN

Wake-on-LAN（WoL）允许你通过发送 magic packet 来启动网络中处于睡眠状态的计算机。至少需要有一个 NIC 支持此功能，并且需要在计算机固件（BIOS/UEFI）配置中启用相应 选项。选项名 *Enable Wake-on-Lan* 到 *Power On By PCIE Device* 不等； 如果不确定，请查阅主板厂商手册。可以 `ethtool` 检查 <interface> 的 WoL 配置：

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

pvenode 允许你通过 WoL 唤醒集群中处于睡眠状态的成员，使用命令：

```
pvenode wakeonlan <node>
```

这会在 UDP 端口 9 上广播 WoL magic packet，其中包含从 `wakeonlan` 属性获取的 <node> MAC 地址。可以使用以下命令设置节点特定的 `wakeonlan` 属性：

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

发送 WoL 数据包所使用的接口由默认路由决定。可以通过以下命令设置 `bind-interface` 来覆盖它：

```
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX,bind-interface=<iface-name>
```

发送 WoL 数据包时使用的广播地址（默认 `255.255.255.255`）还可以通过以下命令 显式设置 `broadcast-address` 来修改：

```
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX,broadcast-address=< ↩
   broadcast-address>
```

3.11.3 任务历史

排查服务器问题时，例如备份作业失败，查看此前运行任务的日志通常很有帮助。在 Proxmox VE 中，可以通过 `pvenode task` 命令访问节点的任务历史。

可以使用 `list` 子命令获取节点已完成任务的过滤列表。例如，要获取与虚拟机 *100* 相关且以错误结束

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

随后可以使用任务的 UPID 打印该任务日志：

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

3.11.4 批量客户机电源管理

如果有许多虚拟机/容器，可以使用 pvenode 的 `startall` 和 `stopall` 子命令对 客户机执行批量启动 `startall` 只会启动已设置为 开机自动启动的虚拟机/容器（参见 虚拟机的自动启动和关闭）；不过可 `--force` 标志覆盖此行为。 这两个命令也都有 `--vms` 选项，可将停止/启动的客户机限制为指定 VMID。

例如，要启动虚拟机 *100*、*101* 和 *102*，无论它们是否设置了 onboot，可以使用：

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

要停止这些客户机（以及可能正在运行的任何其他客户机），请使用命令：

```
pvenode stopall
```

Note
stopall 命令会先尝试执行干净关机，然后等待所有客户机成功关闭，或等待可覆盖的超时时间（默认 3 分钟）到期。到达该状态后，如果 force-stop 参数没有显式 设置为 0（false），所有仍在运行的虚拟客户机都会被强制停止。

3.11.5 首个客户机启动延迟

如果虚拟机/容器依赖启动较慢的外部资源，例如 NFS 服务器，也可以按节点设置延迟： 从 Proxmox VE 启动完成到第一个配置为自动启动的虚拟机/容器启动之间等待一段时间（参见 虚拟机的自动启

可以通过以下设置实现这一点（其中 10 表示以秒为单位的延迟）：

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

3.11.6 批量客户机迁移

如果升级场景要求将所有客户机从一个节点迁移到另一个节点，pvenode 也提供了用于 批量迁移的 `migrateall` 子命令。默认情况下，该命令会将系统上的每个客户机迁移到 目标节点；不过也可以设置

例如，要将虚拟机 *100*、*101* 和 *102* 迁移到节点 *pve2*，并启用本地磁盘在线 迁移，可以运行：

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

3.11.7 Ballooning 的 RAM 使用率目标

自动内存分配的目标百分比默认为 80%。可以通过 设置 `ballooning-target` 属性按节点自定义此目 90% 主机内存 使用率为目标：

```
pvenode config set --ballooning-target 90
```

3.12 证书管理

3.12.1 集群内部通信证书

每个 Proxmox VE 集群默认都会创建自己的（自签名）证书颁发机构（CA），并为每个节点生成由该 CA 签名的证书。 这些证书用于与集群的 `pveproxy` 服务进行加密通信，并在使用 SPICE 时用于 Shell/Console 功能。

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

3.12.2 API 和 Web GUI 证书

REST API 和 Web GUI 由运行在每个节点上的 pveproxy 服务提供。

对于 pveproxy 使用的证书，可以选择以下方式：

1. 默认使用 /etc/pve/nodes/NODENAME/pve-ssl.pem 中特定于节点的证书。 该证书由集群 CA 签名，因此不会被浏览器和操作系统自动信任。
2. 使用外部提供的证书（例如由商业 CA 签名的证书）。
3. 使用 ACME（Let's Encrypt）获取可信证书并自动续期；该功能也集成在 Proxmox VE API 和 Web 界面中。

对于方式 2 和 3，会使用文件 /etc/pve/local/pveproxy-ssl.pem（以及必须不带密码的 /etc/pve/local/pveproxy-ssl.key）。

Note
请记住，/etc/pve/local 是指向 /etc/pve/nodes/NODENAME 的节点专用符号链接。

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

Warning
不要替换或手动修改 /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，也不要修改 /etc/pve/pve-root-ca.pem 和 /etc/pve/priv/pve-root-ca.key 中的集群 CA 文件。

3.12.3 上传自定义证书

如果已经有要用于某个 Proxmox VE 节点的证书，可以直接通过 Web 界面上传该证书。

请注意，如果提供证书密钥文件，则该文件不得受密码保护。

3.12.4 通过 Let's Encrypt (ACME) 获取可信证书

Proxmox VE 包含 **A**utomatic **C**ertificate **M**anagement **E**nvironment（**ACME**）协议的实现，使 Proxmox VE 管理员可以使用 Let's Encrypt 等 ACME 提供方， 轻松设置在现代操作系统和 Web 浏览器中默认被接受并信任的 TLS 证书。

目前实现的两个 ACME 端点是 Let's Encrypt (LE) 生产环境及其 staging 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， 也支持通过 DNS 插件验证 dns-01 挑战；这些 DNS 插件支持 acme.sh 支持的所有 DNS API 端点。

ACME 账户

需要为每个集群在要使用的端点上注册一个 ACME 账户。该账户使用的电子邮件地址将作为 ACME 端点发送续期到期等通知的联系地址。

可以通过 Web 界面 Datacenter -> ACME 注册和停用 ACME 账户，也可以使用 pvenode 命令行工具。

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

Tip
由于存在 速率限制，在实验或首次使用 ACME 时应使用 LE staging。

ACME 插件

ACME 插件的任务是自动验证你以及由你操作的 Proxmox VE 集群确实是某个域名的所有者。 这是自动

ACME 协议定义了不同类型的挑战，例如 http-01：Web 服务器提供包含特定内容的文件，以证明它
有时由于技术限制，或记录地址无法从公网访问，这种方式不可行。此时可以使用 dns-01 挑战。
该挑战通过在域名区域中创建特定 DNS 记录来完成。

Proxmox VE 开箱即支持这两种挑战类型。可以通过 Web 界面 Datacenter -> ACME 配置插件，
也可以使用 pvenode acme plugin add 命令配置。

ACME 插件配置存储在 /etc/pve/priv/acme/plugins.cfg 中。插件可供集群中的所有节点使用。

节点域名

每个域名都是特定于节点的。可以在 Node -> Certificates 下添加新域名条目或管理现有条目，
也可以使用 pvenode config 命令。

为节点配置所需域名并确认选择了所需 ACME 账户后，可以通过 Web 界面订购新证书。 成功后，界面
10 秒后重新加载。

续期将 自动进行。

3.12.5 ACME HTTP 挑战插件

始终会隐式配置一个 standalone 插件，用于通过在端口 80 上启动的内置 Web 服务器验证 http-01
挑战。

Note
名称 standalone 表示它可以自行提供验证，而不需要任何第三方服务。因此，该插件也适用于集群

要将其用于 Let’s Encrypt ACME 的证书管理，需要满足几个前提条件。

- 必须接受 Let’s Encrypt 的 ToS 才能注册账户。

- 节点的 端口 **80** 需要可从互联网访问。
- 端口 80 上 不得 有其他监听程序。
- 请求的（子）域名需要解析到该节点的公网 IP。

3.12.6 ACME DNS API 挑战插件

在无法或不希望通过 `http-01` 方法进行外部访问验证的系统上，可以使用 `dns-01` 验证方法。此验证方法要求 DNS 服务器允许通过 API 配置 TXT 记录。

配置用于验证的 **ACME DNS API**

Proxmox VE 复用为 `acme.sh` [4] 项目开发的 DNS 插件。有关特定 API 的配置详情，请参阅其文档。

使用 DNS API 配置新插件的最简单方式是通过 Web 界面（`Datacenter -> ACME`）。

选择 DNS 作为挑战类型。然后可以选择 API 提供方，并输入通过其 API 访问账户所需的凭据数据。验证延迟决定了设置 DNS 记录到提示 ACME 提供方进行验证之间的时间（秒），因为提供方通常需要

Tip
关于如何获取提供方 API 凭据的更多详细信息，请参见 acme.sh How to use DNS API wiki。

由于 DNS 提供方和 API 端点众多，Proxmox VE 会为部分提供方自动生成凭据表单。 对于其他提供方 KEY=VALUE 对复制进去。

通过 **CNAME** 别名进行 **DNS** 验证

如果主/真实 DNS 不支持通过 API 配置记录，可以使用特殊的 `alias` 模式在另一个域名/DNS 服务器上处理验证。 手动为 `_acme-challenge.domain1.example` 设置一条永久 CNAME 记录，使其 `_acme-challenge.domain2.example`，并在 Proxmox VE 节点配置文件中相应 `acmedomainX` 键上将 `alias` 属性设置为 `domain2.example`，即可允许 `domain2.example` 的 DNS 服务器验证 `domain1.example` 的所有挑战。

插件组合

如果节点可通过多个域名访问，而这些域名具有不同要求或 DNS 配置能力，则可以组合使用 `http-01` 和 `dns-01` 验证。 也可以通过为每个域名指定不同插件实例，混合使用来自多个提供方或实例的 DNS API。

Tip
通过多个域名访问同一服务会增加复杂性，应尽可能避免。

---

[4] acme.sh https://github.com/acmesh-official/acme.sh

3.12.7 ACME 证书自动续期

如果节点已成功配置由 ACME 提供的证书（无论通过 pvenode 还是 GUI），该证书将由 pve-daily-
自动续期。目前，如果证书已经过期，或将在未来 30 天内过期，将尝试续期。

Note
如果使用会颁发短期证书的自定义目录，建议禁用 pve-daily-update.timer
单元的随机延迟， 以避免重启后错过证书续期。

3.12.8 使用 pvenode 的 ACME 示例

示例：用于 **Let’s Encrypt** 证书的 **pvenode** 调用示例

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

示例：设置 **OVH API** 以验证域名

Note
无论使用哪种插件，账户注册步骤都相同，此处不再重复。

Note
OVH_AK 和 OVH_AS 需要根据 OVH API 文档从 OVH 获取。

首先需要获取所有信息，以便你和 Proxmox VE 能够访问该 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”:”ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ”,”state”:” ↩
   pendingValidation”,”validationUrl”:”https://eu.api.ovh.com/auth/? ↩
   credentialToken= ↩
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA”}

b''（b''b''打b''b''开b''b''验b''b''证b'' URLb''，b''b''并b''b''按b''b''照b''b' ↩
   ’说b''b''明b''b''将b'' Application Key b''与b''b''账b''b''户b''/Consumer  ↩
   Key b''关b''b''联b''b''）b''

root@proxmox:~# echo ”OVH_CK=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ” >> /path/to/ ↩
   api-token
```

现在可以设置 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
b''┌b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┬b''b' ↩
   ’─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ’─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ’─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ’─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ’─b''b''─b''b''┐b''
b''│b'' key    b''│b'' value                                        b''│b''
b''╞b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''╡b''b' ↩
   ’=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b' ↩
   ’=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b' ↩
   ’=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b' ↩
   ’=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b''=b''b' ↩
   ’=b''b''=b''b''┤b''
```

```
b''│b'' api    b''│b'' ovh                                          b''│b''
b''├b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┼b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''┤b''
b''│b'' data   b''│b'' OVH_AK=XXXXXXXXXXXXXXXX                     b''│b''
b''│b''        b''│b'' OVH_AS=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY  b''│b''
b''│b''        b''│b'' OVH_CK=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ  b''│b''
b''├b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┼b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''┤b''
b''│b'' digest b''│b'' 867fcf556363ca1bea866863093fcab83edf47a1 b''│b''
b''├b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┼b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''┤b''
b''│b'' plugin b''│b'' example_plugin                               b''│b''
b''├b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┼b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''┤b''
b''│b'' type   b''│b'' dns                                          b''│b''
b''└b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''┴b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b''─b''b' ↩
   ''─b''b''─b''b''┘b''
```

最后，可以配置要为其获取证书的域名，并为其提交证书订单：

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

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

示例：从 **staging** 切换到常规 **ACME** 目录

不支持更改某个账户的 ACME 目录，但由于 Proxmox VE 支持多个账户，可以直接创建一个以生产（可
目录作为端点的新账户。 也可以停用 staging 账户并重新创建它。

示例：使用 **pvenode** 将 **default ACME** 账户从 **staging** 切换到正式目录

```
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
```

3.13 主机引导加载器

Proxmox VE 当前会根据安装程序中选择的磁盘设置，使用两种引导加载器之一。

对于以 ZFS 作为根文件系统安装的 EFI 系统，除非启用了 Secure Boot，否则会使用 `systemd-boot`。GRUB 引导加载器（这通常也适用于安装在 Debian 之上的系统）。

目前在OpenEuler版本上，本工具不可用。

3.13.1 安装程序使用的分区方案

Proxmox VE 安装程序会在所有被选作安装目标的磁盘上创建 3 个分区。

创建的分区包括：

- 一个 1 MB 的 BIOS Boot Partition（gdisk 类型 EF02）
- 一个 512 MB 的 EFI System Partition（ESP，gdisk 类型 EF00）
- 第三个分区，占用设定的 `hdsize` 参数范围，或占用所选存储类型使用的剩余空间

使用 ZFS 作为根文件系统的系统，会通过存储在 512 MB EFI System Partition 上的 内核和 initrd 镜像启动。对于传统 BIOS 系统以及启用了 Secure Boot 的 EFI 系统， 会使用 GRUB；对于未启用 Secure Boot 的 EFI 系统，会使用 `systemd-boot`。两者 都会被安装并配置为指向 ESP。

在所有使用 GRUB 启动的系统上，BIOS 模式的 GRUB（`--target i386-pc`）会安装到 所有所选磁盘 BIOS Boot Partition 上 [5]。

3.13.2 使用 `proxmox-boot-tool` 同步 ESP 内容

`proxmox-boot-tool` 是一个用于确保 EFI System Partition 内容正确配置并保持同步的 工具。它会将 ESP，并配置相应的引导加载器，使其从格式化为 `vfat` 的 ESP 启动。在以 ZFS 作为根文件系统的场景 所有可选功能，而无需受限于 GRUB 中 ZFS 实现也支持的子集，也无需创建单独的小型 boot-pool [6].

在具备冗余的设置中，安装程序会在所有磁盘上划分 ESP。这样即使第一个启动设备故障， 或者 BIOS 只能从特定磁盘启动，系统仍能启动。

在常规运行期间，ESP 不会保持挂载状态。这样有助于在系统崩溃时避免格式化为 `vfat` 的 ESP 文件系统损坏，也避免在主启动设备故障时需要手动调整 `/etc/fstab`。

`proxmox-boot-tool` 处理以下任务：

- 格式化并设置新分区
- 将新的内核镜像和 initrd 镜像复制并配置到所有列出的 ESP
- 在内核升级和其他维护任务期间同步配置

[5] 这些包括根文件系统位于 `ext4` 或 `xfs` 的所有安装，以及非 EFI 系统上根文件系统位于 ZFS 的安装

[6] 使用 GRUB 从 ZFS 根文件系统启动 https://openzfs.github.io/openzfs-docs/Getting%20Started/Debian/Debian%20Bookworm%20Root%20on%20ZFS.html

- 管理要同步的内核版本列表
- 配置引导加载器以启动特定内核版本（固定）

可以运行以下命令查看当前已配置的 ESP 及其状态：

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

设置新分区以作为同步 **ESP** 使用

要将某个分区格式化并初始化为同步 ESP，例如在 rpool 中更换故障 vdev 后，或将早于 同步机制的现
`proxmox-kernel-helper` 提供的 `proxmox-boot-tool`。

Warning
`format` 命令会格式化 `<partition>`，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：

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

要将位于 `/dev/sda2` 且未挂载的现有 ESP 纳入 Proxmox VE 的内核更新同步机制，请使用：

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

或：

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

用于强制使用 GRUB 而不是 `systemd-boot` 初始化，例如用于支持 Secure Boot。

之后，`/etc/kernel/proxmox-boot-uuids` 中应包含一行新增分区的 UUID。`init` 命令 还会自动
ESP 的刷新。

更新所有 **ESP** 上的配置

要复制并配置所有可启动内核，并保持 `/etc/kernel/proxmox-boot-uuids` 中列出的所有 ESP
同步，只需运行：

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

（等同于在根文件系统为 `ext4` 或 `xfs` 的系统上运行 `update-grub`）。

如果更改了内核命令行，或希望同步所有内核和 initrd，则需要执行此操作。

Note
`update-initramfs` 和 `apt`（必要时）都会自动触发刷新。

由 **proxmox-boot-tool** 考虑的内核版本

默认配置以下内核版本：

- 当前正在运行的内核
- 软件包更新中新安装的版本
- 已安装的最新两个内核
- 倒数第二个内核系列的最新版本（例如 5.0、5.3），如适用
- 任何手动选择的内核

手动保持某个内核可启动

如果希望将某个内核和 initrd 镜像添加到可启动内核列表，请使用 `proxmox-boot-tool kernel add`。

例如，运行以下命令可将 ABI 版本为 `5.0.15-1-pve` 的内核添加到需要保留安装并同步到 所有 ESP 的内核列表：

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

`proxmox-boot-tool kernel list` 会列出当前选定用于启动的所有内核版本：

```
# proxmox-boot-tool kernel list
Manually selected kernels:
5.0.15-1-pve

Automatically selected kernels:
5.0.12-1-pve
4.15.18-18-pve
```

运行 `proxmox-boot-tool kernel remove` 可从手动选择的内核列表中移除某个内核，例如：

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

> **Note**
> 在按上述方式手动添加或移除内核后，需要运行 `proxmox-boot-tool refresh` 来更新 所有 EFI System Partition（ESP）。

常用命令示例

以下命令可用于查看 ESP 同步状态、列出已选择的内核，并在更改后刷新 ESP 配置：

```
# proxmox-boot-tool status
# proxmox-boot-tool kernel list
# proxmox-boot-tool refresh
```

3.13.3 确定正在使用的引导加载器

确定正在使用哪个引导加载器的最简单且最可靠的方法，是观察 Proxmox VE 节点的启动过程。

你会看到 GRUB 的蓝色界面，或者简洁的黑白 systemd-boot 界面。

从正在运行的系统判断引导加载器可能并非 100% 准确。最安全的方式是运行以下命令：

```
# efibootmgr -v
```

如果返回 EFI 变量不受支持的信息，则说明正在以 BIOS/Legacy 模式使用 GRUB。

如果输出包含类似以下内容的行，则说明正在以 UEFI 模式使用 GRUB。

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

如果输出包含类似以下内容的行，则说明正在使用 systemd-boot。

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

运行：

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

可以确认是否已配置 proxmox-boot-tool，这能够较好地反映系统的启动方式。

3.13.4 GRUB

多年来，GRUB 一直是启动 Linux 系统的事实标准，并且文档相当完善 [7].

配置

对 GRUB 配置的更改通过默认文件 /etc/default/grub 或 /etc/default/grub.d 中的 配置片段 [8]

```
# update-grub
```

3.13.5 Systemd-boot

systemd-boot 是轻量级 EFI 引导加载器。它会直接从安装位置所在的 EFI Service Partition（ESP）initrd 镜像。直接从 ESP 加载内核的主要优势在于， 无需重新实现用于访问存储的驱动程序。在 Proxmox VE 中， proxmox-boot-tool 用于保持 ESP 上的配置同步。

[7] GRUB Manual https://www.gnu.org/software/grub/manual/grub/grub.html

[8] 使用 proxmox-boot-tool 的系统会在执行 update-grub 时调用 proxmox-boot-tool refresh。

配置

systemd-boot 通过 EFI System Partition（ESP）根目录中的 loader/loader.conf 文件进行配置。loader.conf(5) 手册页。

每个引导加载器条目都会放在 loader/entries/ 目录下的独立文件中。

entry.conf 示例类似如下（/ 表示 ESP 的根目录）：

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

3.13.6 编辑内核命令行

可以根据所使用的引导加载器，在以下位置修改内核命令行：

GRUB

内核命令行需要放在 /etc/default/grub 文件中的 GRUB_CMDLINE_LINUX_DEFAULT 变量内。运行 update-grub 会将其内容追加到 /boot/grub/grub.cfg 中所有 linux 条目。

Systemd-boot

内核命令行需要作为一行写入 /etc/kernel/cmdline。要应用更改，请运行 proxmox-boot-tool refresh，它会将该内容设置为 loader/entries/proxmox-*.conf 中所有配置文件的 option 行。

完整的内核参数列表可在以下地址找到：*https://www.kernel.org/doc/html/v<YOUR-KERNEL-VERSION>/admin-guide/kernel-parameters.html*. 请将 <YOUR-KERNEL-VERSION> 替换为 major.minor 版本。例如，对于基于 6.5 版本的内核，URL 为：https://www.kernel.org/doc/html/v6.5/admin-guide/kernel-parameters.html

可以通过查看 Web 界面（Node -> Summary）或运行以下命令来查找内核版本：

```
# uname -r
```

使用输出开头的前两个数字。

3.13.7 为下次启动覆盖内核版本

要选择当前默认内核之外的内核，可以采用以下方式之一：

- 使用启动过程开始时显示的引导加载器菜单
- 使用 proxmox-boot-tool 将系统一次性或永久（直到重置固定设置）pin 到某个内核版本。

这有助于规避较新内核版本与硬件之间的不兼容问题。

Note
应尽快移除此类固定设置，以便最新内核中的所有当前安全补丁也能应用到系统。

例如：若要永久选择版本 5.15.30-1-pve 用于启动，请运行：

```
# proxmox-boot-tool kernel pin 5.15.30-1-pve
```

Tip
固定功能适用于所有 Proxmox VE 系统，不仅限于使用 proxmox-boot-tool 同步 ESP 内容的系统。如果系统没有使用 proxmox-boot-tool 进行同步，也可以跳过最后的 proxmox-boot-tool refresh 调用。

也可以设置只在下一次系统启动时引导某个内核版本。例如，这可用于测试更新后的内核 是否已解决最 pin 某个版本的问题：

```
# proxmox-boot-tool kernel pin 5.15.30-1-pve --next-boot
```

要移除任何已固定的版本配置，请使用 unpin 子命令：

```
# proxmox-boot-tool kernel unpin
```

虽然 unpin 也有 --next-boot 选项，但它用于清除通过 --next-boot 设置的固定版本。 由于该操作
设置或清除固定版本后，还需要运行 refresh 子命令来同步 ESP 上的内容和配置。

Tip
如果以交互方式调用该工具，对于由 proxmox-boot-tool 管理的系统，会提示你自动执行此操作。

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

3.13.8 Secure Boot

自 Proxmox VE 8.1 起，通过签名软件包以及与 proxmox-boot-tool 的集成，Secure Boot 可开箱即用。
Secure Boot 正常工作需要以下软件包。可以使用 proxmox-secure-boot-support 元软件包一次性

- shim-signed（由 Microsoft 签名的 shim 引导加载器）
- shim-helpers-amd64-signed（由 Proxmox 签名的 fallback 引导加载器和 MOKManager）
- grub-efi-amd64-signed（由 Proxmox 签名的 GRUB EFI 引导加载器）
- proxmox-kernel-6.X.Y-Z-pve-signed（由 Proxmox 签名的内核镜像）

开箱即用的引导加载器仅支持 GRUB，因为其他引导加载器当前不符合 Secure Boot 代码签名条件。
任何新的 Proxmox VE 安装都会自动包含上述所有软件包。
关于 Secure Boot 工作方式以及如何自定义设置的更多详情，请参见 我们的 wiki。

将现有安装切换到 **Secure Boot**

> **Warning**
> 如果操作不正确，在某些情况下这可能导致安装无法启动。重新安装主机时，如果 Secure Boot 可用，会自动完成设置，无需额外交互。请确保你拥有可用且经过充分测试的 **Proxmox VE** 主机备份！

如有需要，可以将现有 UEFI 安装切换到 Secure Boot，而无需从头重新安装 Proxmox VE。

首先，确保整个系统均为最新状态。然后安装 `proxmox-secure-boot-support`。GRUB 会自动创建 shim 启动所需的 EFI 启动项。

systemd-boot

如果使用 `systemd-boot` 作为引导加载器（参见 确定正在使用的引导加载器），则需要一些额外设置。只有在 Proxmox VE 以 ZFS-on-root 方式安装时才会如此。

要检查后一种情况，请运行：

```
# findmnt /
```

如果主机确实使用 ZFS 作为根文件系统，则 FSTYPE 列应包含 `zfs`：

```
TARGET SOURCE           FSTYPE OPTIONS
/      rpool/ROOT/pve-1 zfs    rw,relatime,xattr,noacl,casesensitive
```

接下来，必须找到合适的潜在 ESP（EFI system partition）。可以使用如下 `lsblk` 命令完成：

```
# lsblk -o +FSTYPE
```

输出应类似如下：

```
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS FSTYPE
sda      8:0    0   32G  0 disk
b''├b''b''─b''sda1   8:1    0 1007K  0 part
b''├b''b''─b''sda2   8:2    0  512M  0 part             vfat
b''└b''b''─b''sda3   8:3    0 31.5G  0 part             zfs_member
sdb      8:16   0   32G  0 disk
b''├b''b''─b''sdb1   8:17   0 1007K  0 part
b''├b''b''─b''sdb2   8:18   0  512M  0 part             vfat
b''└b''b''─b''sdb3   8:19   0 31.5G  0 part             zfs_member
```

在此示例中，分区 `sda2` 和 `sdb2` 是目标。可以通过它们的 512M 大小以及 FSTYPE 为 `vfat` 来识别；这里对应的是一个 ZFS RAID-1 安装。

必须使用 `proxmox-boot-tool` 正确设置这些分区，以便通过 GRUB 启动。以下命令（以 `sda2` 为例）必须分别针对每个 ESP 单独运行：

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

之后，可以运行以下命令对设置进行基本检查：

```
# efibootmgr -v
```

该列表应包含类似如下的条目：

```
[..]
Boot0009* proxmox       HD(2,GPT,..,0x800,0x100000)/File(\EFI\proxmox\ ↩
   shimx64.efi)
[..]
```

Note
旧的 `systemd-boot` 引导加载器会被保留，但会优先使用 GRUB。这样，如果由于任何原因无法在 Secure Boot 模式下使用 GRUB 启动，仍可在关闭 Secure Boot 后使用 `systemd-boot` 启动系统。

现在可以重启主机，并在 UEFI 固件设置工具中启用 Secure Boot。

重启后，UEFI 固件启动菜单中应可选择一个名为 `proxmox` 的新条目，它会使用预签名的 EFI shim 启动。

如果由于任何原因在 UEFI 启动菜单中找不到 `proxmox` 条目，可以尝试手动添加（如果固件支持），将 `\EFI\proxmox\shimx64.efi` 添加为自定义启动项。

Note
已知某些 UEFI 固件会在重启时丢弃 `proxmox` 启动选项。如果 `proxmox` 启动项指向某个磁盘上的 GRUB 安装，但该磁盘本身不是启动选项，就可能发生这种情况。如果可行，请尝试在 UEFI 固件设置工具中将该磁盘添加为启动选项，然后再次运行 `proxmox-boot-tool`。

Tip
如需注册自定义密钥，请参见配套的 Secure Boot wiki 页面。

在 **Secure Boot** 下使用 **DKMS**/第三方模块

在启用了 Secure Boot 的系统上，内核会拒绝加载未由受信任密钥签名的模块。内核软件包 随附的默认

要加载其他模块，例如使用 DKMS 构建或手动构建的模块，需要用 Secure Boot 栈信任的 密钥对其签 `mokutil` 将这些密钥注册为 Machine Owner Key（MOK）。

`dkms` 工具会自动在 `/var/lib/dkms/mok.key` 和 `/var/lib/dkms/mok.pub` 中生成密钥对和证书，并用其为它构建和安装的内核模块签名。

可以使用以下命令查看证书内容：

```
# openssl x509 -in /var/lib/dkms/mok.pub -noout -text
```

并使用以下命令将其注册到系统中：

```
# mokutil --import /var/lib/dkms/mok.pub
input password:
input password again:
```

mokutil 命令会要求输入两次（临时）密码，该密码需要在流程的下一步中再输入一次。 重启系统后 MOKManager EFI 二进制程序，它允许你验证密钥/证书， 并使用通过 mokutil 开始注册时选择的密码 DKMS 构建的模块（这些模块使用已注册的 MOK 签名）。如有需要，MOK 也可用于签名自定义 EFI 二进制文件和内核镜像。

同一流程也可用于不由 DKMS 管理的自定义/第三方模块，但在这种情况下，需要手动完成 密钥/证书生

3.14 Kernel Samepage Merging (KSM)

Kernel Samepage Merging（KSM）是 Linux 内核提供的一项可选内存去重功能， 在 Proxmox VE 中默认启用。KSM 的工作方式是扫描一段物理内存页，查找内容相同的页面， 并识别映射到这使它们都指向同一个物理页，并释放旧页面。虚拟页会被标记为“写时复制”， 因此对它们的任何写入都

3.14.1 KSM 的影响

在虚拟化环境中，KSM 可以优化内存使用，因为运行相似操作系统或工作负载的多台虚拟机 可能共享大

不过，虽然 KSM 能降低内存使用量，它也带来一些安全风险，因为它可能使虚拟机暴露于 侧信道攻击。KSM 的某些特性，通过同一主机上的另一台虚拟机推断 正在运行的虚拟机中的信息。

因此，如果使用 Proxmox VE 提供托管服务，应考虑禁用 KSM，以便为用户提供额外的安全性。此外，还应检查所在国家或地区的法规，因为禁用 KSM 可能是法律要求。

3.14.2 禁用 KSM

要查看 KSM 是否处于活动状态，可以检查以下命令的输出：

```
# systemctl status ksmtuned
```

如果处于活动状态，可以使用以下命令立即禁用：

```
# systemctl disable --now ksmtuned
```

最后，要取消当前所有已合并页面的合并，请运行：

```
# echo 2 > /sys/kernel/mm/ksm/run
```

常用命令示例：

```
systemctl status ksmtuned
cat /sys/kernel/mm/ksm/run
cat /sys/kernel/mm/ksm/pages_shared
```

Chapter 4

图形用户界面

Proxmox VE 使用简单，无需安装单独的管理工具，所有操作都可以通过 Web 浏览器完成（建议使用最
Firefox 或 Google Chrome）。内置 HTML5 控制台可用于访问客户机控制台。也可以改用 SPICE。

由于使用 Proxmox cluster file system（pmxcfs），你可以连接到任意节点来管理整个集群。
每个节点都可以管理整个集群，因此不需要专用管理节点。

可以使用任何现代浏览器访问基于 Web 的管理界面。当 Proxmox VE 检测到你从移动设备连接时，
会重定向到更简单的触控式用户界面。

Web 界面可通过 https://youripaddress:8006 访问（默认登录用户为 *root*，密码在安装 过程中指定

4.1 功能

- 对 Proxmox VE 集群进行无缝集成和管理
- 使用 AJAX 技术动态更新资源
- 通过 SSL 加密（https）安全访问所有虚拟机和容器
- 快速、搜索驱动的界面，可处理数百台甚至可能上千台 VM
- 安全的 HTML5 控制台或 SPICE
- 面向所有对象（VM、存储、节点等）的基于角色的权限管理
- 支持多个认证源（例如 local、MS ADS、LDAP 等）
- 双因素认证（OATH、Yubikey）
- 基于 ExtJS 7.x JavaScript 框架

4.2 登录

连接到服务器时，首先会看到登录窗口。Proxmox VE 支持多种认证后端（*Realm*），也可以在此处
选择语言。GUI 已翻译为 20 多种语言。

Note
可以勾选底部复选框，将用户名保存在客户端。这样下次登录时可以减少输入。

4.3 GUI 概览

Proxmox VE 用户界面由四个区域组成。

页眉	位于顶部。显示状态信息，并包含最重要操作的按钮。
资源树	位于左侧。可在该导航树中选择特定对象。
内容面板	位于中间区域。所选对象的配置选项和状态会显示在这里。
日志面板	位于底部。显示最近任务的日志条目。可以双击这些日志条目查看更多详情，或中止正在运行的任务。

Note
可以缩小和展开资源树及日志面板，也可以完全隐藏日志面板。在小屏幕上工作、需要更多空间查看其他内容时，这会很有帮助。

4.3.1 页眉

左上角首先看到的是 Proxmox 标志。旁边是当前运行的 Proxmox VE 版本。在附近的搜索栏中，可以搜索特定对象（VM、容器、节点等）。有时这比在资源树中选择对象更快。

页眉右侧包含四个按钮：

Documentation	打开新的浏览器窗口，显示参考文档。
Create VM	打开虚拟机创建向导。
Create CT	打开容器创建向导。
User Menu	显示当前登录用户身份；点击后会打开包含用户专属选项的菜单。在用户菜单中，可以找到提供本地 UI 设置的 *My Settings* 对话框。其下方有 *TFA*（双因素认证）和 *Password* 自助服务快捷入口。还可以找到用于更改 *Language* 和 *Color Theme* 的选项。菜单底部是 *Logout* 选项。

4.3.2 我的设置

My Settings 窗口允许设置本地保存的设置。其中包括 *Dashboard Storages*，可启用或禁用特定存储等同于启用每一个存储。

在仪表板设置下方，可以看到已保存的用户名、清除用户名的按钮，以及将 GUI 中所有布局重置为默认值的按钮。

右侧是 *xterm.js Settings*，包含以下选项：

Font-Family	xterm.js 中使用的字体（例如 Arial）。
Font-Size	首选字体大小。
Letter Spacing	增大或减小文本中字母之间的间距。
Line Height	指定一行的绝对高度。

4.3.3 资源树

这是主导航树。树顶部可以选择一些预定义视图，这会改变下方树的结构。默认视图是 **Server View**

Datacenter	包含集群范围设置（与所有节点相关）。
Node	表示集群中的主机，客户机在其上运行。
Guest	VM、容器和模板。
Storage	数据存储。
Pool	可以使用池对客户机分组，以简化管理。

可使用以下视图类型：

Server View	显示所有类型对象，并按节点分组。
Folder View	显示所有类型对象，并按对象类型分组。
Pool View	显示 VM 和容器，并按池分组。
Tag View	显示 VM 和容器，并按标签分组。

4.3.4 日志面板

日志面板的主要用途是显示集群中当前正在发生的事情。创建新 VM 等操作会在后台执行，这类后台作
task。

此类任务的任何输出都会保存到单独的日志文件中。只需双击任务日志条目即可查看该日志。 也可以在

请注意，这里会显示所有集群节点上的最新任务。因此可以实时看到其他人何时正在另一个集群节点上
工作。

Note
为保持列表简短，我们会从日志面板中移除较旧且已完成的任务。但仍可在节点面板的 *Task History* 中找到这些任务。

一些短时操作只会向所有集群成员发送日志。可以在 *Cluster log* 面板中看到这些消息。

4.4 内容面板

在资源树中选择项目时，对应对象会在内容面板中显示配置和状态信息。以下章节会简要概述这些功能。更多详细信息请参阅参考文档中的相应章节。

4.4.1 数据中心

在数据中心级别，可以访问集群范围的设置和信息。

- **Search:** 对节点、VM、容器、存储设备和池执行集群范围搜索。
- **Summary:** 简要概览集群健康状态和资源使用情况。
- **Cluster:** 提供创建或加入集群所需的功能和信息。
- **Options:** 查看并管理集群范围默认设置。
- **Storage:** 提供管理集群存储的界面。
- **Backup:** 计划备份任务。该功能在集群范围内运行，因此调度时无需关心 VM/容器位于集群中的哪个节点。
- **Replication:** 查看并管理复制任务。
- **Permissions:** 管理用户、组和 API token 权限，以及 LDAP、MS-AD 和双因素认证。
- **HA:** 管理 Proxmox VE High Availability。
- **ACME:** 为服务器节点设置 ACME（Let’s Encrypt）证书。
- **Firewall:** 配置 Proxmox Firewall，并创建集群范围模板。
- **Metric Server:** 为 Proxmox VE 定义外部指标服务器。
- **Notifications:** 配置 Proxmox VE 的通知行为和目标。
- **Support:** 显示支持订阅信息。

4.4.2 节点

可以在该级别分别管理集群中的节点。

顶部页眉包含 *Reboot*、*Shutdown*、*Shell*、*Bulk Actions* 和 *Help* 等常用按钮。 *Shell* 包含 *noVNC*、*SPICE* 和 *xterm.js* 选项。*Bulk Actions* 包含 *Bulk Start*、*Bulk Shutdown* 和 *Bulk Migrate* 选项。

- **Search:** 在节点中搜索 VM、容器、存储设备和池。
- **Summary:** 简要显示节点资源使用情况。
- **Notes:** 使用 Markdown 语法编写自定义备注。
- **Shell:** 访问节点的 shell 界面。
- **System:** 配置网络、DNS 和时间设置，并访问 syslog。
- **Updates:** 升级系统并查看可用的新软件包。
- **Firewall:** 管理特定节点的 Proxmox Firewall。
- **Disks:** 概览已连接磁盘，并管理其使用方式。
- **Ceph:** 仅在主机上安装了 Ceph server 时使用。在这种情况下，可以在此管理 Ceph 集群并查看其状态。
- **Replication:** 查看并管理复制任务。
- **Task History:** 查看历史任务列表。
- **Subscription:** 上传订阅密钥，并生成用于支持案例的系统报告。

4.4.3 客户机

客户机有两种不同类型，并且都可以转换为模板。一种是 Kernel-based Virtual Machine（KVM），另一种是 Linux Container（LXC）。两者的导航大体相同，仅部分选项不同。

要访问各种客户机管理界面，请从左侧菜单选择 VM 或容器。

页眉包含电源管理、迁移、控制台访问和类型、克隆、HA、帮助等命令。其中一些按钮带有下拉 菜单 *Shutdown* 还包含其他电源选项，*Console* 包含不同控制台类型：*SPICE*、 *noVNC* 和 *xterm.js*。

右侧面板会显示左侧菜单中所选项目对应的界面。

可用界面如下。

- **Summary:** 简要概览 VM 活动，并提供用于 Markdown 语法备注的 `Notes` 字段。
- **Console:** 访问 VM/容器的交互式控制台。
- **(KVM)Hardware:** 定义 KVM VM 可用的硬件。
- **(LXC)Resources:** 定义 LXC 可用的系统资源。
- **(LXC)Network:** 配置容器网络设置。

- **(LXC)DNS:** 配置容器 DNS 设置。
- **Options:** 管理客户机选项。
- **Task History:** 查看与所选客户机相关的所有历史任务。
- **(KVM) Monitor:** 与 KVM 进程交互通信的界面。
- **Backup:** 创建和还原系统备份。
- **Replication:** 查看并管理所选客户机的复制任务。
- **Snapshots:** 创建和还原 VM 快照。
- **Firewall:** 在 VM 级别配置防火墙。
- **Permissions:** 管理所选客户机的权限。

4.4.4 存储

与客户机界面类似，存储界面由左侧的存储元素菜单和右侧用于管理这些元素的界面组成。

该视图采用左右两栏的分割视图。左侧是存储选项，右侧显示所选选项的内容。

- **Summary:** 显示存储的重要信息，例如类型、使用情况以及存放的内容。
- **Content:** 针对该存储保存的每种内容类型提供菜单项，例如 Backups、ISO Images、 CT Templates。
- **Permissions:** 管理该存储的权限。

4.4.5 池

同样，池视图由两部分组成：左侧菜单，以及右侧每个菜单项对应的界面。

- **Summary:** 显示池的描述。
- **Members:** 显示并管理池成员（客户机和存储）。
- **Permissions:** 管理池的权限。

4.5 标签

出于组织管理目的，可以为客户机设置 tags。目前这些标签仅为用户提供信息价值。标签会显示在 Web 界面的两个位置：Resource Tree 中，以及选择客户机时的状态行中。

点击客户机状态行中的 pencil 图标，可以添加、编辑和移除标签。按 + 按钮可以添加多个 标签，按 - 按钮可以移除标签。可分别使用 ✓ 和 x 按钮保存或取消更改。

也可以通过 CLI 设置标签，多个标签用分号分隔。例如：

```
# qm set ID --tags 'myfirsttag;mysecondtag'
```

4.5.1 样式配置

默认情况下，标签颜色会根据其文本以确定性方式派生。可以自定义颜色、资源树中的形状、 大小写敏
Web 界面的 *Datacenter → Options → Tag Style Override* 完成，也可以通过 CLI 完成。例如：

```
# pvesh set /cluster/options --tag-style color-map=example:000000:FFFFFF
```

将标签 `example` 的背景颜色设置为黑色（`#000000`），文本颜色设置为白色（`#FFFFFF`）。

4.5.2 权限

默认情况下，在客户机（`/vms/ID`）上拥有 `VM.Config.Options` 权限的用户可以设置任意标签（参见 权限管理）。如果要限制此行为，可以在 *Datacenter → Options → User Tag Access* 下设置相应权限：

- `free`: 用户设置标签不受限制（默认）
- `list`: 用户可以基于预定义标签列表设置标签
- `existing`: 类似 list，但用户也可以使用已经存在的标签
- `none`: 用户被限制，不能使用标签

同样也可以通过 CLI 完成。

请注意，在 `/` 上拥有 `Sys.Modify` 权限的用户始终可以设置或删除任何标签，不受此处设置 影响。此外
`registered tags` 列表，只有在 `/` 上拥有 `Sys.Modify` 权限的用户才能添加和移除其中的标签。reg
tags 列表可以在 *Datacenter → Options → Registered Tags* 下编辑，也可以通过 CLI 编辑。

关于具体选项以及如何在 CLI 中调用它们的更多详情，请参见 数据中心配置。

4.6 同意横幅

可以在 *Datacenter → Options → Consent Text* 中配置自定义同意横幅，用户必须先接受它
才能登录。如果没有内容，则不会显示同意横幅。该文本会以 base64 字符串形式存储在 `/etc/pve/c`
配置文件中。

Chapter 5

集群管理器

Proxmox VE 集群管理器 pvecm 是用于创建一组物理服务器的工具。这类服务器组称为 集群。我们使用 Corosync Cluster Engine 实现可靠的组通信。 集群中的节点数量没有明确的硬性限制。实践中，实际 目前（2021 年）已有使用高端企业硬件、超过 50 个节点的生产集群案例。

pvecm 可用于创建新集群、将节点加入集群、离开集群、获取状态信息，以及执行各种其他集群相关任 **P**rox**m**o**x** **C**luster **F**ile **S**ystem（“pmxcfs”）用于将集群配置透明地分发到所有集群节点。

将节点组织为集群具有以下优势：

- 集中式、基于 Web 的管理
- 多主集群：每个节点都可以执行所有管理任务
- 使用 pmxcfs 这个数据库驱动的文件系统存储配置文件，并通过 corosync 在所有节点上实时复制
- 可在物理主机之间轻松迁移虚拟机和容器
- 快速部署
- 提供防火墙和 HA 等集群范围服务

5.1 要求

- 所有节点之间必须能够通过 UDP 端口 5405-5412 互相连接，corosync 才能正常工作。
- 日期和时间必须同步。
- 节点之间需要 TCP 端口 22 上的 SSH 隧道。
- 如果需要高可用，至少需要三个节点才能获得可靠仲裁。所有节点应使用相同版本。
- 建议为集群流量使用专用 NIC，尤其是在使用共享存储时。
- 添加节点时需要集群节点的 root 密码。
- 虚拟机在线迁移仅在各节点 CPU 来自同一厂商时受支持。其他情况下可能可以工作，但绝不保证。

Note
不能将 Proxmox VE 3.x 及更早版本与 Proxmox VE 4.X 集群节点混用。

Note
虽然可以混用 Proxmox VE 4.4 和 Proxmox VE 5.0 节点，但这种做法不支持作为生产配置，只应在整个集群从一个主版本升级到另一个主版本期间临时使用。

Note
无法将 Proxmox VE 6.x 与更早版本组成集群。Proxmox VE 6.x 与更早版本之间的集群协议（corosync）发生了根本变化。Proxmox VE 5.4 的 corosync 3 软件包仅用于升级到 Proxmox VE 6.0 的过程。

5.2 准备节点

首先，在所有节点上安装 Proxmox VE。确保每个节点都使用最终的主机名和 IP 配置进行安装。集群创建后无法更改主机名和 IP。

通常会在 `/etc/hosts` 中记录所有节点名称及其 IP（或通过其他方式让这些名称可解析），但这并不是通过 SSH 从一个节点连接到另一个节点（另见 Link Address Types）。 请注意，我们始终建议在集群 IP 地址引用节点。

5.3 创建集群

可以在控制台上创建集群（通过 `ssh` 登录），也可以使用 Proxmox VE Web 界面（*Datacenter → Cluster*）通过 API 创建集群。

Note
请为集群使用唯一名称。该名称之后无法更改。集群名称遵循与节点名称相同的规则。

5.3.1 通过 Web GUI 创建

在 *Datacenter → Cluster* 下，点击 **Create Cluster**。输入集群名称，并从下拉列表中选择一个网络 0）。 默认情况下，它使用通过节点主机名解析得到的 IP。

从 Proxmox VE 6.2 开始，一个集群最多可以添加 8 个备用链路。要添加冗余链路，请点击 *Add* 按钮，并在相应字段中选择链路编号和 IP 地址。在 Proxmox VE 6.2 之前，要添加第二条链路作为备用，可以勾选 *Advanced* 复选框，并选择额外网络接口（Link 1，另见 Corosync Redundancy）。

Note
请确保用于集群通信的网络没有用于网络存储或在线迁移等高流量用途。
集群网络本身产生的数据量很小，但对延迟非常敏感。请查看完整的 集群网络要求。

5.3.2 通过命令行创建

通过 ssh 登录到第一个 Proxmox VE 节点，并运行以下命令：

```
hp1# pvecm create CLUSTERNAME
```

要检查新集群状态，请使用：

```
hp1# pvecm status
```

5.3.3 同一网络中的多个集群

可以在同一物理或逻辑网络中创建多个集群。在这种情况下，每个集群都必须拥有唯一名称，以避免集

虽然 corosync 集群的带宽需求相对较低，但数据包延迟和每秒包数（PPS）才是限制因素。同一网络

5.4 向集群添加节点

Caution

加入集群时，/etc/pve 中的所有现有配置都会被覆盖。尤其是，加入的节点不能承载任何客
否则客户机 ID 可能冲突，并且该节点会继承集群的存储配置。若要加入一个已有客户机的节点
可作为变通方式先为每个客户机创建备份（使用 vzdump），加入集群后再用不同 ID
还原。如果该节点的存储布局不同，则需要重新添加该节点的存储，并调整每个存储的节点限
以反映该存储实际在哪些节点上可用。

5.4.1 通过 GUI 将节点加入集群

登录到已有集群节点的 Web 界面。在 *Datacenter → Cluster* 下，点击顶部的 **Join Information** 按钮。然后点击 **Copy Information** 按钮。也可以手动复制 *Information* 字段中的字符串。

接下来，登录要添加节点的 Web 界面。在 *Datacenter → Cluster* 下，点击 **Join Cluster**。将之前复
Join Information 文本填入 *Information* 字段。加入集群所需的大多数设置都会自动填写。出于安全原

Note

要手动输入所有必需数据，可以禁用 *Assisted Join* 复选框。

点击 **Join** 按钮后，集群加入流程会立即开始。节点加入集群后，其当前节点证书会被替换为由集群证书
这意味着当前会话会在几秒后停止工作。之后可能需要强制重新加载 Web 界面，并使用集群凭据重新登

现在，该节点应该可以在 *Datacenter → Cluster* 下看到。

5.4.2 通过命令行将节点加入集群

通过 ssh 登录要加入现有集群的节点。

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

对于 IP-ADDRESS-CLUSTER，请使用现有集群节点的 IP 或主机名。建议使用 IP 地址（见 Link Address Types）。

要检查集群状态，请使用：

```
# pvecm status
```

添加 **4** 个节点后的集群状态

```
 # pvecm status
Cluster information
~~~~~~~~~~~~~~~~~~~
Name:             prod-central
Config Version:   3
Transport:        knet
Secure auth:      on

Quorum information
~~~~~~~~~~~~~~~~~~
Date:             Tue Sep 14 11:06:47 2021
Quorum provider:  corosync_votequorum
Nodes:            4
Node ID:          0x00000001
Ring ID:          1.1a8
Quorate:          Yes

Votequorum information
~~~~~~~~~~~~~~~~~~~~~~
Expected votes:   4
Highest expected: 4
Total votes:      4
Quorum:           3
Flags:            Quorate

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

如果只想获取所有节点列表，请使用：

```
# pvecm nodes
```

列出集群中的节点

```
 # pvecm nodes

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

5.4.3 使用独立集群网络添加节点

向使用独立集群网络的集群添加节点时，需要使用 *link0* 参数设置该节点在该网络上的地址：

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

如果要使用 Kronosnet 传输层内置的 冗余，还需要使用 *link1* 参数。

使用 GUI 时，可以在 **Cluster Join** 对话框中相应的 *Link X* 字段选择正确接口。

5.5 移除集群节点

Caution
继续之前请仔细阅读该流程，因为它可能并不是你想要或需要的操作。

将该节点上的所有虚拟机移走。确保已复制任何需要保留的本地数据或备份。 此外，确保移除所有指向

Caution
如果在移除节点之前没有移除指向该节点的复制任务，会导致复制任务变得无法移除。
尤其要注意，如果迁移已复制的虚拟机，复制会自动切换方向；因此，从待删除节点迁移已复
复制任务会自动设置为指向该节点。

如果待移除节点配置了 Ceph：

1. 确保仍有足够的 Proxmox VE 节点运行 OSD（up 且 in）。

   Note
   默认情况下，Ceph 池的 size/min_size 为 3/2，并在对象均衡器 CRUSH 中以完整节点作为 failure domain。 因此，如果在线且运行 OSD 的节点少于 size（3），数据冗余会降级。 如果在线节点少于 min_size，池 I/O 会被阻塞，受影响的客户机可能崩溃。

---

2. 确保仍有足够的 monitors、 managers，以及在使用 CephFS 时的 metadata servers 可用。
3. 为维持数据冗余，每次销毁 OSD（尤其是节点上的最后一个 OSD）都会触发数据再均衡。因此，请确保剩余节点上的 OSD 有足够可用空间。
4. 要从待删除节点移除 Ceph，请先逐个 销毁 其 OSD。
5. 一旦 CEPH status 再次变为 HEALTH_OK，继续执行：
    1. 通过 Web 界面的 *Ceph → CephFS* 或运行以下命令，销毁其 metadata server：
    ```
    # pveceph mds destroy <local hostname>
    ```
    2. 销毁其 monitor
    3. 销毁其 manager
6. 最后，运行以下命令从 CRUSH 层次结构中移除现在已经为空的 bucket（待移除的 Proxmox VE 节点）：
    ```
    # ceph osd crush remove <hostname>
    ```

在以下示例中，我们将从集群中移除节点 hp4。

登录到*另一个*集群节点（不是 hp4），并执行 `pvecm nodes` 命令以确定要移除的节点 ID：

```
 hp1# pvecm nodes

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

此时，必须关闭 hp4，并确保它不会以当前配置再次在该网络中启动。

Important
如上所述，在移除*之前*关闭节点非常关键，并且要确保它不会以当前配置在现有集群网络中
如果按原样启动该节点，集群可能损坏，并且可能难以恢复到可用状态。

关闭节点 hp4 后，即可安全地将其从集群中移除。

```
 hp1# pvecm delnode hp4
 Killing node 4
```

Note
此时可能会收到 `Could not kill node (error = CS_ERR_NOT_EXIST)` 错误消息。这并不表示节点删除实际失败，而只是 corosync 尝试 kill 一个离线节点时失败。因此可以安全忽略。

再次使用 pvecm nodes 或 pvecm status 检查节点列表。它应类似如下：

```
hp1# pvecm status

...

Votequorum information
~~~~~~~~~~~~~~~~~~~~~~
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate

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

如果出于任何原因想让该服务器再次加入同一集群，必须：

- 在其上全新安装 Proxmox VE，
- 然后按上一节说明将其加入集群。

已移除节点的配置文件仍会保留在 */etc/pve/nodes/hp4* 中。请恢复仍需要的任何配置， 之后移除该目

Note
移除节点后，其 SSH 指纹仍会保留在其他节点的 *known_hosts* 中。 如果使用相同 IP 或主机名重新加入节点后收到 SSH 错误，请在重新添加的节点上运行一次 pvecm updatecerts，以在集群范围内更新其指纹。

5.5.1 不重装而分离节点

Caution
这*不是*推荐方法，请谨慎操作。如果不确定，请使用前一种方法。

也可以在不从头重装的情况下将节点从集群中分离出来。但从集群移除该节点后，它仍然可以访问任何
在开始从集群移除节点之前，必须先解决这一点。一个 Proxmox VE 集群不能与另一个集群共享完全相
因为存储锁无法跨集群边界工作。此外，这还可能导致 VMID 冲突。

建议创建一个新存储，仅允许要分离的节点访问。举例来说，这可以是 NFS 上的新导出， 也可以是新的
Ceph 池。关键是不能让多个集群访问完全相同的存储。设置好该存储后， 将该节点上的所有数据和虚

Warning
请确保所有共享资源都已清晰分离！否则会遇到冲突和问题。

首先，在该节点上停止 corosync 和 pve-cluster 服务：

```
systemctl stop pve-cluster
systemctl stop corosync
```

以本地模式重新启动集群文件系统：

```
pmxcfs -l
```

删除 corosync 配置文件：

```
rm /etc/pve/corosync.conf
rm -r /etc/corosync/*
```

现在可以再次将文件系统作为普通服务启动：

```
killall pmxcfs
systemctl start pve-cluster
```

该节点现在已经从集群中分离。可以在集群中任一剩余节点上使用以下命令将其删除：

```
pvecm delnode oldnode
```

如果命令因剩余节点丢失仲裁而失败，可以将预期票数设置为 1 作为临时处理：

```
pvecm expected 1
```

然后再次执行 *pvecm delnode* 命令。

现在切回已分离的节点，删除其上所有剩余的集群文件。这可以确保该节点之后能顺利加入其他集群。

```
rm /var/lib/corosync/*
```

由于其他节点的配置文件仍在集群文件系统中，你可能也希望清理它们。在完全确认节点名称正确后，可以直接从 */etc/pve/nodes/NODENAME* 递归移除整个目录。

Caution
该节点的 SSH 密钥会保留在 *authorized_key* 文件中。这意味着节点之间仍然可以使用公钥认证互相连接。应通过从 */etc/pve/priv/authorized_keys* 文件中移除相应密钥来修正这一点。

5.6 仲裁

Proxmox VE 使用基于仲裁的技术，在所有集群节点之间提供一致状态。

> 仲裁是分布式事务为了获准在分布式系统中执行某项操作而必须获得的最小票数。
>
> — 来自 Wikipedia *Quorum (distributed computing)*

在发生网络分区时，状态变更要求多数节点在线。如果集群失去仲裁，就会切换到只读模式。

Note
Proxmox VE 默认给每个节点分配一票。

5.7 集群网络

集群网络是集群的核心。所有通过它发送的消息都必须按各自顺序可靠地传递到所有节点。 在 Proxmox VE 中，这部分由 corosync 完成；corosync 是一个高性能、低开销、高可用开发工具包的实现。它服务于我们的去中心化配置文件系统（`pmxcfs`）。

5.7.1 网络要求

Proxmox VE 集群栈要求所有节点之间具备可靠网络，并且延迟低于 5 毫秒（LAN 性能），才能稳定运在节点数量较少的配置中，延迟更高的网络_可能_可以工作，但这并不保证； 当节点超过三个且延迟高10 ms 时，成功可能性会明显降低。

该网络不应被其他成员大量使用，因为 corosync 虽然不需要太多带宽，但对延迟抖动很敏感； 理想情应运行在物理隔离的专用网络上。尤其不要让 corosync 和存储共享同一网络（除非在 冗余配置中作为

设置集群前，最佳实践是检查网络是否适合该用途。为确保各节点能在集群网络上互相连接， 可以使用`ping` 工具测试它们之间的连通性。

如果启用了 Proxmox VE 防火墙，将自动生成 corosync 的 ACCEPT 规则，无需手动操作。

Note
Corosync 在 3.0 版本之前使用组播（该版本在 Proxmox VE 6.0 中引入）。 现代版本依赖 Kronosnet 进行集群通信，目前它只支持常规 UDP 单播。

Caution
仍然可以在 corosync.conf 中将 transport 设置为 udp 或 udpu 来启用组播或旧式单播，但请记住，这会禁用所有加密和冗余支持。
因此不推荐这样做。

5.7.2 独立集群网络

在不带任何参数创建集群时，corosync 集群网络通常会与 Web 界面和虚拟机网络共享。 根据配置，甚corosync 是对时间敏感的实时应用。

设置新网络

首先，必须设置新的网络接口。它应位于物理隔离的网络上。请确保该网络满足 集群网络要求。

在创建集群时分离

可以通过用于创建新集群的 *pvecm create* 命令的 *linkX* 参数实现。

如果已经设置了一个静态地址为 10.10.10.1/25 的额外 NIC，并希望通过该接口收发所有集群通信，则可以执行：

```
pvecm create test --link0 10.10.10.1
```

要检查一切是否正常工作，请执行：

```
systemctl status corosync
```

之后，按上文说明继续 使用独立集群网络添加节点。

在创建集群后分离

如果已经创建了集群，并希望在不重建整个集群的情况下将通信切换到另一个网络，可以执行该操作。由于节点必须重启 corosync，并在新网络上逐个恢复运行，此变更可能导致集群短暂丢失仲裁。

请先查看如何 编辑 corosync.conf 文件。 然后打开该文件，应能看到类似如下内容：

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

nodelist {

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

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

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

```
  }

}

quorum {
  provider: corosync_votequorum
}

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

}
```

Note
ringX_addr 实际上指定的是 corosync 链路地址。名称 "ring" 是旧版 corosync 遗留下来的叫法，为向后兼容而保留。

首先，如果节点条目中尚未包含 *name* 属性，请添加它们。这些属性*必须*与节点名称匹配。

然后，将所有节点 *ring0_addr* 属性中的地址替换为新地址。这里可以使用普通 IP 地址或主机名。如果使用主机名，请确保所有节点都能解析这些主机名（另见 Link Address Types）。

在本示例中，我们希望将集群通信切换到 10.10.10.0/25 网络，因此相应修改每个节点的 *ring0_addr*

Note
完全相同的流程也可用于更改其他 *ringX_addr* 值。不过建议一次只更改一个链路地址，这样在出现问题时更容易恢复。

增加 *config_version* 属性后，新的配置文件应如下所示：

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

nodelist {

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

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

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

}

quorum {
  provider: corosync_votequorum
}

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

}
```

然后，最后检查所有变更信息是否正确，保存文件，并再次按照 编辑 corosync.conf 文件一节使其生效

这些变更会实时应用，因此并不严格要求重启 corosync。如果还更改了其他设置， 或注意到 corosyn 报错，可以选择触发一次重启。

在单个节点上执行：

```
systemctl restart corosync
```

现在检查一切是否正常：

```
systemctl status corosync
```

如果 corosync 重新开始工作，也在所有其他节点上重启它。 随后这些节点会在新网络上逐个加入集群

5.7.3 Corosync 地址

corosync 链路地址（为向后兼容，在 `corosync.conf` 中表示为 *ringX_addr* ）可以通过两种方式指定

- **IPv4/v6** 地址可以直接使用。建议使用它们，因为它们是静态的，通常不会被随意更改。

- 主机名会使用 getaddrinfo 解析，这意味着默认情况下，如果 IPv6 地址可用，会优先使用 IPv6 地址（另见 man gai.conf）。请记住这一点，尤其是在将现有集群升级到 IPv6 时。

Caution
应谨慎使用主机名，因为它们解析到的地址可以在不触碰 corosync 或其所在节点的情况下被更改，这可能导致地址变更时没有考虑对 corosync 的影响。

如果偏好使用主机名，建议为 corosync 使用单独的静态主机名。同时确保集群中的每个节点都能正确

自 Proxmox VE 5.1 起，虽然仍支持主机名，但主机名会在录入时解析，只有解析后的 IP 会保存到配置

在更早版本中加入集群的节点，可能仍在 corosync.conf 中使用未解析的主机名。按上文所述，将其 IP 或单独主机名可能是个好做法。

5.8 Corosync 冗余

Corosync 默认通过其集成的 Kronosnet 层支持冗余网络（旧式 udp/udpu 传输不支持）。可以通过 pvecm 的 *--linkX* 参数、在 GUI 中指定 **Link 1**（创建集群或添加新节点时），或在 corosync.conf 中指定多个 *ringX_addr*。

Note
为提供有效故障切换，每条链路都应位于自己的物理网络连接上。

链路会根据优先级设置使用。可以通过在 corosync.conf 中相应 interface 段设置 *knet_link_priori* 来配置优先级，或者更推荐在使用 pvecm 创建集群时使用 *priority* 参数：

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

这会使 *link1* 优先使用，因为它拥有更高优先级。

如果没有手动配置优先级（或两条链路优先级相同），链路会按编号顺序使用，编号较低者优先级更高

即使所有链路都正常工作，也只有最高优先级链路会承载 corosync 流量。链路优先级不能混用，这意

由于低优先级链路只有在所有更高优先级链路都失败时才会承载流量，因此将用于其他任务（虚拟机、高延迟或更拥塞的连接也可能好过完全没有连接。

5.8.1 向现有集群添加冗余链路

要向运行中的配置添加新链路，请先查看如何 编辑 corosync.conf 文件。

然后，在 nodelist 段中为每个节点添加新的 *ringX_addr*。确保为所有节点添加时使用相同的 *X*，并且该地址对每个节点都是唯一的。

最后，如下所示，在 totem 段中添加新的 *interface*，并将 *X* 替换为上面选择的链路编号。

假设添加的是编号为 1 的链路，新的配置文件可能如下所示：

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

nodelist {

  node {
    name: due
    nodeid: 2
    quorum_votes: 1
    ring0_addr: 10.10.10.2
    ring1_addr: 10.20.20.2
  }

  node {
    name: tre
    nodeid: 3
    quorum_votes: 1
    ring0_addr: 10.10.10.3
    ring1_addr: 10.20.20.3
  }

  node {
    name: uno
    nodeid: 1
    quorum_votes: 1
    ring0_addr: 10.10.10.1
    ring1_addr: 10.20.20.1
  }

}

quorum {
  provider: corosync_votequorum
}

totem {
  cluster_name: testcluster
  config_version: 4
  ip_version: ipv4-6
  secauth: on
  version: 2
  interface {
    linknumber: 0
  }
  interface {
    linknumber: 1
  }
}
```

按照 编辑 corosync.conf 文件的最后步骤操作后，新链路会立即启用。 应不需要重启。可以使用以下命

corosync 是否加载了新链路：

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

可以通过临时断开一个节点上的旧链路，并确认断开期间其状态仍保持在线，来测试新链路：

```
pvecm status
```

如果看到健康的集群状态，说明正在使用新链路。

5.9 SSH 在 Proxmox VE 集群中的作用

Proxmox VE 将 SSH 隧道用于多种功能。

- 代理控制台/shell 会话（节点和客户机）
  连接到节点 A 时使用节点 B 的 shell，会连接到节点 A 上的终端代理， 该代理再通过非交互式 SSH 隧道连接到节点 B 上的登录 shell。
- 在 *secure* 模式下迁移虚拟机和 CT 内存及本地存储。
  迁移期间，会在源节点和目标节点之间建立一个或多个 SSH 隧道，用于交换迁移信息并传输内存和磁
- 存储复制

5.9.1 SSH 设置

在 Proxmox VE 系统上，会对 SSH 配置/设置进行以下更改：

- `root` 用户的 SSH 客户端配置会设置为优先使用 AES 而不是 ChaCha20
- `root` 用户的 `authorized_keys` 文件会链接到 `/etc/pve/priv/authorized_keys`，以合并集
- `sshd` 会配置为允许使用密码以 root 身份登录

Note
较旧系统还可能将 `/etc/ssh/ssh_known_hosts` 设置为指向 `/etc/pve/priv/known_hosts` 的符号链接，其中包含所有节点主机密钥的合并版本。该机制已在 `pve-cluster <<INSERT VERSION>>` 中由显式 host key pinning 取代；如果该符号链接仍存在，可以运行 `pvecm updatecerts --unmerge-known-hosts` 解除配置。

5.9.2 自动执行 .bashrc 及类似文件带来的陷阱

如果存在自定义 .bashrc，或配置的 shell 在登录时会执行的类似文件，ssh 会在会话成功建立后自动
这可能导致一些意外行为，因为这些命令可能会在上述任一操作中以 root 权限执行， 从而可能产生有

为避免这类复杂情况，建议在 /root/.bashrc 中添加检查，确保会话是交互式会话后， 才运行 .bashrc 命令。

可以将以下片段添加到 .bashrc 文件开头：

```
# Early exit if not running interactively to avoid side-effects!
case $- in
    *i*) ;;
      *) return;;
esac
```

5.10 Corosync 外部投票支持

本节介绍在 Proxmox VE 集群中部署外部投票者的方法。配置后，集群可以承受更多节点故障，同时不破坏集群通信的安全属性。

该功能涉及两个服务：

- 运行在每个 Proxmox VE 节点上的 QDevice 守护进程
- 运行在独立服务器上的外部投票守护进程

这样即使在较小配置中（例如 2+1 节点），也可以实现更高可用性。

5.10.1 QDevice 技术概览

Corosync Quorum Device（QDevice）是运行在每个集群节点上的守护进程。 它会根据外部运行的
其主要用途是让集群能够承受超过标准仲裁规则允许数量的节点故障。这样做是安全的， 因为外部设备
只有当该节点集合在获得第三方投票后能够（重新）达到仲裁时，才会执行这一操作。

目前，仅支持 *QDevice Net* 作为第三方仲裁者。它是一个守护进程：如果能够通过网络访问某个集群分
就向该分区提供一票。在任意时刻，它只会给集群的一个分区投票。 它设计为支持多个集群，并且几乎
QDevice 的主机上不需要配置文件。

外部主机的唯一要求是能够通过网络访问集群，并且有可用的 corosync-qnetd 软件包。 我们为基于 Debian 的主机提供软件包，其他 Linux 发行版也应可通过各自的软件包管理器获得相应软件包。

Note
与 corosync 本身不同，QDevice 通过 TCP/IP 连接到集群。 该守护进程也可以运行在集群 LAN 之外，并不受 corosync 低延迟要求限制。

5.10.2 支持的配置

我们支持在偶数节点集群中使用 QDevice；如果 2 节点集群需要提供更高可用性，也建议使用它。
对于奇数节点集群，目前不建议使用 QDevice。原因在于 QDevice 为不同集群类型提供的票数不同。
偶数节点集群会获得额外的一票，这只会提高可用性，因为如果 QDevice 本身失败，你所处的位置与
QDevice 时相同。

另一方面，对于奇数节点规模的集群，QDevice 会提供 *(N-1)* 票，其中 *N* 对应集群节点数。这种替代
该算法允许除一个节点外的所有节点（自然也包括 QDevice 本身）发生故障。不过，这有两个缺点：

- 如果 QNet 守护进程本身失败，则不能再有任何其他节点失败，否则集群会立即失去仲裁。例如，在
  15 节点集群中，集群失去仲裁前可以有 7 个节点失败。但如果这里配置了 QDevice，且 QDe-
  vice 本身失败，则 15 个节点中任何单个节点都不能失败。在这种情况下，QDevice 几乎相当于单点
- 除一个节点外所有节点加 QDevice 都可以失败，这一点乍看很有吸引力，但它可能导致 HA
  服务大规模恢复，从而压垮唯一剩余的节点。此外，如果只剩下 *((N-1)/2)* 个或更少节点在线，Ceph
  服务器会停止提供服务。

如果理解这些缺点和影响，可以自行决定是否在奇数节点集群配置中使用该技术。

5.10.3 QDevice-Net 设置

建议将任何为 corosync-qdevice 提供票数的守护进程以非特权用户身份运行。Proxmox VE 和
Debian 提供的软件包已经配置为这样运行。守护进程与集群之间的流量必须加密，以确保 QDe-
vice 安全地集成到 Proxmox VE 中。

首先，在外部服务器上安装 *corosync-qnetd* 软件包：

```
external# apt install corosync-qnetd
```

并在所有集群节点上安装 *corosync-qdevice* 软件包：

```
pve# apt install corosync-qdevice
```

完成后，确保集群中的所有节点都在线。

现在可以在其中一个 Proxmox VE 节点上运行以下命令设置 QDevice：

```
pve# pvecm qdevice setup <QDEVICE-IP>
```

集群中的 SSH 密钥会自动复制到 QDevice。

> **Note**
> 请确保在外部服务器上为 root 用户设置基于密钥的访问，或在设置阶段临时允许 root
> 使用密码登录。如果此阶段收到类似 *Host key verification failed.* 的错误，运行 `pvecm updatecerts` 可能可以修复该问题。

所有步骤成功完成后，会看到 ”Done”。可以使用以下命令验证 QDevice 是否已设置：

```
pve# pvecm status

...

Votequorum information
~~~~~~~~~~~~~~~~~~~~~~
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate Qdevice

Membership information
~~~~~~~~~~~~~~~~~~~~~~
    Nodeid      Votes    Qdevice Name
    0x00000001      1    A,V,NMW 192.168.22.180 (local)
    0x00000002      1    A,V,NMW 192.168.22.181
    0x00000000      1            Qdevice
```

QDevice 状态标志

如上所示，QDevice 的状态输出通常包含三列：

- A / NA：Alive 或 Not Alive。表示与外部 corosync-qnetd 守护进程的通信是否正常。
- V / NV：QDevice 是否会为该节点投票。在脑裂情况下，如果节点之间的 corosync 连接中断，但它们仍都能与外部 corosync-qnetd 守护进程通信，则只有一个节点会获得投票。
- MW / NMW：Master wins（MV）或 not（NMW）。默认值为 NMW，见 [1]。
- NR：QDevice 未注册。

Note
如果 QDevice 显示为 Not Alive（上方输出中的 NA），请确保外部服务器的端口 5403（qnetd 服务器默认端口）可通过 TCP/IP 访问。

5.10.4 常见问题

平局决胜

当出现平局时，即两个大小相同的集群分区无法互相看到、但都能看到 QDevice，QDevice 会随机选

[1] votequorum_qdevice_master_wins manual page https://manpages.debian.org/bookworm/-libvotequorum-dev/votequorum_qdevice_master_wins.3.en.html

可能的负面影响

对于偶数节点集群，使用 QDevice 没有负面影响。如果它无法工作，效果与完全没有 QDevice 相同。

QDevice 设置后添加/删除节点

如果要在已配置 QDevice 的集群中添加新节点或移除现有节点，需要先移除 QDevice。 之后即可正常 QDevice。

移除 **QDevice**

如果使用官方 pvecm 工具添加了 QDevice，可以通过运行以下命令将其移除：

```
pve# pvecm qdevice remove
```

5.11 Corosync 配置

/etc/pve/corosync.conf 文件在 Proxmox VE 集群中具有核心作用。它控制集群成员关系和集群网 更多信息请查看 corosync.conf 手册页：

```
man corosync.conf
```

对于节点成员关系，应始终使用 Proxmox VE 提供的 pvecm 工具。 对于其他变更，可能需要手动编辑配

5.11.1 编辑 corosync.conf

编辑 corosync.conf 文件并不总是很直接。每个集群节点上有两个该文件： 一个在 /etc/pve/coros /etc/corosync/corosync.conf。 编辑集群文件系统中的文件会将变更传播到本地文件，但反过来

文件一旦变更，配置就会自动更新。这意味着能够集成到运行中 corosync 的变更会立即生效。 因此，

```
cp /etc/pve/corosync.conf /etc/pve/corosync.conf.new
```

然后，使用喜欢的编辑器打开配置文件，例如每个 Proxmox VE 节点上预安装的 nano 或 vim.tiny。

Note
配置变更后始终递增 *config_version* 数字；遗漏这一步可能导致问题。

完成必要变更后，再创建一份当前工作配置文件的副本。如果新配置应用失败或导致其他问题， 该副本

```
cp /etc/pve/corosync.conf /etc/pve/corosync.conf.bak
```

然后使用新配置文件替换旧配置文件：

```
mv /etc/pve/corosync.conf.new /etc/pve/corosync.conf
```

可以使用以下命令检查变更是否已自动应用：

```
systemctl status corosync
journalctl -b -u corosync
```

如果变更无法自动应用，可能需要通过以下命令重启 corosync 服务：

```
systemctl restart corosync
```

如发生错误，请查看下面的故障排查章节。

5.11.2 故障排查

问题：***quorum.expected_votes must be configured***

当 corosync 启动失败，并在系统日志中看到以下消息时：

```
[...]
corosync[1647]:  [QUORUM] Quorum provider: corosync_votequorum failed to  ↩
   initialize.
corosync[1647]:  [SERV  ] Service engine 'corosync_quorum' failed to load  ↩
   for reason
    'configuration error: nodelist or quorum.expected_votes must be  ↩
       configured!'
[...]
```

这意味着配置中为 corosync *ringX_addr* 设置的主机名无法解析。

在无仲裁时写入配置

如果需要在无仲裁节点上更改 */etc/pve/corosync.conf*，并且清楚自己在做什么，请使用：

```
pvecm expected 1
```

这会将预期票数设置为 1，并使集群达到仲裁。随后可以修复配置，或将其还原到最后一个可工作的备

如果 corosync 已经无法启动，这还不够。在这种情况下，最好编辑 */etc/corosync/corosync.conf* 中 corosync 配置的本地副本，让 corosync 能重新启动。 请确保所有节点上的该配置内容一致，以避

5.11.3 Corosync 配置术语表

ringX_addr
: 这表示节点之间 Kronosnet 连接的不同链路地址。

5.12 集群冷启动

显然，当所有节点都离线时，集群不具备仲裁。这是断电后常见的情况。

Note
使用不间断电源（“UPS”，也称为 “battery backup”）来避免这种状态始终是好做法，尤其是在需要 HA 时。

节点启动时，`pve-guests` 服务会启动并等待仲裁。一旦具备仲裁，它会启动所有设置了 `onboot` 标志的客户机。

打开节点电源，或断电后恢复供电时，某些节点可能会比其他节点启动得更快。 请记住，客户机启动会

5.13 客户机 VMID 自动选择

创建新客户机时，Web 界面会自动向后端请求一个空闲 VMID。默认搜索范围为 `100` 到 `1000000`（低
schema 强制执行的最大允许 VMID）。

有时管理员希望在单独范围内分配新 VMID，例如便于将临时虚拟机与手动选择 VMID 的虚拟机区分开。
有时只是希望提供长度稳定的 VMID；例如将下边界设置为 `100000`，可留出更多空间。

为满足这一用例，可以通过 `datacenter.cfg` 配置文件设置下边界、上边界或同时设置两者。 该文件
Web 界面 *Datacenter* → *Options* 下编辑。

Note
该范围仅用于 next-id API 调用，因此不是硬性限制。

5.14 客户机迁移

将虚拟客户机迁移到其他节点是集群中的一项有用功能。有一些设置可以控制这类迁移的行为。 可以通
`datacenter.cfg` 配置文件进行设置，也可以针对特定迁移通过 API 或命令行参数设置。

客户机处于在线还是离线状态，或者是否具有本地资源（如本地磁盘），都会造成差异。

关于虚拟机迁移的详细信息，请参见 QEMU/KVM Migration Chapter。

关于容器迁移的详细信息，请参见 Container Migration Chapter。

5.14.1 迁移类型

迁移类型定义迁移数据应通过加密（`secure`）通道还是未加密（`insecure`）通道发送。 将迁移类型
`insecure` 意味着虚拟客户机的 RAM 内容也会以未加密方式传输， 这可能导致客户机内部关键数据（

因此，如果不能完全控制网络，且无法保证无人窃听，强烈建议使用安全通道。

Note
存储迁移不遵循该设置。目前，它始终通过安全通道发送存储内容。

加密需要大量计算能力，因此该设置经常被改为 insecure 以获得更好性能。 现代系统的影响较低，因
AES 加密。在高速网络中， 性能影响尤其明显，例如可以传输 10 Gbps 或更高的网络。

5.14.2 迁移网络

默认情况下，Proxmox VE 使用承载集群通信的网络发送迁移流量。这并不理想， 因为敏感的集群流量

设置迁移网络参数后，可以为所有迁移流量使用专用网络。除内存外，这也会影响离线迁移的存储流量

迁移网络以 CIDR 表示法作为网络来设置。这样做的优点是无需为每个节点设置单独 IP 地址。 Prox-
mox VE 可以根据 CIDR 形式指定的网络，确定目标节点上的实际地址。要启用这一点， 必须以每个节
IP 的方式指定网络。

示例

假设有一个三节点配置，并具有三个独立网络。一个用于与 Internet 的公共通信， 一个用于集群通信

这种配置的网络配置可能如下所示：

```
iface eno1 inet manual

# public network
auto vmbr0
iface vmbr0 inet static
    address 192.X.Y.57/24
    gateway 192.X.Y.1
    bridge-ports eno1
    bridge-stp off
    bridge-fd 0

# cluster network
auto eno2
iface eno2 inet static
    address  10.1.1.1/24

# fast network
auto eno3
iface eno3 inet static
    address  10.1.2.1/24
```

这里，我们将网络 10.1.2.0/24 用作迁移网络。对于单次迁移，可以使用命令行工具的 migration_n
参数完成：

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

要将其配置为集群中所有迁移的默认网络，请设置 /etc/pve/datacenter.cfg 文件的 migration
属性：

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

Note
在 /etc/pve/datacenter.cfg 中设置迁移网络时，必须始终设置迁移类型。

5.14.3 常用命令示例

以下命令可用于查看集群、节点和迁移相关状态：

```
# pvecm status
# pvecm nodes
# qm migrate <vmid> <target-node> --online
```

Chapter 6

Proxmox Cluster File System (pmx-cfs)

Proxmox Cluster file system（“pmxcfs”）是一个由数据库驱动的文件系统，用于存储配置文件，并
corosync 实时复制到所有集群节点。系统使用它来 存储所有与 Proxmox VE 相关的配置文件。

虽然该文件系统会将所有数据存储在磁盘上的持久数据库中，但数据副本也驻留在 RAM 中。因此最大大
128 MiB。对于存储数千台虚拟机的配置， 这仍然足够。

该系统提供以下优势：

- 将所有配置实时无缝复制到所有节点
- 提供强一致性检查，避免 VM ID 重复
- 当节点失去 quorum 时进入只读状态
- 自动将 corosync 集群配置更新到所有节点
- 包含分布式锁机制

6.1 POSIX 兼容性

该文件系统基于 FUSE，因此行为类似 POSIX。但有些功能并未实现，因为系统并不需要：

- 可以生成普通文件和目录，但不能生成符号链接，等等
- 不能重命名非空目录（因为这样更容易保证 VMID 唯一）。
- 不能更改文件权限（权限基于路径）
- 0_EXCL 创建操作不是原子的（类似旧版 NFS）
- 0_TRUNC 创建操作不是原子的（FUSE 限制）

6.2 文件访问权限

所有文件和目录的所有者均为用户 root，所属组为 www-data。只有 root 拥有写权限，但 www-data 组可以读取大多数文件。以下路径下的文件仅 root 可访问：

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

6.3 技术

系统使用 Corosync Cluster Engine 进行集群通信，并使用 SQlite 作为数据库文件。该文件系统通过 FUSE 在用户空间实现。

6.4 文件系统布局

该文件系统挂载在：

```
/etc/pve
```

6.4.1 文件

authkey.pub	ticket 系统使用的公钥
ceph.conf	Ceph 配置文件（注意：/etc/ceph/ceph.conf 是指向该文件的符号链接）
corosync.conf	Corosync 集群配置文件（在 Proxmox VE 4.x 之前，该文件名为 cluster.conf）
datacenter.cfg	Proxmox VE 数据中心范围配置（键盘布局、代理等）
domains.cfg	Proxmox VE 认证域
firewall/cluster.fw	应用于所有节点的防火墙配置
firewall/<NAME>.fw	单个节点的防火墙配置
firewall/<VMID>.fw	虚拟机和容器的防火墙配置
ha/crm_commands	显示 CRM 当前正在执行的 HA 操作
ha/manager_status	集群上 HA 服务的 JSON 格式信息
ha/resources.cfg	由高可用管理的资源及其当前状态
nodes/<NAME>/config	节点专用配置
nodes/<NAME>/lxc/<VMID>.conf	LXC 容器的 VM 配置数据
nodes/<NAME>/openvz/	在 Proxmox VE 4.0 之前用于容器配置数据（已弃用，即将移除）
nodes/<NAME>/pve-ssl.key	pve-ssl.pem 的私有 SSL 密钥
nodes/<NAME>/pve-ssl.pem	Web 服务器的公共 SSL 证书（由集群 CA 签名）

nodes/<NAME>/pveproxy-ssl.key	pveproxy-ssl.pem 的私有 SSL 密钥（可选）
nodes/<NAME>/pveproxy-ssl.pem	Web 服务器的公共 SSL 证书（链）（用于可选覆盖 pve-ssl.pem）
nodes/<NAME>/qemu-server/<VMID>.conf	KVM 虚拟机的 VM 配置数据
priv/authkey.key	ticket 系统使用的私钥
priv/authorized_keys	用于认证的集群成员 SSH 密钥
priv/ceph*	Ceph 认证密钥及相关能力
priv/known_hosts	用于验证的集群成员 SSH 密钥
priv/lock/*	各服务用于确保集群范围操作安全的锁文件
priv/pve-root-ca.key	集群 CA 的私钥
priv/shadow.cfg	PVE Realm 用户的 shadow 密码文件
priv/storage/<STORAGE-ID>.pw	以明文包含存储密码
priv/tfa.cfg	Base64 编码的双因素认证配置
priv/token.cfg	所有 token 的 API token secret
pve-root-ca.pem	集群 CA 的公共证书
pve-www.key	用于生成 CSRF token 的私钥
sdn/*	Software Defined Networking（SDN）的共享配置文件
status.cfg	Proxmox VE 外部指标服务器配置
storage.cfg	Proxmox VE 存储配置
user.cfg	Proxmox VE 访问控制配置（用户/组等）
virtual-guest/cpu-models.conf	用于存储自定义 CPU 模型
vzdump.cron	集群范围的 vzdump 备份作业计划

6.4.2 符号链接

集群文件系统中的某些目录使用符号链接，以指向节点自身的配置文件。因此，下表中 指向的文件在集

local	nodes/<LOCAL_HOST_NAME>
lxc	nodes/<LOCAL_HOST_NAME>/lxc/
openvz	nodes/<LOCAL_HOST_NAME>/openvz/ (deprecated, removed soon)
qemu-server	nodes/<LOCAL_HOST_NAME>/qemu-server/

6.4.3 用于调试的特殊状态文件（JSON）

.version	文件版本（用于检测文件修改）
.members	集群成员信息
.vmlist	所有虚拟机列表
.clusterlog	集群日志（最近 50 条）
.rrd	RRD 数据（最近条目）

6.4.4 启用/禁用调试

可以通过以下命令启用详细 syslog 消息：

```
echo ”1” >/etc/pve/.debug
```

并通过以下命令禁用详细 syslog 消息：

```
echo ”0” >/etc/pve/.debug
```

常用命令示例：

```
systemctl status pve-cluster
pvecm status
```

6.5 恢复

如果 Proxmox VE 主机出现严重问题，例如硬件故障，复制 pmxcfs 数据库文件 /var/lib/pve-clu
并将其移动到新的 Proxmox VE 主机可能会有所帮助。 在新主机上（没有运行任何内容时），需要停止
pve-cluster 服务并替换 config.db 文件（所需权限为 0600）。随后，根据丢失的 Proxmox
VE 主机调整 /etc/hostname 和 /etc/hosts，然后重启并检查（不要忘记 VM/CT 数据）。

6.5.1 移除集群配置

建议的方式是在将节点从集群中移除后重新安装该节点。这可以确保所有机密的 集群/ssh 密钥以及任何

在某些情况下，可能希望在不重新安装的情况下将节点恢复为本地模式，相关内容见 Separate A Node Without Reinstalling

6.5.2 从故障节点恢复/移动客户机

对于 nodes/<NAME>/qemu-server/（虚拟机）和 nodes/<NAME>/lxc/（容器）中的 客户机配置文
VE 会将包含这些文件的节点 <NAME> 视为相应客户机的 所有者。该概念允许使用本地锁，而不必使用成
客户机配置更改。

因此，如果某个客户机的所有者节点发生故障（例如断电、fencing 事件等），常规迁移将无法进行（
所有者节点上取得这种本地锁。对于由 HA 管理的客户机，这不是问题，因为 Proxmox VE 的 High
Availability 栈包含必要的（集群范围）锁和 watchdog 功能，可确保从被 fenced 的节点正确、自动

如果一个未由 HA 管理的客户机只有共享磁盘（且没有其他仅在故障节点上可用的 本地资源），则可以
/etc/pve/ 中 故障节点目录移动到在线节点目录即可（这会更改该客户机的逻辑所有者或位置）。

例如，要将 ID 为 100 的虚拟机从离线的 node1 恢复到另一个节点 node2，可在集群任一成员节点上以
root 身份运行以下命令：

```
mv /etc/pve/nodes/node1/qemu-server/100.conf /etc/pve/nodes/node2/ ↩
   qemu-server/
```

Warning
以这种方式手动恢复客户机之前，务必确认故障源节点确实已经关机或 被 fenced。否则，mv 命令会破坏 Proxmox VE 的锁原则，可能产生不可预期的后果。

Warning
带有本地磁盘（或其他仅在离线节点上可用的本地资源）的客户机无法
通过这种方式恢复。请等待故障节点重新加入集群，或从备份恢复这些客户机。

Chapter 7

Proxmox VE 存储

Proxmox VE 的存储模型非常灵活。虚拟机镜像既可以存储在一个或多个本地存储上，也可以存储在 NFS 或 iSCSI（NAS、SAN）等共享存储上。系统没有固定数量限制，可以按需配置任意数量的存储池 Linux 可用的所有存储技术都可以使用。

将虚拟机存储在共享存储上的一个主要优势是，可以在不停机的情况下实时迁移正在运行的机器，因为集群中的所有节点都可以直接访问虚拟机磁盘镜像。这种情况下无需复制虚拟机镜像数据，因此实

存储库（软件包 `libpve-storage-perl`）使用灵活的插件系统，为所有存储类型提供统一接口。将来可以很容易地扩展以包含更多存储类型。

7.1 存储类型

存储类型基本分为两大类：

文件级存储
: 基于文件级的存储技术允许访问功能完整的（POSIX）文件系统。一般来说，它们比任何块级存储可能是最先进的系统，并且完整支持快照和克隆。

块级存储
: 允许存储大型 raw 镜像。通常无法在这类存储类型上存储其他文件（ISO、备份等）。大多数现代和 GlusterFS 是分布式系统，会将存储数据复制到不同节点。

Table 7.1: 可用存储类型

说明	插件类型	级别	共享	快照	稳定性
ZFS（本地）	`zfspool`	both[1]	no	yes	yes
目录	`dir`	file	no	no[2]	yes
BTRFS	`btrfs`	file	no	yes	technology preview
NFS	`nfs`	file	yes	no[2]	yes
CIFS	`cifs`	file	yes	no[2]	yes
Proxmox Backup	`pbs`	both	yes	n/a	yes
GlusterFS	`glusterfs`	file	yes	no[2]	yes

Table 7.1: (continued)

说明	插件类型	级别	共享	快照	稳定性
CephFS	cephfs	file	yes	yes	yes
LVM	lvm	block	no[3]	no	yes
LVM-thin	lvmthin	block	no	yes	yes
iSCSI/kernel	iscsi	block	yes	no	yes
iSCSI/libiscsi	iscsidirect	block	yes	no	yes
Ceph/RBD	rbd	block	yes	yes	yes
ZFS over iSCSI	zfs	block	yes	yes	yes

[1]: 虚拟机磁盘镜像存储在提供块设备功能的 ZFS volume（zvol）数据集中。

[2]: 在基于文件的存储上，可以通过 qcow2 格式实现快照。

[3]: 可以在基于 iSCSI 或 FC 的存储之上使用 LVM。这样可以获得 shared LVM 存储。

7.1.1 精简配置

许多存储以及 QEMU 镜像格式 qcow2 都支持“精简配置”。启用精简配置后，只有客户机系统实际使用

例如，创建一台带 32GB 硬盘的虚拟机，安装客户机操作系统后，虚拟机根文件系统中包含 3GB 数据。在这种情况下，即使客户机虚拟机看到的是 32GB 硬盘，也只有 3GB 会写入存储。 通过这种方式较大的磁盘镜像，并在需要时为存储添加更多磁盘，而无需调整虚拟机文件系统大小。

所有具备“快照”特性的存储类型也都支持精简配置。

Caution
如果某个存储被写满，所有使用该存储上卷的客户机都会收到 IO 错误。这可能导致文件系统不一致，并可能损坏数据。因此，建议避免过度配置存储资源，或仔细监控可用空间以避免此类情况。

7.2 存储配置

所有与 Proxmox VE 相关的存储配置都保存在单个文本文件 /etc/pve/storage.cfg 中。 由于该文件位于 /etc/pve/ 内，它会自动分发到所有集群节点。因此，所有节点共享同一份 存储配置。

对于共享存储，共享存储配置是很自然的，因为所有节点都能访问同一个“共享”存储。 但这对本地存储也有效，但物理上是不同的，也可能包含完全不同的内容。

7.2.1 存储池

每个存储池都有一个 <type>，并通过其 <STORAGE_ID> 唯一标识。存储池配置如下所示：

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

<type>: <STORAGE_ID> 行开始一个存储池定义，随后是属性列表。大多数属性需要值。 有些属性具

更具体地说，可以查看安装后的默认存储配置。它包含一个名为 local 的特殊本地存储池， 指向目录 /var/lib/vz，并且始终可用。Proxmox VE 安装程序会根据安装时选择的存储类型 创建额外的存储

默认存储配置（**/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
```

Caution

让多个存储配置指向完全相同的底层存储会带来问题。这类 *aliased* 存储配置可能导致两个不同的卷 ID（volid）指向同一个磁盘镜像。Proxmox VE 期望镜像的卷 ID 是唯一的。为 *aliased* 存储配置选择不同内容类型在某些情况下可行，但不推

7.2.2 通用存储属性

一些存储属性在不同存储类型之间通用。

nodes

可以使用或访问该存储的集群节点名称列表。可以使用该属性将存储访问限制在有限的一组节点上

content

一个存储可以支持多种内容类型，例如虚拟磁盘镜像、cdrom ISO 镜像、容器模板或容器根目录。并非所有存储类型都支持所有内容类型。可以设置该属性以选择该存储的用途。

images

QEMU/KVM 虚拟机镜像。

rootdir
允许存储容器数据。

vztmpl
容器模板。

backup
备份文件（vzdump）。

iso
ISO 镜像

snippets
片段文件，例如客户机 hook 脚本

shared
表示这是一个在所有节点（或 nodes 选项列出的所有节点）上内容相同的单一存储。 它不会让本

disable
可以使用该标志完全禁用存储。

maxfiles
已弃用，请改用 prune-backups。每台虚拟机的最大备份文件数量。使用 0 表示不限制。

prune-backups
备份保留选项。详情请参见 备份保留。

format
默认镜像格式（raw|qcow2|vmdk）

preallocation
文件型存储上 raw 和 qcow2 镜像的预分配模式（off|metadata|falloc|full）。 默认值为 metadata，对于 raw 镜像会按 off 处理。当网络存储与大型 qcow2 镜像配合使用时，使用 off 有助于避免超时。

Warning
不建议在不同 Proxmox VE 集群上使用同一个存储池。某些存储操作需要对存储进行独占访问 因此需要正确的锁定机制。该机制已在集群内部实现，但不能跨不同集群工作。

7.3 卷

系统使用一种特殊表示法来定位存储数据。从存储池分配数据时，会返回这样的卷标识符。 卷由 <STOR 标识，后接依赖于存储类型的卷名称，中间以冒号分隔。 有效的 <VOLUME_ID> 如下所示：

```
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
```

要获取 <VOLUME_ID> 的文件系统路径，请使用：

```
pvesm path <VOLUME_ID>
```

7.3.1 卷所有权

image 类型卷存在所有权关系。每个此类卷都归属于一台虚拟机或一个容器。 例如，卷 local:230/e
归属于虚拟机 230。大多数存储后端会将该所有权信息 编码到卷名称中。

删除虚拟机或容器时，系统也会删除其拥有的所有关联卷。

7.4 使用命令行界面

建议熟悉存储池和卷标识符背后的概念，但在实际使用中，并不要求在命令行执行这些低级操作。
通常，卷的分配和移除由虚拟机与容器管理工具完成。

不过，系统提供了名为 pvesm（“Proxmox VE Storage Manager”）的命令行工具， 可执行常见的存

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
```

更改或设置存储选项

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

移除存储池。这不会删除任何数据，也不会断开连接或卸载任何内容。它只会移除存储配置。

```
pvesm remove <STORAGE_ID>
```

分配卷

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

在本地存储中分配一个 4G 卷。如果将空字符串作为 <name> 传入，名称会自动生成。

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

释放卷

```
pvesm free <VOLUME_ID>
```

Warning
这会真正销毁该卷上的所有数据。

列出存储状态

```
pvesm status
```

列出存储内容

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

列出按 VMID 分配的卷

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

列出 ISO 镜像

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

列出容器模板

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

显示卷的文件系统路径

```
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 import 配合使用。流格式 qcow2+size 不同于 qcow2 格式。 因此，导出的文件不能简单地附加到虚拟机。其

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

常用命令示例：

```
pvesm status
pvesm list local
pvesm path <VOLUME_ID>
```

7.5 目录后端

存储池类型：dir

Proxmox VE 可以使用本地目录或本地挂载的共享作为存储。目录是一种文件级存储，因此可以存储虚
镜像或备份文件等任何内容类型。

Note
可以通过标准 Linux /etc/fstab 挂载其他存储，然后为该挂载点定义目录存储。通过这种方式，可以使用 Linux 支持的任何文件系统。

该后端仅假定底层目录兼容 POSIX，不再要求其他特性。这意味着不能在存储层 创建快照。但对于使用
qcow2 文件格式的虚拟机镜像存在一种替代方式， 因为该格式内部支持快照。

Tip
某些存储类型不支持 O_DIRECT，因此不能在这类存储上使用 none 缓存模式。可改用 writeback 缓存模式。

系统使用预定义的目录布局，将不同内容类型存储到不同子目录中。所有文件级 存储后端都会使用该布

Table 7.2: 目录布局

内容类型	子目录
虚拟机镜像	images/<VMID>/
ISO 镜像	template/iso/
容器模板	template/cache/
备份文件	dump/
片段	snippets/

7.5.1 配置

该后端支持所有通用存储属性，并额外增加两个属性。path 属性用于指定目录。 该目录必须是绝对文
可选的 content-dirs 属性允许更改默认布局。它由逗号分隔的标识符列表组成， 格式如下：
vtype=path

其中 vtype 是该存储允许的内容类型之一，path 是相对于该存储挂载点的路径。

配置示例（**/etc/pve/storage.cfg**）

```
dir: backup
        path /mnt/backup
        content backup
        prune-backups keep-last=7
        max-protected-backups 3
        content-dirs backup=custom/backup/dir
```

上述配置定义了一个名为 backup 的存储池。该存储池可为每台虚拟机最多保存 7 个普通备份（keep-
3 个受保护备份。备份文件的实际路径为 /mnt/backup/custom/backup/dir/...。

7.5.2 文件命名约定

该后端对虚拟机镜像使用明确的命名方案：

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

<VMID>
指定所属虚拟机。

<NAME>
可以是不包含空白字符的任意名称（ascii）。后端默认使用 disk-[N]，其中 [N] 会替换为整

<FORMAT>
指定镜像格式（raw|qcow2|vmdk）。

创建虚拟机模板时，所有虚拟机镜像都会被重命名，以表示它们现在是只读的，并可作为克隆的基础镜

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

> **Note**
> 这类基础镜像用于生成克隆镜像。因此，确保这些文件只读且永不被修改
> 非常重要。后端会将访问模式更改为 0444，并在存储支持时设置 immutable 标志（chattr
> +i）。

7.5.3 存储特性

如上所述，大多数文件系统本身不直接支持快照。为规避该问题，此后端可以使用 qcow2 的内部快照能

克隆也是同样的情况。后端使用 qcow2 基础镜像功能来创建克隆。

Table 7.3: 后端 dir 的存储特性

内容类型	镜像格式	共享	快照	克隆
images rootdir vztmpl iso backup snippets	raw qcow2 vmdk subvol	no	qcow2	qcow2

7.5.4 示例

请使用以下命令在存储 local 上分配一个 4GB 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'
```

Note
镜像名称必须符合上述命名约定。

可以通过以下命令显示真实文件系统路径：

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

可以使用以下命令删除该镜像：

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

常用命令示例：

```
pvesm status
pvesm list local
```

7.6 NFS 后端

存储池类型：nfs

NFS 后端基于目录后端，因此共享其中的大多数属性。目录布局和文件命名约定也相同。 它的主要优势
NFS 服务器属性，使后端能够自动挂载共享，无需修改 /etc/fstab。该后端还可以检测服务器是否在

7.6.1 配置

该后端支持除 shared 标志之外的所有通用存储属性；shared 标志始终为启用状态。 此外，以下属性用
NFS 服务器：

server
服务器 IP 或 DNS 名称。为避免 DNS 查询延迟，通常更建议使用 IP 地址而不是 DNS 名称，除非
DNS 服务器，或者已将该服务器写入本地 `/etc/hosts` 文件。

export
NFS 导出路径（即 `pvesm nfsscan` 列出的路径）。

也可以设置 NFS 挂载选项：

path
本地挂载点（默认为 `/mnt/pve/<STORAGE_ID>/`）。

content-dirs
对默认目录布局的覆盖配置。可选。

options
NFS 挂载选项（参见 `man nfs`）。

配置示例（**/etc/pve/storage.cfg**）

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

Tip
NFS 请求超时后，默认会无限重试。这可能导致客户端出现意外的长时间挂起。
对于只读内容，值得考虑使用 NFS `soft` 选项，它会将重试次数限制为三次。

7.6.2 eNFS 多路径

eNFS 是 openEuler 对 NFS 客户端的增强，用于在一个 NFS 挂载点上建立多条访问路径， 提供多路径
属于客户端增强，不要求 NFS 服务器部署 PXVIRT 组件；但服务器端需要启用 NFSv3，因为 eNFS
当前仅支持 NFSv3。

更多信息可参见 openEuler eNFS 使用指南： eNFS 使用指南。

典型架构

在典型部署中，PXVIRT 主机和 NFS 服务器各自具有多个可互通的业务 IP。例如：

- PXVIRT 主机：10.13.13.81、10.13.14.81
- NFS 服务器：10.13.13.3、10.13.14.3

启用 eNFS 后，同一个 NFS 存储可以同时通过多条路径访问 NFS 服务器。eNFS 会在 可用路径之间进行
IO 切换到其他可用路径。

```
+---------------------+                    +---------------------+
|     PXVIRT b''主b''b''机b''      |                    |      NFS b''服b''b ↩
   ''务b''b''器b''       |
|                      |                    |                      |
|  +-------------+     |                    |  +-------------+     |
|  | eNFS b''客b''b''户b''b''端b''  |     |                    |  | NFS  ↩
   Server  |     |
|  +-------------+     |                    |  +-------------+     |
|          |           |                    |          |           |
|    10.13.13.81  ---+----------------------> 10.13.13.3         |
|                      |      b''连b''b''接b''b''路b''b''径b'' 1     |  ↩
                       |
|    10.13.14.81  ---+----------------------> 10.13.14.3         |
|                      |      b''连b''b''接b''b''路b''b''径b'' 2     |  ↩
                       |
+---------------------+                    +---------------------+
```

系统要求

使用 eNFS 需要满足以下条件：

- PXVIRT 主机使用 openEuler 内核 6.6-openeuler，且版本不低于 6.6.0-10。
- NFS 服务器启用 NFSv3 支持。
- PXVIRT 主机到 remoteaddrs 中配置的所有 NFS 服务器地址均可达。

eNFS 配置文件

可以在每个 PXVIRT 节点上创建 /etc/enfs/config.ini 配置文件。以下示例启用多路径：

文件 **/etc/enfs/config.ini**

```
multipath_disable=0
```

该配置是可选项。未创建该文件时，eNFS 使用内核模块的默认配置。

存储配置

创建 NFS 存储时，需要使用 NFSv3，并在 options 中配置 remoteaddrs。例如：

eNFS 存储配置示例（/etc/pve/storage.cfg）

```
nfs: nfs
        export /NFS/vm
        path /mnt/pve/nfs
        server 10.13.14.3
        content images
        options remoteaddrs=10.13.14.3~10.13.13.3,vers=3
        prune-backups keep-all=1
```

也可以先在 Web 界面中创建普通 NFS 存储，并选择 NFS 版本 3。随后在 /etc/pve/storage.cfg 中为该存储添加 options 行。

remoteaddrs 中的 IP 地址必须属于同一台 NFS 服务器或同一个 NFS 服务集群。 多个地址使用 ~ 分隔。eNFS 会根据这些远端地址和本机可达地址建立多条访问路径。

验证

可以使用 mount 查看当前挂载选项。启用 eNFS 后，输出中应包含 remoteaddrs 和 enfs_info：

```
# mount
10.13.14.3:/NFS/vm on /mnt/pve/nfs type nfs
(rw,relatime,vers=3,rsize=1048576,wsize=1048576,namlen=255,hard,
proto=tcp,timeo=600,retrans=2,sec=sys,mountaddr=10.13.14.3,
mountvers=3,mountproto=udp,local_lock=none,addr=10.13.14.3,
remoteaddrs=10.13.14.3~10.13.13.3,enfs_info=10.13.14.3_1)
```

也可以通过 /proc/enfs/<nfsserver>/stat 查看 eNFS 路径和 IO 统计信息：

```
# cat /proc/enfs/*/stat
id    local_addr     remote_addr    r_count  r_rtt  r_exec  w_count  w_rtt  ↩
    w_exec  queuelen
0     10.13.14.81    10.13.14.3     0        0      0       499      7  ↩
        24      0
1     10.13.13.81    10.13.13.3     0        0      0       525      6  ↩
        26      0
```

7.6.3 存储特性

NFS 不支持快照，但该后端会使用 qcow2 特性实现快照和克隆。

Table 7.4: 后端 nfs 的存储特性

内容类型	镜像格式	共享	快照	克隆
images rootdir vztmpl iso backup snippets	raw qcow2 vmdk	yes	qcow2	qcow2

7.6.4 示例

可以使用以下命令获取导出的 NFS 共享列表：

```
# pvesm scan nfs <server>
```

常用命令示例：

```
# pvesm status
# pvesm scan nfs <server>
```

7.7 CIFS 后端

存储池类型：cifs

CIFS 后端扩展了目录后端，因此无需手动设置 CIFS 挂载。可以直接通过 Proxmox VE API 或 Web UI 添加此类存储，并继续使用后端提供的优势，例如服务器心跳检查，以及便捷地选择 导出的共享

7.7.1 配置

该后端支持所有通用存储属性，但 shared 标志除外，该标志始终会被设置。此外，还可使用 以下 CIFS 专用属性：

server
: 服务器 IP 或 DNS 名称。必填。

> **Tip**
> 为避免 DNS 查询延迟，通常最好使用 IP 地址而不是 DNS 名称，除非你拥有非常可靠的 DNS 服务器，或已在本地 /etc/hosts 文件中列出该服务器。

share
: 要使用的 CIFS 共享（可通过 pvesm scan cifs <address> 或 Web UI 获取可用共享）。必填。

username
CIFS 存储使用的用户名。可选，默认为 ‘guest’。

password
用户密码。可选。它会保存到仅 root 可读的文件中 (/etc/pve/priv/storage/<STORAGE-ID

domain
设置该存储的用户域（工作组）。可选。

smbversion
SMB 协议版本。可选，默认为 3。由于安全问题，不支持 SMB1。

path
本地挂载点。可选，默认为 /mnt/pve/<STORAGE_ID>/。

content-dirs
默认目录布局的覆盖设置。可选。

options
额外的 CIFS 挂载选项（参见 man mount.cifs）。部分选项会自动设置，不应在此处设置。Proxmox VE 始终会设置 soft 选项。根据配置，以下选项会自动设置：username, credentia guest, domain, vers。

subdir
要挂载的共享子目录。可选，默认为共享的根目录。

配置示例 **(/etc/pve/storage.cfg)**

```
cifs: backup
        path /mnt/pve/backup
        server 10.0.0.11
        share VMData
        content backup
        options noserverino,echo_interval=30
        username anna
        smbversion 3
        subdir /data
```

7.7.2 存储特性

CIFS 不支持存储级快照。但如果仍希望使用快照和克隆功能，可以使用 qcow2 后端文件。

Table 7.5: 后端 cifs 的存储特性

Content types	Image formats	Shared	Snapshots	Clones
images rootdir vztmpl iso backup snippets	raw qcow2 vmdk	yes	qcow2	qcow2

7.7.3 示例

可以使用以下命令获取导出的 CIFS 共享列表：

```
# pvesm scan cifs <server> [--username <username>] [--password]
```

然后可以使用以下命令将其中一个共享作为存储添加到整个 Proxmox VE 集群：

```
# pvesm add cifs <storagename> --server <server> --share <share> [-- ↩
   username <username>] [--password]
```

7.8 Proxmox Backup Server

存储池类型：pbs

该后端允许像集成其他存储一样，将 Proxmox Backup Server 直接集成到 Proxmox VE 中。可以通过 Proxmox VE API、CLI 或 Web 界面直接添加 Proxmox Backup 存储。

7.8.1 配置

该后端支持所有通用存储属性，但 shared 标志除外，该标志始终会被设置。此外， Proxmox Backup Server 还支持以下专用属性：

server
: 服务器 IP 或 DNS 名称。必填。

port
: 使用该端口替代默认端口 8007。可选。

username
: Proxmox Backup Server 存储使用的用户名。必填。

Tip
不要忘记在用户名中添加 realm。例如 root@pam 或 archiver@pbs。

---

password
用户密码。该值会保存到 /etc/pve/priv/storage/<STORAGE-ID>.pw 文件中，并限制为 root 用户访问。必填。

datastore
要使用的 Proxmox Backup Server datastore ID。必填。

fingerprint
Proxmox Backup Server API TLS 证书的指纹。可以在服务器 Dashboard 中获取，也可以使用 proxmox-backup-manager cert info 命令获取。对于自签名证书，或主机不信任服务器 CA 的其他证书，该项为必填。

encryption-key
用于在客户端侧加密备份数据的密钥。目前仅支持未受密码保护的密钥（无 key derive function，kdf）。密钥会保存到 /etc/pve/priv/storage/<STORAGE-ID>.enc 文件中，并限制为 root 用户访问。使用特殊值 autogen 可通过 proxmox-backup-client key create --kdf none <path> 自动生成新密钥。可选。

master-pubkey
用于在备份任务中加密备份加密密钥的 RSA 公钥。它会保存到 /etc/pve/priv/storage/<ST 文件中，并限制为 root 用户访问。 备份加密密钥的加密副本会附加到每个备份中，并存储在 Proxmox Backup Server 实例上，以供恢复使用。可选，但要求配置 encryption-key。

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:3 ↩
           c:b2
        prune-backups keep-all=1
        username archiver@pbs
        encryption-key a9:ee:c8:02:13:..snip..:2d:53:2c:98
        master-pubkey 1
```

7.8.2 存储特性

Proxmox Backup Server 仅支持备份，备份可以基于块级或文件级。Proxmox VE 对虚拟机使用块级对容器使用文件级备份。

Table 7.6: 后端 pbs 的存储特性

内容类型	镜像格式	共享	快照	克隆
backup	n/a	是	n/a	n/a

7.8.3 加密

可以选择配置使用 AES-256 GCM 模式的客户端侧加密。可以通过 Web 界面配置加密，也可以在 CLI 中使用 encryption-key 选项（见上文）。密钥会保存到 /etc/pve/priv/storage/<STORAG 文件中，该文件仅 root 用户可访问。

Warning
如果没有对应密钥，备份将无法访问。因此，应有序保管密钥，并将其存放在与被备份内容分
例如，如果使用某个系统上的密钥备份整个系统，而该系统随后因任何原因无法访问并需要还
由于加密密钥会随故障系统一同丢失，还原将无法完成。

建议将密钥安全保存，同时确保在灾难恢复时能够快速取得。因此，最佳存放位置是密码管理器，这样可以立即恢复。作为补充，也应将密钥保存到 USB 闪存盘，并存放在安全位置。这样密钥就与任
还应考虑将密钥的纸质副本锁存在安全位置。paperkey 子命令可用于创建密钥的 QR 编码版本。
以下命令会将 paperkey 命令的输出发送到文本文件，以便打印。

```
# 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：将所有执行加密备份的客户端配置为使用同一个主
之后所有加密备份都会包含一份经 RSA 加密的已用 AES 加密密钥副本。即使客户端系统不再可用，对应的主私钥也允许恢复 AES 密钥并解密备份。

Warning
主密钥对适用与普通加密密钥相同的安全保管规则。没有私钥副本就无法恢复！
paperkey 命令支持生成主私钥的纸质副本，以便存放在安全的物理位置。

由于加密是在客户端侧管理的，同一服务器 datastore 可同时用于未加密备份和加密备份，即使这些备
因此通常最好创建单独的 datastore。

Note
如果加密没有实际收益，请不要使用加密。例如，当服务器在受信任网络中本地运行时，
从未加密备份中恢复总是更简单。

7.8.4 示例：通过 CLI 添加存储

可以使用以下命令获取可用 Proxmox Backup Server datastore 列表：

```
# pvesm scan pbs <server> <username> [--password <string>] [--fingerprint < ↩
   string>]
```

然后可以使用以下命令将其中一个 datastore 作为存储添加到整个 Proxmox VE 集群：

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

7.9 GlusterFS 后端

存储池类型：glusterfs

GlusterFS 是一种可扩展的网络文件系统。该系统采用模块化设计，可运行在通用硬件上，并能以较低
PB 规模，并可处理数千个客户端。

Note
在节点或 brick 崩溃后，GlusterFS 会执行完整的 rsync 以确保数据一致。
对于大文件，这可能耗费很长时间，因此该后端不适合存储大型虚拟机镜像。

7.9.1 配置

该后端支持所有通用存储属性，并额外增加以下 GlusterFS 专用选项：

server
GlusterFS volfile 服务器 IP 或 DNS 名称。

server2
备用 volfile 服务器 IP 或 DNS 名称。

volume
GlusterFS 卷。

transport
GlusterFS 传输方式：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 文件命名约定

目录布局和文件命名约定继承自 dir 后端。

7.9.3 存储特性

该存储提供文件级接口，但没有原生快照或克隆实现。

Table 7.7: 后端 glusterfs 的存储特性

内容类型	镜像格式	共享	快照	克隆
images vztmpl iso backup snippets	raw qcow2 vmdk	yes	qcow2	qcow2

7.9.4 示例

可以使用以下命令获取可用 GlusterFS 卷列表：

```
# pvesm scan glusterfs <server>
```

常用命令示例：

```
pvesm scan glusterfs <server>
pvesm status
```

7.10 本地 ZFS Pool 后端

存储池类型：zfspool

此后端允许访问本地 ZFS pool (或这些 pool 内的 ZFS 文件系统) 。

7.10.1 配置

该后端支持通用存储属性 content、nodes、disable , 以及以下 ZFS 专用属性：

pool
: 选择 ZFS pool/文件系统。所有分配都在该 pool 内完成。

blocksize
: 设置 ZFS blocksize 参数。

sparse
: 使用 ZFS thin-provisioning。sparse 卷是指 reservation 不等于卷大小的卷。

mountpoint
: ZFS pool/文件系统的挂载点。更改此项不会影响 zfs 看到的数据集 mountpoint 属性。默认为 /<pool>。

配置示例（**/etc/pve/storage.cfg**）

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

7.10.2 文件命名约定

该后端对虚拟机镜像使用以下命名方案：

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

<VMID>
: 指定所属虚拟机。

<NAME>
: 可以是不包含空白字符的任意名称（ascii）。后端默认使用 disk[N]，其中 [N] 会替换为整数

7.10.3 存储特性

在快照和克隆方面，ZFS 可能是最先进的存储类型。该后端对虚拟机镜像（格式 raw）和容器数据（格 subvol）均使用 ZFS dataset。ZFS 属性会从父 dataset 继承，因此可以直接在父 dataset 上设置默认值。

Table 7.8: 后端 zfs 的存储特性

内容类型	镜像格式	共享	快照	克隆
images rootdir	raw subvol	否	是	是

7.10.4 示例

建议创建一个额外的 ZFS 文件系统来存放虚拟机镜像：

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

要在新分配的文件系统上启用压缩：

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

可以使用以下命令获取可用 ZFS 文件系统列表：

```
# pvesm scan zfs
```

7.10.5 常用命令示例

查看当前 Proxmox VE 存储状态：

```
pvesm status
```

列出节点上可供存储后端使用的 ZFS 文件系统：

```
pvesm scan zfs
```

7.11 LVM 后端

存储池类型：`lvm`

LVM 是构建在硬盘和分区之上的轻量级软件层。它可用于将可用磁盘空间划分为较小的逻辑卷。LVM 在 Linux 上被广泛使用，可简化硬盘管理。

另一种使用场景是在大型 iSCSI LUN 之上部署 LVM。这样可以方便地管理该 iSCSI LUN 上的空间；否则 iSCSI 规范未定义用于空间分配的管理接口，这类管理将无法实现。

7.11.1 配置

LVM 后端支持通用存储属性 `content`、`nodes`、`disable`，以及以下 LVM 专用属性：

vgname
: LVM 卷组名称。必须指向一个已存在的卷组。

base
: 基础卷。访问该存储前，此卷会被自动激活。当 LVM 卷组位于远程 iSCSI 服务器上时，此项尤其有

saferemove
: 在 Web UI 中称为 "Wipe Removed Volumes"。移除 LV 时将数据清零。移除卷时，此项可确保 extent 的其他 LV 无法访问这些数据。该操作成本较高，但在某些环境中可能是必要的安全措施。

saferemove_throughput
: 擦除吞吐量（`cstream -t` 参数值）。

配置示例（**/etc/pve/storage.cfg**）

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

7.11.2 文件命名约定

该后端基本使用与 ZFS pool 后端相同的命名约定。

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

7.11.3 存储特性

LVM 是典型的块存储，但此后端不支持快照和克隆。遗憾的是，普通 LVM 快照效率较低，因为在快照期
它的一大优势是可以部署在共享存储之上，例如 iSCSI LUN。该后端自身实现了适当的集群范围锁定。

Tip
较新的 LVM-thin 后端支持快照和克隆，但不支持共享存储。

Table 7.9: 后端 `lvm` 的存储特性

内容类型	镜像格式	共享	快照	克隆
`images rootdir`	`raw`	可行	否	否

7.11.4 示例

可以使用以下命令获取可用 LVM 卷组列表：

```
# pvesm scan lvm
```

7.11.5 常用命令示例

查看当前 Proxmox VE 存储配置：

```
pvesm status
```

扫描节点上可用的 LVM 卷组：

```
pvesm scan lvm
```

7.12 LVM thin 后端

存储池类型：`lvmthin`

普通 LVM 通常会在创建卷时分配块。LVM thin pool 则在写入数据时才分配块。这种行为称为 thin-provisioning，因为卷的逻辑大小可以大于实际可用的物理空间。

可以使用常规 LVM 命令行工具来管理和创建 LVM thin pool（详情请参见 `man lvmthin`）。假设已经
pve 的 LVM 卷组，以下命令会创建一个名为 `data`、 大小为 100G 的 LVM thin pool：

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

7.12.1 配置

LVM thin 后端支持通用存储属性 content、nodes、disable，以及以下 LVM 专用属性：

vgname
: LVM 卷组名称。必须指向一个已存在的卷组。

thinpool
: LVM thin pool 的名称。

配置示例（**/etc/pve/storage.cfg**）

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

7.12.2 文件命名约定

该后端基本使用与 ZFS pool 后端相同的命名约定。

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

7.12.3 存储特性

LVM thin 是块存储，但可以高效地完整支持快照和克隆。新卷会自动初始化为零。

需要注意，LVM thin pool 不能跨多个节点共享，因此只能作为本地存储使用。

Table 7.10: 后端 lvmthin 的存储特性

内容类型	镜像格式	共享	快照	克隆
images rootdir	raw	否	是	是

7.12.4 示例

可以使用以下命令获取卷组 pve 上可用的 LVM thin pool 列表：

```
# pvesm scan lvmthin pve
```

7.12.5 常用命令示例

查看当前存储状态：

```
# pvesm status
```

列出节点上的 LVM 卷组：

```
# pvesm scan lvm
```

7.13 Open-iSCSI initiator

存储池类型：iscsi

iSCSI 是一种广泛用于连接存储服务器的技术。几乎所有存储厂商都支持 iSCSI。也有 可用的开源 iSCSI target 解决方案，例如基于 Debian 的 OpenMediaVault。

要使用该后端，需要安装 Open-iSCSI（open-iscsi）软件包。这是标准 OpenEuler 软件包，但为了

```
# dnf install open-iscsi
```

低级 iscsi 管理任务可以使用 iscsiadm 工具完成。

7.13.1 配置

该后端支持通用存储属性 content、nodes、disable，以及以下 iSCSI 专用属性：

portal
: iSCSI portal（IP 或 DNS 名称，可带端口）。

target
: 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
```

> **Tip**
> 如果希望在 iSCSI 之上使用 LVM，将 content none 设为该值是合理的。这样就不能直接使用 iSCSI LUN 创建虚拟机。

7.13.2 文件命名约定

iSCSI 协议没有定义用于分配或删除数据的接口。这些操作需要在 target 侧完成，并且 与厂商实现相关只是将它们作为带编号的 LUN 导出。因此，Proxmox VE iSCSI 卷名 只是编码了 Linux 内核所看到的 LUN 的一些信息。

7.13.3 存储特性

iSCSI 是块级存储类型，不提供管理接口。因此，通常最好导出一个较大的 LUN，并在该 LUN 之上设置 LVM。随后可以使用 LVM 插件管理该 iSCSI LUN 上的存储。

Table 7.11: 后端 `iscsi` 的存储特性

内容类型	镜像格式	共享	快照	克隆
`images none`	`raw`	yes	no	no

7.13.4 常用命令示例

可以扫描远程 iSCSI portal，并使用以下命令获取可能的 target 列表：

```
pvesm scan iscsi <HOST[:PORT]>
```

7.14 用户态 **iSCSI** 后端

存储池类型：`iscsidirect`

该后端提供的功能基本与 Open-iSCSI 后端相同，但通过用户态库实现。要使用该后端，需要安装 `libiscsi` 软件包。

需要注意的是，该后端不涉及内核驱动，因此可视为一种性能优化方式。但它也有一个缺点： 不能在这 iSCSI LUN 之上使用 LVM。因此，所有空间分配都需要在存储服务器端管理。

7.14.1 配置

用户态 iSCSI 后端使用与 Open-iSCSI 后端相同的配置选项。

配置示例（**/etc/pve/storage.cfg**）

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

7.14.2 存储特性

Note
该后端仅适用于虚拟机。容器不能使用此驱动。

Table 7.12: 后端 iscsidirect 的存储特性

内容类型	镜像格式	共享	快照	克隆
images	raw	是	否	否

7.14.3 常用命令示例

查看当前存储状态：

```
# pvesm status
```

扫描可用 iSCSI target：

```
# pvesm scan iscsi <portal>
```

7.15 Ceph RADOS 块设备（RBD）

存储池类型：rbd

Ceph 是一种分布式对象存储和文件系统，设计目标是提供优秀的性能、可靠性和可扩展性。 RADOS 块设备实现了功能丰富的块级存储，并具备以下优势：

- thin provisioning
- 可调整大小的卷
- 分布式且具备冗余能力（跨多个 OSD 条带化）
- 完整的快照和克隆能力
- 自愈能力
- 无单点故障
- 可扩展到 EB 级别
- 同时提供内核态和用户态实现

Note
对于较小规模的部署，也可以直接在 Proxmox VE 节点上运行 Ceph 服务。现代硬件通常具备充足的 CPU 算力和内存，因此在同一节点上同时运行存储服务和虚拟机是可

7.15.1 配置

该后端支持通用存储属性 nodes、disable、content，以及以下 rbd 专用属性：

monhost
monitor 守护进程 IP 列表。可选，仅当 Ceph 未运行在 Proxmox VE 集群上时需要。

pool
Ceph pool 名称。

username
RBD 用户 ID。可选，仅当 Ceph 未运行在 Proxmox VE 集群上时需要。注意这里只应使用用户 ID，必须省略 client. 类型前缀。

krbd
强制通过 krbd 内核模块访问 RADOS 块设备。可选。

Note
无论该选项取值如何，容器都会使用 krbd。

外部 **Ceph** 集群的配置示例（**/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
```

Tip
可以使用 rbd 工具执行低层管理任务。

7.15.2 认证

Note
如果 Ceph 安装在本地 Proxmox VE 集群上，添加存储时会自动完成以下操作。

如果使用默认启用的 cephx 认证，需要提供外部 Ceph 集群的 keyring。

要通过 CLI 配置该存储，首先需要让包含 keyring 的文件可用。一种方法是将该文件从外部 Ceph 集群直接复制到某个 Proxmox VE 节点。以下示例会将它复制到当前执行命令节点的 /root 目录：

```
# scp <external cephserver>:/etc/ceph/ceph.client.admin.keyring /root/rbd. ↩
   keyring
```

然后使用 pvesm CLI 工具配置外部 RBD 存储，并通过 --keyring 参数指定刚复制的 keyring 文件路径。例如：

```
# pvesm add rbd <name> --monhost "10.1.1.20 10.1.1.21 10.1.1.22" --content ↩
   images --keyring /root/rbd.keyring
```

通过 GUI 配置外部 RBD 存储时，可以将 keyring 复制并粘贴到相应字段中。

keyring 将存储在：

```
# /etc/pve/priv/ceph/<STORAGE_ID>.keyring
```

Tip
连接外部集群时，建议创建仅具备所需权限的 keyring。有关 Ceph 用户管理的更多信息，请参见 Ceph 文档。[a]

[a] Ceph User Management

7.15.3 Ceph 客户端配置（可选）

连接外部 Ceph 存储时，并不总是允许在外部集群的配置数据库中设置客户端专用选项。可以在 Ceph keyring 旁边添加一个 ceph.conf，用于调整该存储的 Ceph 客户端配置。

ceph.conf 需要与该存储使用相同名称。

```
# /etc/pve/priv/ceph/<STORAGE_ID>.conf
```

可用设置请参见 RBD 配置参考 [1]。

Note
不要轻易修改这些设置。Proxmox VE 会将 <STORAGE_ID>.conf 与存储配置合并使用。

7.15.4 存储特性

rbd 后端是块级存储，并实现了完整的快照和克隆功能。

Table 7.13: 后端 rbd 的存储特性

内容类型	镜像格式	共享	快照	克隆
images rootdir	raw	是	是	是

[1] RBD configuration reference https://docs.ceph.com/en/quincy/rbd/rbd-config-ref/

7.15.5 常用命令示例

查看 Proxmox VE 存储状态：

```
# pvesm status
```

使用 rbd 工具列出指定 Ceph pool 中的镜像：

```
# rbd ls <pool>
```

7.16 Ceph 文件系统（CephFS）

存储池类型：cephfs

CephFS 实现了兼容 POSIX 的文件系统，并使用 Ceph 存储集群保存 数据。由于 CephFS 构建在 Ceph 之上，它继承了 Ceph 的大多数属性，包括冗余、 可扩展性、自愈能力和高可用性。

> **Tip**
> Proxmox VE 可以 管理 Ceph 部署，这使 CephFS 存储配置更为简单。现代硬件提供了大量处理能力和内存，因此在同一节点上运行存储服务和虚拟机是可行的，且不会产生显著性能影响。

7.16.1 配置

该后端支持通用存储属性 nodes、disable、content，以及以下 cephfs 专用属性：

fs-name
: Ceph FS 的名称。

monhost
: monitor 守护进程地址列表。可选，仅在 Ceph 未运行于 Proxmox VE 集群上时需要。

path
: 本地挂载点。可选，默认为 /mnt/pve/<STORAGE_ID>/。

username
: Ceph 用户 ID。可选，仅在 Ceph 未运行于 Proxmox VE 集群上时需要；在本地集群场景中默认为 admin。

subdir
: 要挂载的 CephFS 子目录。可选，默认为 /。

fuse
: 通过 FUSE 而不是内核客户端访问 CephFS。可选，默认为 0。

外部 **Ceph** 集群的配置示例（**/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
        fs-name cephfs
```

Note
如果未禁用 cephx，请不要忘记设置客户端的密钥文件。

7.16.2 认证

Note
如果 Ceph 本地安装在 Proxmox VE 集群上，添加存储时以下操作会自动完成。

如果使用默认启用的 cephx 认证，则需要提供来自外部 Ceph 集群的 secret。

要通过 CLI 配置存储，首先需要提供包含 secret 的文件。一种方式是直接从外部 Ceph 集群将该文件复
Proxmox VE 节点。以下示例会将其复制到运行命令节点的 /root 目录：

```
# scp <external cephserver>:/etc/ceph/cephfs.secret /root/cephfs.secret
```

然后使用 pvesm CLI 工具配置外部 RBD 存储，并使用 --keyring 参数；该参数 需要指向已复制的
secret 文件路径。例如：

```
# pvesm add cephfs <name> --monhost "10.1.1.20 10.1.1.21 10.1.1.22" -- ↩
   content backup --keyring /root/cephfs.secret
```

通过 GUI 配置外部 RBD 存储时，可以将 secret 复制并粘贴到相应字段中。

该 secret 仅包含密钥本身，不同于 rbd 后端；后者还包含 [client.userid] 段。

secret 将存储在：

```
# /etc/pve/priv/ceph/<STORAGE_ID>.secret
```

可以使用以下命令以 Ceph 管理员身份从 Ceph 集群获取 secret，其中 userid 是已配置 为可访问该集
ID。有关 Ceph 用户管理的更多信息，请参见 Ceph 文档。[a]

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

7.16.3 存储特性

cephfs 后端是在 Ceph 集群之上提供的兼容 POSIX 的文件系统。

Table 7.14: 后端 cephfs 的存储特性

内容类型	镜像格式	共享	快照	克隆
vztmpl iso backup snippets	none	yes	yes[1]	no

[1] 虽然目前没有已知缺陷，但由于缺少充分测试，快照尚不能保证稳定。

7.16.4 常用命令示例

以下命令可用于查看存储状态，并通过 CLI 添加外部 CephFS 存储：

```
# pvesm status
# pvesm add cephfs <name> --monhost ”10.1.1.20 10.1.1.21 10.1.1.22” -- ↩
   content backup --keyring /root/cephfs.secret
```

7.17 BTRFS 后端

存储池类型：btrfs

从表面上看，这种存储类型与目录存储类型非常相似，因此可参见目录后端章节 了解总体概览。

主要区别在于，使用这种存储类型时，raw 格式的磁盘会放置在子卷中，以便 创建快照，并支持在保留

Note
BTRFS 在打开文件时会遵循 O_DIRECT 标志，这意味着虚拟机不应使用 none 缓存模式，否则会出现校验和错误。

7.17.1 配置

该后端的配置方式与目录存储类似。注意，将某个目录添加为 BTRFS 存储时， 如果该目录本身并不是挂
is_mountpoint 选项指定实际挂载点。

例如，如果 BTRFS 文件系统挂载在 /mnt/data2，并且要将其 pve-storage/ 子目录（建议可以是一
data2 的存储池，可以使用以下条目：

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

7.17.2 快照

为子卷或 raw 文件创建快照时，系统会创建一个只读子卷作为快照，其路径为 原路径后接 @ 和快照名称

常用命令示例：

```
pvesm add btrfs data2 --path /mnt/data2/pve-storage
pvesm status
```

7.18 ZFS over iSCSI 后端

存储池类型：zfs

该后端通过 ssh 访问一台将 ZFS 池作为存储并实现 iSCSI target 的远程机器。 对于每个客户机磁盘，ZVOL，并将其导出为 iSCSI LUN。 Proxmox VE 使用该 LUN 作为客户机磁盘。

支持以下 iSCSI target 实现：

- LIO (Linux)
- IET (Linux)
- ISTGT (FreeBSD)
- Comstar (Solaris)

Note
该插件需要具备 ZFS 能力的远程存储设备，不能用它在普通 Storage Appliance/SAN 上创建 ZFS 池。

7.18.1 配置

要使用 ZFS over iSCSI 插件，需要配置远程机器（target），使其接受来自 Proxmox VE 节点的 ssh 连接。Proxmox VE 会连接到 target，以创建 ZVOL 并通过 iSCSI 导出它们。 认证通过存储在 /etc/pve/priv/zfs/<target_ip>_id_rsa 的 ssh-key （无密码保护）完成。

以下步骤会创建 ssh-key，并将其分发到 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
```

该后端支持通用存储属性 content、nodes、disable，并支持以下 ZFS over iSCSI 专用属性：

pool
: iSCSI target 上的 ZFS 池或文件系统。所有分配都在该池内完成。

portal
: iSCSI portal（IP 或 DNS 名称，可带端口）。

---

target
: iSCSI target。

iscsiprovider
: 远程机器使用的 iSCSI target 实现

comstar_tg
: comstar views 的 target group。

comstar_hg
: comstar views 的 host group。

lio_tpg
: Linux LIO targets 的 target portal group

nowritecache
: 在 target 上禁用写缓存

blocksize
: 设置 ZFS blocksize 参数。

sparse
: 使用 ZFS 精简配置。sparse 卷是 reservation 不等于卷大小的卷。

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
```

7.18.2 存储特性

ZFS over iSCSI 插件提供支持快照的共享存储。需要确保 ZFS 设备不会成为部署中的单点故障。

Table 7.15: 后端 `iscsi` 的存储特性

内容类型	镜像格式	共享	快照	克隆
images	raw	yes	yes	no

常用命令示例：

```
pvesm status
pvesm list <STORAGE_ID>
pvesm path <VOLUME_ID>
```

Chapter 8

部署超融合 Ceph 集群

8.1 简介

Proxmox VE 将计算系统和存储系统统一起来，也就是说，你可以在同一个集群中的相同物理 节点上同和存储资源可以整合为单一的超融合设备。独立的存储网络（SAN）以及通过网络附加 存储（NAS）建 Ceph 这一开源软件定义存储平台， Proxmox VE 能够直接在 hypervisor 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。

在 Proxmox VE 上使用 Ceph 的一些优势包括：

- 可通过 CLI 和 GUI 轻松设置和管理
- 精简置备
- 支持快照
- 自愈
- 可扩展到 EB 级别
- 提供块存储、文件系统和对象存储
- 可设置具备不同性能和冗余特征的池
- 数据会进行复制，从而具备容错能力
- 可运行在通用硬件上
- 不需要硬件 RAID 控制器
- 开源

对于中小规模部署，可以直接在 Proxmox VE 集群节点上安装 Ceph 服务器，以使用 RADOS Block Devices（RBD）或 CephFS（参见 Ceph RADOS Block Devices (RBD)）。现代硬件拥有大量 CPU 计算能力和内存，因此可以在同一 节点上同时运行存储服务和虚拟客户机。

为简化管理，Proxmox VE 提供原生集成，可通过内置 Web 界面或 *pveceph* 命令行工具在 Proxmox VE 节点上安装和管理 Ceph 服务。

8.2 术语

作为 RBD 存储使用时，Ceph 由多个守护进程组成：

- Ceph Monitor (ceph-mon，或 MON)
- Ceph Manager (ceph-mgr，或 MGR)
- Ceph Metadata Service (ceph-mds，或 MDS)
- Ceph Object Storage Daemon (ceph-osd，或 OSD)

Tip
强烈建议熟悉 Ceph [a]，其架构 [b] 以及术语 [c].

[a] Ceph intro https://docs.ceph.com/en/quincy/start/
[b] Ceph architecture https://docs.ceph.com/en/quincy/architecture/
[c] Ceph glossary https://docs.ceph.com/en/quincy/glossary

8.3 健康 Ceph 集群的建议

要构建超融合 Proxmox + Ceph 集群，必须至少使用三台服务器，并且最好使用相同配置 的服务器。

也请查看 Ceph 网站中的建议。

Note
以下建议应视为选择硬件时的粗略指导。因此，仍然必须根据具体需求进行调整。
应测试你的部署，并持续监控健康状态和性能。

CPU

Ceph 服务可分为两类：

- CPU 使用密集型，受益于较高的 CPU 基础频率和多个核心。此类别包括：
  - Object Storage Daemon (OSD) 服务
  - CephFS 使用的 Meta Data Service (MDS)
- 中等 CPU 使用型，不需要多个 CPU 核心。此类别包括：
  - Monitor (MON) 服务
  - Manager (MGR) 服务

作为简单经验法则，应为每个 Ceph 服务至少分配一个 CPU 核心（或线程），以提供 稳定且持久的 Ceph 性能所需的最低资源。

例如，如果计划在一个节点上运行一个 Ceph monitor、一个 Ceph manager 和 6 个 Ceph OSD 服务，并以基础且稳定的性能为目标，则应为 Ceph 专门保留 8 个 CPU 核心。

请注意，OSD 的 CPU 使用率主要取决于磁盘性能。磁盘可能提供的 IOPS（**IO O**perations per **S**econd）越高，OSD 服务可利用的 CPU 就越多。对于现代企业级 SSD 磁盘，例如可以长期承受超过 100’000 IOPS 且延迟低于毫秒级的 NVMe，每个 OSD 都可能使用多个 CPU 线程；对于非常高性能的 NVMe 支撑的 OSD 使用四到六 个 CPU 线程是可能的。

内存

尤其是在超融合部署中，需要仔细规划并监控内存消耗。除了虚拟机和容器的预期内存 使用量外，还必 Ceph 保留足够内存，以提供出色且稳定的性能。

作为经验法则，每个 OSD 大约会为 **1 TiB** 数据使用 **1 GiB** 内存。正常情况下使用量 可能更低，但在恢 backfilling 等关键操作期间会使用最多内存。这 意味着应避免在正常运行时就耗尽可用内存，而应保留

OSD 服务本身也会使用额外内存。该守护进程的 Ceph BlueStore 后端默认需要 **3-5 GiB** 内存（可调整

网络

建议至少使用 10 Gbps 或更高带宽的网络专门承载 Ceph 流量。对于三到五节点集群， 如果没有可用的 10+ Gbps 交换机，也可以选择网状网络部署 [1] 。

Important
流量规模，尤其是恢复期间的流量，会干扰同一网络上的其他服务；其中对延迟敏感的 Proxmox VE corosync 集群栈尤其可能受到影响，并可能导致集群 quorum 丢失。将 Ceph 流量迁移到专用且物理隔离的网络，可以避免这种干扰；这不仅适用于 corosync，也 适用于任何虚拟客户机提供的网络服务。

估算带宽需求时，需要考虑磁盘性能。单块 HDD 可能无法打满 1 Gb 链路，但每个节点上 多个 HDD OSD 也可能已经打满 10 Gbps。如果使用现代 NVMe SSD，单块磁盘就可能打满 10 Gbps 或更高带宽。对于这类高性能部署，建议至少使用 25 Gbps；为了充分利用底层 磁盘的性能潜力，甚至 40 Gbps 或 100+ Gbps。

如果不确定，对于高性能部署，建议使用三个（物理）独立网络：

- 一个超高带宽（25+ Gbps）网络，用于 Ceph（内部）集群流量。
- 一个高带宽（10+ Gbps）网络，用于 ceph server 和 ceph client 之间的 Ceph （public）存储流量。根据需求，也可用于承载虚拟客户机流量和虚拟机在线迁移流量。
- 一个中等带宽（1 Gbps）网络，专用于对延迟敏感的 corosync 集群通信。

[1] Full Mesh Network for Ceph https://docs.pxvirt.lierfang.comFull_Mesh_Network_for_Ceph_Server

磁盘

规划 Ceph 集群规模时，必须考虑恢复时间。尤其是在小型集群中，恢复可能需要较长 时间。建议在小 SSD 而不是 HDD，以缩短恢复时间，并尽量降低恢复期间 发生后续故障事件的可能性。

通常，SSD 会比机械磁盘提供更多 IOPS。考虑到这一点，并结合更高成本，实施基于 类别的池分离可 OSD 的 方法，是使用更快的磁盘作为 journal 或 DB/**W**rite-**A**head-**L**og 设备，参见 创建 Ceph OSD。如果将更快的磁盘用于多个 OSD，必须在 OSD 与 WAL/DB（或 journal）磁盘之间选择合适平 OSD 的瓶颈。

除磁盘类型外，当每个节点上的磁盘容量一致且数量均匀分布时，Ceph 性能最佳。例如， 每个节点使 4 块 500 GB 磁盘，优于混合使用一块 1 TB 磁盘和三块 250 GB 磁盘。

还需要平衡 OSD 数量和单个 OSD 容量。更高容量可以提升存储密度，但也意味着单个 OSD 故障会迫 Ceph 一次恢复更多数据。

避免 **RAID**

由于 Ceph 自行处理数据对象冗余以及对磁盘（OSD）的多路并行写入，使用 RAID 控制器通常不会提 的设计目标是自行处理整块磁盘，中间 不需要任何抽象层。RAID 控制器并非为 Ceph 工作负载设计，可 有时甚至降低性能，因为其写入和缓存算法可能会干扰 Ceph 的算法。

Warning
避免使用 RAID 控制器。请改用 host bus adapter（HBA）。

8.4 Ceph 初始安装与配置

8.4.1 使用基于 Web 的向导

使用 Proxmox VE 时，可以受益于易用的 Ceph 安装向导。点击某个集群节点，并在菜单树中 导航到 Ceph 部分。如果尚未安装 Ceph，你会看到安装提示。

向导分为多个部分，每一部分都需要成功完成，才能使用 Ceph。

首先需要选择要安装的 Ceph 版本。优先选择与其他节点相同的版本；如果这是第一个 安装 Ceph 的节点，则选择最新版本。

开始安装后，向导会从 Proxmox VE 的 Ceph 仓库下载并安装所有必需软件包。

完成安装步骤后，需要创建配置。每个集群只需要执行一次此步骤，因为该配置会通过 Proxmox VE 的集群化 配置文件系统（pmxcfs）自动分发 到所有剩余集群成员。

配置步骤包括以下设置：

- **Public Network:** 此网络用于 public 存储通信（例如使用 Ceph RBD 后端磁盘的 虚拟机，或 CephFS 挂载），以及不同 Ceph 服务之间的通信。此设置为必填项。
  强烈建议将 Ceph 流量与 Proxmox VE 集群通信（corosync）分离，并尽可能与虚拟客户机 面向前端的（public）网络分离。否则，Ceph 的高带宽 IO 流量可能会干扰其他依赖 低延迟的服务。

- **Cluster Network:** 指定是否也分离 OSD 复制和心跳流量。此 设置为可选项。
  建议使用物理隔离的网络，因为它可以减轻 Ceph public 网络和虚拟客户机网络的负载， 同时显著提
  Ceph 性能。
  Ceph cluster network 可以在之后配置并迁移到另一个物理隔离网络。

还有两个被视为高级设置的选项，因此只有在明确理解其作用时才应更改。

- **Number of replicas**: 定义对象被复制的次数。
- **Minimum replicas**: 定义将 I/O 标记为完成所需的最小副本数。

此外，还需要选择第一个 monitor 节点。此步骤为必需项。

至此完成。现在应在最后一步看到成功页面，其中包含后续操作说明。系统现在已准备好 开始使用 Ceph。要开始使用，需要创建一些额外的 monitors、 OSDs 以及至少一个 pool。

本章后续内容将指导你充分利用基于 Proxmox VE 的 Ceph 部署。这包括前述提示，以及更多内容，例如 CephFS；它是新 Ceph 集群的有用补充。

8.4.2 通过 CLI 安装 Ceph 软件包

除了推荐使用 Web 界面中的 Proxmox VE Ceph 安装向导外，也可以在每个节点上使用以下 CLI 命令：

```
pveceph install
```

这会在 `/etc/apt/sources.list.d/ceph.list` 中设置 apt 软件包仓库，并安装必需 软件。

8.4.3 通过 CLI 进行 Ceph 初始配置

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
```

这会在 `/etc/pve/ceph.conf` 创建初始配置，并为 Ceph 使用专用网络。该文件会通过 pmx-cfs 自动分发到所有 Proxmox VE 节点。该命令还会在 `/etc/ceph/ceph.conf` 创建指向该文件的符号
Ceph 命令， 而无需指定配置文件。

8.4.4 常用命令示例

以下命令可用于安装 Ceph、初始化集群网络，并查看 Ceph 当前状态：

```
pveceph install
pveceph init --network 10.10.10.0/24
ceph -s
```

8.5 Ceph Monitor

Ceph Monitor（MON）[2] 维护集群映射的主副本。为了实现高可用，至少需要 3 个 monitor。如果使
则已经安装了一个 monitor。只要集群为中小规模，就不需要超过 3 个 monitor。只有 非常大的集群才

8.5.1 创建 Monitor

在每个要放置 monitor 的节点上（建议使用三个 monitor），可通过 GUI 中的 *Ceph → Monitor* 选项卡创建，或运行：

```
pveceph mon create
```

8.5.2 销毁 Monitor

要通过 GUI 移除 Ceph Monitor，首先在树视图中选择一个节点，并进入 **Ceph → Monitor** 面板。选择 MON 并点击 **Destroy** 按钮。

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

```
pveceph mon destroy
```

Note
quorum 至少需要三个 Monitor。

8.6 Ceph Manager

Manager 守护进程与 monitor 一起运行。它提供用于监控集群的接口。自 Ceph luminous 发布以来 ceph-mgr 守护进程 [3]。

8.6.1 创建 Manager

可以安装多个 Manager，但任意时刻只有一个 Manager 处于 active 状态。

```
pveceph mgr create
```

Note
建议在 monitor 节点上安装 Ceph Manager。为了实现高可用，请安装多个 manager。

[2] Ceph Monitor https://docs.ceph.com/en/quincy/rados/configuration/mon-config-ref/
[3] Ceph Manager https://docs.ceph.com/en/quincy/mgr/

8.6.2 销毁 Manager

要通过 GUI 移除 Ceph Manager，首先在树视图中选择一个节点，并进入 **Ceph → Monitor** 面板。选择 Manager 并点击 **Destroy** 按钮。

要通过 CLI 移除 Ceph Manager，首先连接到运行该 Manager 的节点。然后执行以下 命令：

```
pveceph mgr destroy
```

Note
虽然 manager 不是硬依赖，但它对 Ceph 集群非常关键，因为它负责 PG-autoscaling、设备健康监控、telemetry 等重要功能。

8.7 Ceph OSDs

Ceph **O**bject **S**torage **D**aemons 通过网络为 Ceph 存储对象。建议每块物理磁盘 使用一个 OSD。

8.7.1 创建 OSD

可以通过 Proxmox VE Web 界面创建 OSD，也可以通过 CLI 使用 pveceph 创建。例如：

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

Tip
建议 Ceph 集群至少包含三个节点和至少 12 个 OSD，并在各节点之间均匀分布。

如果磁盘此前已被使用（例如用于 ZFS 或作为 OSD），首先需要清除所有使用痕迹。要 移除分区表、 OSD 残留，可以使用以下命令：

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

Warning
上述命令会销毁磁盘上的所有数据！

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 [4]. 自 Ceph Luminous 起，这是创建 OSD 时的默认类型。

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

[4] Ceph Bluestore https://ceph.com/community/new-luminous-bluestore/

Block.db 与 block.wal

如果希望为 OSD 使用单独的 DB/WAL 设备，可以通过 *-db_dev* 和 *-wal_dev* 选项 指定。如果未单独指
WAL，它会与 DB 放在一起。

```
pveceph osd create /dev/sd[X] -db_dev /dev/sd[Y] -wal_dev /dev/sd[Z]
```

可以分别使用 *-db_size* 和 *-wal_size* 参数直接选择它们的大小。如果未给出这些 参数，将按顺序使用以

- Ceph 配置中的 bluestore_block_{db,wal}_size...
  - ... database，*osd* 段
  - ... database，*global* 段
  - ... file，*osd* 段
  - ... file，*global* 段
- OSD 大小的 10%（DB）/1%（WAL）

> **Note**
> DB 存储 BlueStore 的内部元数据，WAL 是 BlueStore 的内部 journal 或 write-ahead log。建议使用快速 SSD 或 NVRAM 以获得更好性能。

Ceph Filestore

在 Ceph Luminous 之前，Filestore 是 Ceph OSD 的默认存储类型。自 Ceph Nautilus 起，Proxmox VE 不再支持使用 *pveceph* 创建此类 OSD。如果仍想创建 filestore OSD，请 直接使用 *ceph-volume*。

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

8.7.2 销毁 OSD

如果遇到 OSD 或其磁盘问题，请先尝试 排查，以决定是否 需要 替换。

要销毁 OSD，请导航到 *<Node>* → *Ceph* → *OSD* 面板，或在该 OSD 所在节点上使用 下述 CLI 命令。

1. 确保集群有足够空间处理 OSD 移除。 在 *Ceph* → *OSD* 面板中，如果待销毁 OSD 仍为 up 和 in（AVAIL 为非零值）， 请确保所有 OSD 的 Used (%) 值都明显低于默认 85% 的 nearfull_ratio。

   这样可以降低即将发生的重新平衡带来的风险；重新平衡可能导致 OSD 写满，从而阻塞 Ceph pool 上的 I/O。

   使用以下命令可在 CLI 上获取相同信息：

   ```
   ceph osd df tree
   ```

2. 如果待销毁 OSD 尚未 out，请选择该 OSD 并点击 **Out**。这会将它从数据分布中 排除，并启动重
   以下命令执行相同操作：
   ```
   ceph osd out <id>
   ```

3. 如果可以，请等待 Ceph 完成重新平衡，以确保始终有足够副本。该 OSD 会变为空； 一旦变空，*0 PGs*。

4. 点击 **Stop**。如果此时停止尚不安全，会出现警告，此时应点击 **Cancel**。稍等 片刻后再试。
   可以使用以下命令检查是否可以安全停止，并停止该 OSD：
   ```
   ceph osd ok-to-stop <id>
   pveceph stop --service osd.<id>
   ```

5. 最后：
   要从 Ceph 移除 OSD 并删除所有磁盘数据，首先点击 **More → Destroy**。启用 cleanup 选项以清理分区表和其他结构。这样即可在 Proxmox VE 中立即复用该磁盘。然后点击 **Remove**。
   销毁 OSD 的 CLI 命令为：
   ```
   pveceph osd destroy <id> [--cleanup]
   ```

8.8 Ceph Pools

pool 是用于存储对象的逻辑组。它包含一组对象集合，称为 **P**lacement **G**roups（PG，pg_num）。

8.8.1 创建和编辑 Pool

可以通过命令行创建和编辑 pool，也可以在任意 Proxmox VE 主机的 Web 界面中通过 **Ceph → Pools** 进行操作。

如果未给出选项，默认设置为 **128 PGs**、**size** 为 **3** 个副本以及 **min_size** 为 **2** 个副本，以确保任意 OSD 发生故障时不会丢失数据。

Warning
不要将 **min_size** 设置为 **1**。min_size 为 1 的 replicated pool 会允许在 对象只有 1 个副本时执行 I/O，这可能导致数据丢失、PG 不完整或对象无法找到。

建议启用 PG-Autoscaler，或根据部署情况计算 PG 数量。可以在线找到公式和 PG 计算器 [5]。从 Ceph Nautilus 开始，可以在部署后更改 PG 数量 [6]。

PG autoscaler [7] 可以在后台自动扩缩 pool 的 PG 数量。设置 Target Size 或 Target Ratio 高级参数，有助于 PG-Autoscaler 做出更好的决策。

[5] PG calculator https://web.archive.org/web/20210301111112/http://ceph.com/pgcalc/
[6] Placement Groups https://docs.ceph.com/en/quincy/rados/operations/placement-groups/
[7] Automated Scaling https://docs.ceph.com/en/quincy/rados/operations/placement-groups/#automated-scaling

通过 **CLI** 创建 **pool** 的示例

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

Tip
如果还希望自动为 pool 定义一个存储，请在 Web 界面中保持 'Add as Storage' 复选框选中，或在创建 pool 时使用命令行选项 *--add_storages*。

Pool 选项

以下选项可在创建 pool 时使用，其中部分选项也可在编辑 pool 时使用。

Name
pool 名称。必须唯一，之后不能更改。

Size
每个对象的副本数。Ceph 始终会尝试为对象保留这么多副本。默认值：3。

PG Autoscale Mode
该 pool 的自动 PG 扩缩模式 [7]。如果设置为 `warn`，当 pool 的 PG 数量不是最优时会生成警告消息

Add as Storage
使用新 pool 配置虚拟机或容器存储。默认值：`true`（仅在创建时 可见）。

高级选项

Min. Size
每个对象的最小副本数。如果某个 PG 的副本数低于该值，Ceph 会拒绝该 pool 上的 I/O。默认值

Crush Rule
用于在集群中映射对象放置位置的规则。这些规则定义数据在集群内的 放置方式。有关基于设备的
Ceph CRUSH & device classes。

# of PGs
pool 初始应拥有的 placement group 数量 [6]。 默认值：128。

Target Ratio
pool 中预期数据量的比例。PG autoscaler 使用该比例相对于其他比例集 进行计算。如果同时设
`target size`，该选项优先。

Target Size
pool 中预期数据量的估计值。PG autoscaler 使用此大小估算最优 PG 数量。

Min. # of PGs
placement group 的最小数量。该设置用于微调该 pool 的 PG 数量下限。 PG autoscaler 不会将 PG 合并到低于此阈值。

有关 Ceph pool 处理的更多信息，请参见 Ceph pool operation 手册 [8]。

[8] Ceph pool operation https://docs.ceph.com/en/quincy/rados/operations/pools/

8.8.2 纠删码 Pool

Erasure coding（EC）是一种 'forward error correction' 编码形式，允许从一定量的 数据丢失中恢复
replicated pools 相比，erasure coded pools 可以提供更多可用 空间，但代价是性能下降。

作为对比：在经典 replicated pools 中，会存储数据的多个副本（`size`）；而在 erasure coded pool 中，数据会拆分为 k 个数据块，并附加 m 个编码（校验）块。 当数据块缺失时，这些编码块可用于

编码块数量 `m` 定义了在不丢失任何数据的情况下可以丢失多少个 OSD。存储的对象总量 为 `k + m`。

创建 EC Pool

可以使用 pveceph CLI 工具创建 erasure coded（EC）pool。规划 EC pool 时需要考虑 它与 replicated pools 的工作方式不同。

EC pool 的默认 `min_size` 取决于 `m` 参数。如果 `m = 1`，EC pool 的 `min_size` 将为 k。如果 `m > 1`，`min_size` 将为 `k + 1`。Ceph 文档建议采用更保守的 `min_size`，即 `k + 2` [9].

如果可用 OSD 数量少于 `min_size`，该 pool 的任何 IO 都会被阻塞，直到再次有足够 OSD 可用。

> **Note**
> 规划 erasure coded pool 时，请关注 `min_size`，因为它定义了需要有多少 OSD 可用。否则，IO 会被阻塞。

例如，`k = 2` 且 `m = 1` 的 EC pool 将具有 `size = 3`、`min_size = 2`，并且在 一个 OSD 故障时仍可运行。如果 pool 配置为 `k = 2`、`m = 2`，则它会具有 `size = 4` 和 `min_size = 3`，并在丢失一个 OSD 时仍可运行。

要创建新的 EC pool，请运行以下命令：

```
pveceph pool create <pool-name> --erasure-coding k=2,m=1
```

可选参数包括 `failure-domain` 和 `device-class`。如果需要更改该 pool 使用的任何 EC profile 设置，则必须使用新 profile 创建新的 pool。

这会创建一个新的 EC pool，并创建所需的 replicated pool 来存储 RBD omap 和其他 元数据。最终会 `<pool name>-data` 和 `<pool name>-metadata` 两个 pool。默认 行为还会创建匹配的存储配置。要 `--add_storages 0` 参数禁用。手动配置存储配置时，请记住需要设置 `data-pool` 参数。只有这样 pool 才会用于存储数据对象。例如：

> **Note**
> 可选参数 `--size`、`--min_size` 和 `--crush_rule` 会用于 replicated metadata pool，但不会用于 erasure coded data pool。如果需要更改 data pool 上的 `min_size`，可以稍后执行。erasure coded pools 上不能更改 `size` 和 `crush_rule` 参数。

[9] Ceph Erasure Coded Pool Recovery https://docs.ceph.com/en/quincy/rados/operations/erasure-code/#erasure-coded-pool-recovery

如果需要进一步自定义 EC profile，可以直接使用 Ceph 工具创建 [10]，并使用 `profile` 参数指定要使用的 profile。

例如：

```
pveceph pool create <pool-name> --erasure-coding profile=<profile-name>
```

将 **EC Pool** 添加为存储

可以将现有 EC pool 作为存储添加到 Proxmox VE。其工作方式与添加 RBD pool 相同，但 需要额外的 `data-pool` 选项。

```
pvesm add rbd <storage-name> --pool <replicated-pool> --data-pool <ec-pool>
```

Tip
对于任何不由本地 Proxmox VE 集群管理的外部 Ceph 集群，不要忘记添加 `keyring` 和 `monhost` 选项。

8.8.3 销毁 Pool

要通过 GUI 销毁 pool，请在树视图中选择一个节点，并进入 **Ceph → Pools** 面板。 选择要销毁的 pool 并点击 **Destroy** 按钮。要确认销毁 pool，需要输入 pool 名称。

运行以下命令可销毁 pool。指定 *-remove_storages* 还会移除关联存储。

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

Note
Pool 删除会在后台运行，可能需要一些时间。在此过程中，你会看到集群中的数据使用量逐步下降。

8.8.4 PG Autoscaler

PG autoscaler 允许集群考虑每个 pool 中存储的（预期）数据量，并自动选择合适的 pg_num 值。它自 Ceph Nautilus 起可用。

在调整生效之前，可能需要先激活 PG autoscaler 模块。

```
ceph mgr module enable pg_autoscaler
```

autoscaler 按 pool 配置，并具有以下模式：

[10] Ceph Erasure Code Profile https://docs.ceph.com/en/quincy/rados/operations/erasure-code/-#erasure-code-profiles

warn	如果建议的 pg_num 值与当前值差异过大，则发出健康警告。
on	自动调整 pg_num，无需任何人工交互。
off	不自动调整 pg_num，即使 PG 数量不是最优，也不会发出警告。

可以使用 target_size、target_size_ratio 和 pg_num_min 选项调整扩缩因子，以便 适应未来

Warning
默认情况下，如果 pool 的 PG 数量偏差达到 3 倍，autoscaler 就会考虑调整 该 pool 的 PG 数量。这会导致数据放置发生明显变化，并可能给集群带来高负载。

可以在 Ceph Blog 中找到对 PG autoscaler 更深入的介绍： New in Nautilus: PG merging and autotuning.

8.9 Ceph CRUSH 与设备类别

CRUSH [11] （**C**ontrolled **R**eplication **U**nder **S**calable **H**ashing）算法是 Ceph 的基础。

CRUSH 计算数据应存储在哪里以及从哪里检索。其优势在于不需要中心化索引服务。 CRUSH 使用由 OSD、bucket（设备位置）和 pool 的 ruleset（数据复制）组成的映射来 工作。

Note
更多信息可在 Ceph 文档的 CRUSH map 章节中找到 [a]。

[a] CRUSH map https://docs.ceph.com/en/quincy/rados/operations/crush-map/

可以修改此映射以反映不同的复制层级。对象副本可以被分离（例如按 failure domain），同时保持其

一种常见配置是为不同 Ceph pool 使用不同类别的磁盘。因此，Ceph 从 luminous 开始 引入 device classes，以满足轻松生成 ruleset 的需求。

可以在 *ceph osd tree* 输出中看到设备类别。这些类别表示各自的 root bucket，可 通过以下命令查看

```
ceph osd crush tree --show-shadow
```

上述命令的示例输出：

```
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
```

[11] https://ceph.com/assets/pdfs/weil-crush-sc06.pdf

```
 -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
```

要指示 pool 仅在特定设备类别上分布对象，首先需要为该设备类别创建 ruleset：

```
ceph osd crush rule create-replicated <rule-name> <root> <failure-domain> < ↩
   class>
```

<rule-name>	规则名称，用于关联 pool（可在 GUI 和 CLI 中看到）
<root>	它应属于哪个 crush root（默认 Ceph root 为 "default"）
<failure-domain>	对象应在哪个 failure-domain 上分布（通常为 host）
<class>	要使用的 OSD 后端存储类型（例如 nvme、ssd、hdd）

规则进入 CRUSH map 后，即可让 pool 使用该 ruleset。

```
ceph osd pool set <pool-name> crush_rule <rule-name>
```

Tip
如果 pool 中已经包含对象，则必须相应移动这些对象。根据部署情况，这可能会对集群产生较大性能影响。作为替代方案，可以创建新 pool 并单独移动磁盘。

8.10 Ceph Client

按照前面章节完成设置后，可以配置 Proxmox VE 使用这些 pool 存储虚拟机和容器镜像。只需使用 GUI 添加新的 RBD 存储（参见 Ceph RADOS Block Devices (RBD) 章节）。

对于外部 Ceph 集群，还需要将 keyring 复制到预定义位置。如果 Ceph 安装在 Proxmox 节点本身，则

Note
文件名需要是 <storage_id> + `.keyring，其中 <storage_id> 是 /etc/pve/storage.cfg 中 *rbd:* 后面的表达式。在以下示例中，my-ceph-storage 就是 <storage_id>：

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

8.11 CephFS

Ceph 还提供文件系统，它运行在与 RADOS block devices 相同的对象存储之上。Ceph 使用 **M**eta**d**ata **S**erver（MDS）将 RADOS 后端对象映射为文件和目录，使 Ceph 能够提供兼容 POSIX 的复制文件系统。这使你可以轻松配置集群化、高可用的共享文件 系统。Ceph 的 Metadata Servers 保证文件在整个 Ceph 集群中均匀分布。因此，即使 在高负载场景下，也不会压垮单个主机；而这可能 NFS）中的问题。

Proxmox VE 既支持创建超融合 CephFS，也支持使用现有 CephFS as storage 保存备份、ISO 文件和容器模板。

8.11.1 Metadata Server（MDS）

CephFS 至少需要配置并运行一个 Metadata Server 才能工作。可以通过 Proxmox VE Web GUI 的 Node -> CephFS 面板创建 MDS，也可以使用命令行：

```
pveceph mds create
```

一个集群中可以创建多个 metadata server，但在默认设置下，任意时刻只有一个可以处于 active 状态。如果某个 MDS 或其节点无响应（或崩溃），另一个 standby MDS 会被 提升为 active。*hotstandby* 参数选项，加快 active 与 standby MDS 之间的切换；如果已经创建，也可以设置/添加

```
mds standby replay = true
```

到 /etc/pve/ceph.conf 中相应的 MDS 段。启用后，指定 MDS 会保持在 warm 状态，并轮询 active MDS，从而在出现问题时更快接管。

Note
这种主动轮询会对系统和 active MDS 带来额外性能影响。

多个 **Active MDS**

自 Luminous（12.2.x）起，可以同时运行多个 active metadata server，但通常只有在有 大量客户 很少是系统瓶颈。如果要设置此模式，请 参阅 Ceph 文档。 [12]

8.11.2 创建 CephFS

借助 Proxmox VE 对 CephFS 的集成，可以使用 Web 界面、CLI 或外部 API 接口轻松创建 CephFS。要使其正常工作，需要满足一些前提条件：

成功设置 CephFS 的前提条件：

- Install Ceph packages - 如果此前已经执行过，可能需要在 最新系统上重新运行，以确保安装所有 CephFS 相关软件包。
- Setup Monitors

[12] Configuring multiple active MDS daemons https://docs.ceph.com/en/quincy/cephfs/multimds/

- Setup your OSDs
- Setup at least one MDS

完成后，可以通过 Web GUI 的 Node -> CephFS 面板，或使用命令行工具 pveceph 创建 CephFS，

```
pveceph fs create --pg_num 128 --add-storage
```

这会创建名为 *cephfs* 的 CephFS，使用名为 *cephfs_data* 的数据 pool，其中包含 *128* 个 placement groups；并使用名为 *cephfs_metadata* 的元数据 pool，其中的 placement groups 数量为数 pool 的四分之一（32）。请查看 Proxmox VE managed Ceph pool chapter 或访问 Ceph 文档，了解适合 你部署的 placement group 数量（pg_num）[6]。此外，*--add-storage* 参数会在 CephFS 成功创建后，将其添加到 Proxmox VE 存储配置中。

8.11.3 销毁 CephFS

Warning
销毁 CephFS 会使其中所有数据不可用。此操作无法撤销！

要完整且平稳地移除 CephFS，需要执行以下步骤：

- 断开每个非 Proxmox VE 客户端（例如在客户机中卸载 CephFS）。
- 禁用所有相关的 CephFS Proxmox VE 存储条目（以防止其被自动挂载）。
- 从客户机中移除位于要销毁 CephFS 上的所有已使用资源（例如 ISO）。
- 使用以下命令在所有集群节点上手动卸载 CephFS 存储

  ```
  umount /mnt/pve/<STORAGE-NAME>
  ```

  其中 <STORAGE-NAME> 是 Proxmox VE 中 CephFS 存储的名称。
- 现在确保没有 metadata server（MDS）正在为该 CephFS 运行，可通过停止或销毁它们 来实现。这
  Web 界面完成，也可以通过命令行接口完成；对于后者，可执行 以下命令：

  ```
  pveceph stop --service mds.NAME
  ```

  以停止它们，或

  ```
  pveceph mds destroy NAME
  ```

  以销毁它们。

  请注意，当 active MDS 被停止或移除时，standby server 会自动提升为 active， 因此最好先停止用
  standby server。
- 现在可以使用以下命令销毁 CephFS

  ```
  pveceph fs destroy NAME --remove-storages --remove-pools
  ```

  这会自动销毁底层 Ceph pool，并从 pve 配置中移除存储。

完成这些步骤后，CephFS 应已被完全移除；如果还有其他 CephFS 实例，可以再次启动 已停止的 metadata server，让它们作为 standby 运行。

8.12 Ceph 维护

8.12.1 替换 OSD

通过以下步骤可以替换 OSD 的磁盘，这是 Ceph 中最常见的维护任务之一。如果某个 OSD 出现问题，
故障排查 章节。

1. 如果磁盘发生故障，请获取同类型、同容量的 推荐替换磁盘。
2. 销毁有问题的 OSD。
3. 从服务器上拆下旧磁盘并接入新磁盘。
4. 再次 创建 OSD。
5. 自动重新平衡后，集群状态应切回 HEALTH_OK。对于仍列出的任何 crash，可以运行 以下命令确

```
ceph crash archive-all
```

8.12.2 Trim/Discard

在虚拟机和容器中定期运行 *fstrim*（discard）是一项良好实践。这会释放文件系统 不再使用的数据块
磁盘发出这类 discard 命令。只需要确保虚拟机启用了 disk discard option。

8.12.3 Scrub & Deep Scrub

Ceph 通过对 placement group 进行 *scrubbing* 来确保数据完整性。Ceph 会检查 PG 中 每个对象的
有两种形式：每日执行的低开销元数据检查，以及每周 执行的深度数据检查。每周 deep scrub
会读取对象并使用校验和确保数据完整性。如果 正在运行的 scrub 干扰业务（性能）需求，可以调整
scrub [13] 执行时间。

8.12.4 关闭 Proxmox VE + Ceph HCI 集群

要关闭整个 Proxmox VE + Ceph 集群，首先停止所有 Ceph 客户端。这些主要是虚拟机和容器。
如果还有可能访问 Ceph FS 或已安装 RADOS GW 的其他客户端，也应将其停止。通过 Prox-
mox VE 工具关机时，高可用客户机会将其状态切换为 *stopped*。

一旦所有客户端、虚拟机和容器都已关闭，或不再访问 Ceph 集群，请确认 Ceph 集群 处于健康状态。
Web UI 或 CLI 执行：

```
ceph -s
```

要禁用所有自愈操作，并暂停 Ceph 集群中的任何客户端 IO，请在 **Ceph → OSD** 面板 中或通过
CLI 启用以下 OSD 标志：

[13] Ceph scrubbing https://docs.ceph.com/en/quincy/rados/configuration/osd-config-ref/#scrubbing

```
ceph osd set noout
ceph osd set norecover
ceph osd set norebalance
ceph osd set nobackfill
ceph osd set nodown
ceph osd set pause
```

开始关闭没有 monitor（MON）的节点。这些节点关闭后，再继续关闭运行 monitor 的 节点。

启动集群时，先启动带有 monitor（MON）的节点。所有节点启动并运行后，在取消设置 OSD 标志之 Ceph 服务都已启动并运行：

```
ceph osd unset pause
ceph osd unset nodown
ceph osd unset nobackfill
ceph osd unset norebalance
ceph osd unset norecover
ceph osd unset noout
```

现在可以启动客户机。高可用客户机开机后会将其状态更改为 *started*。

8.13 Ceph 监控与故障排查

从一开始就持续监控 Ceph 部署的健康状态非常重要，可以使用 Ceph 工具，也可以通过 Proxmox VE API 访问状态。

以下 Ceph 命令可用于查看集群是否健康（*HEALTH_OK*）、是否存在警告（*HEALTH_WARN*），甚至 下面的状态命令还会概览当前事件和应采取的操作。要停止执行，请按 CTRL-C。

持续观察集群状态：

```
watch ceph --status
```

打印一次集群状态（不会持续更新），并持续追加状态事件行：

```
ceph --watch
```

8.13.1 故障排查

本节包含常用故障排查信息。更多信息可在 Ceph 官方网站的故障排查部分找到 [14].

受影响节点上的相关日志

- 磁盘健康监控
- *System* → *System Log*，或通过 CLI 查看，例如最近 2 天：

  ```
  journalctl --since "2 days ago"
  ```

[14] Ceph troubleshooting https://docs.ceph.com/en/quincy/rados/troubleshooting/

- IPMI 和 RAID 控制器日志

可以运行以下命令列出并详细查看 Ceph 服务 crash：

```
ceph crash ls
ceph crash info <crash_id>
```

可运行以下命令确认标记为 new 的 crash：

```
ceph crash archive-all
```

要获得更详细视图，每个 Ceph 服务在 /var/log/ceph/ 下都有日志文件。如果需要更多 细节，可以[15]。

Ceph 问题的常见原因

- 网络问题，例如拥塞、交换机故障、接口关闭或防火墙阻断。检查所有 Proxmox VE 节点在 corosyn 集群网络以及 Ceph public 和 cluster 网络上是否都可可靠访问。
- 磁盘或连接部件存在以下情况：
  - 有缺陷
  - 未牢固安装
  - 在较高负载下缺少 I/O 性能（例如使用 HDD、消费级硬件或 不建议使用的 RAID 控制器时）
- 未满足健康 Ceph 集群的 建议。

常见 Ceph 问题

OSDs down/crashed

故障 OSD 会被报告为 down，并且通常会在 10 分钟后（自动）变为 out。根据原因，它也可能自动再次变为 up 和 in。要尝试通过 Web 界面手动激活，请进入 *Any node* → *Ceph* → *OSD*，选择该 OSD，并点击 **Start**、**In** 和 **Reload**。 使用 shell 时，请在受影响

```
ceph-volume lvm activate --all
```

要激活失败的 OSD，可能需要 安全重启相应节点； 或者作为最后手段，重新创建或替换该 OSD。

[15] Ceph log and debugging https://docs.ceph.com/en/quincy/rados/troubleshooting/log-and-debug/

Chapter 9

存储复制

pvesr 命令行工具用于管理 Proxmox VE 存储复制框架。存储复制可为使用本地存储的 来宾提供冗余

它会将来宾卷复制到另一个节点，因此无需使用共享存储即可让所有数据保持可用。 复制使用快照来尽以增量方式发送。如果发生节点故障，来宾数据仍可在复制目标节点上使用。

复制会按可配置的间隔自动完成。最小复制间隔为一分钟，最大间隔为每周一次。用于指定 这些间隔的 systemd 日历事件的一个子集，请参见 调度格式 章节：

可以将一个来宾复制到多个目标节点，但不能对同一个目标节点复制两次。

可以限制每个复制任务的带宽，以避免存储或服务器过载。

如果将来宾迁移到一个已拥有该来宾副本的节点，则只需要传输自上次复制以来的变更（即所谓的 deltas）。这会显著减少所需时间。如果将来宾迁移到复制目标节点， 复制方向会自动切换。

例如：VM100 当前位于 nodeA，并复制到 nodeB。将其迁移到 nodeB 后，它现在会 自动从 nodeB 反向复制回 nodeA。

如果迁移到一个没有该来宾副本的节点，则必须传输完整磁盘数据。迁移完成后，复制 作业会继续将该

Important
高可用可以与存储复制结合使用，但从最后一次同步到节点发生故障之间可能会有部分数据丢失。

9.1 支持的存储类型

Table 9.1: 存储类型

描述	插件类型	快照	稳定
ZFS (local)	zfspool	是	是

9.2 调度格式

复制使用 日历事件 配置调度。

9.3 错误处理

如果复制作业遇到问题，会被置于错误状态。在此状态下，已配置的复制间隔会被临时挂起。 失败的复
30 分钟为间隔反复重试。一旦重试成功，原始调度会再次激活。

9.3.1 可能的问题

下列是一些最常见的问题。根据具体设置，也可能存在其他原因。

- 网络未正常工作。
- 复制目标存储没有剩余可用空间。
- 目标节点上没有相同存储 ID 的存储。

Note
始终可以使用复制日志查找问题原因。

9.3.2 发生错误时迁移来宾

在发生严重错误时，虚拟来宾可能会滞留在故障节点上。此时需要手动将其再次移动到 正常工作的节点

9.3.3 示例

假设有两个来宾（VM 100 和 CT 200）运行在节点 A 上，并复制到节点 B。 节点 A 发生故障且无法恢
B。

- 通过 ssh 连接到节点 B，或通过 Web UI 打开其 shell
- 检查集群是否具备法定票数

```
# pvecm status
```

- 如果没有法定票数，强烈建议先修复该问题，使节点重新可操作。只有在当前无法做到这一点时，
  才可以使用以下命令在当前节点上强制设置法定票数：

```
# pvecm expected 1
```

Warning
设置 expected votes 后，应尽一切可能避免会影响集群的变更（例如添加/移除节点、存储、虚拟来宾）。仅应将其用于让关键来宾重新运行，或用于解决法定票数问题本身。

- 将两个来宾配置文件从原始节点 A 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
```

- 现在可以再次启动来宾：

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

请记得将 VMID 和节点名称替换为实际值。

9.4 管理作业

可以使用 Web GUI 轻松创建、修改和移除复制作业。此外，也可以使用命令行接口（CLI）工具 pvesr 完成这些操作。

在 Web GUI 的所有层级（数据中心、节点、虚拟来宾）都可以找到复制面板。不同层级显示 的作业范

添加新作业时，如果尚未选中来宾，则需要指定来宾以及目标节点。如果不希望使用默认的 all 15 minutes，可以设置复制 调度。 也可以为复制作业施加速率限制。速率限制有助于让存储负载保持在可

复制作业由一个集群范围内唯一的 ID 标识。该 ID 由 VMID 加上作业编号组成。 只有使用 CLI 工具时才需要手动指定该 ID。

9.5 网络

复制流量会使用与来宾在线迁移相同的网络。默认情况下，这是管理网络。若要为迁移使用 其他网络，Web 界面的 Datacenter -> Options -> Migration Settings 中配置 Migration Network datacenter.cfg 中配置。更多详情请参见 迁移网络。

9.6 命令行接口示例

为 ID 为 100 的来宾创建一个每 5 分钟运行一次、带宽限制为 10 Mbps（兆字节每秒）的 复制作业。

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

禁用 ID 为 100-0 的活动作业。

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

启用 ID 为 100-0 的已停用作业。

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

将 ID 为 100-0 的作业调度间隔改为每小时一次。

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

Chapter 10

QEMU/KVM 虚拟机

QEMU（Quick Emulator 的简称）是一个开源虚拟化管理程序，用于模拟一台物理计算机。 从运行 QEMU 的主机系统角度看，QEMU 是一个用户程序，可以访问分区、文件、网卡等本地资源， 并将这些

运行在模拟计算机中的客户操作系统会访问这些设备，并像运行在真实硬件上一样工作。 例如，可以将 ISO 镜像作为参数传递给 QEMU，模拟计算机中的操作系统就会看到一张真实的 CD-ROM 已插入 CD 驱动器。

QEMU 可以模拟从 ARM 到 Sparc 的多种硬件，但 Proxmox VE 只关注 32 位和 64 位 PC 兼容机模拟， 因为它代表了绝大多数服务器硬件。PC 兼容机模拟也是速度最快的模拟类型之一， 这是因为当模拟架构与主机架构相同时，可用的处理器扩展可以显著加速 QEMU。

Note
有时会遇到术语 *KVM*（Kernel-based Virtual Machine）。它表示 QEMU 通过 Linux KVM 模块，在虚拟化处理器扩展的支持下运行。在 Proxmox VE 语境中，*QEMU* 和 *KVM* 可以互换使用，因为 Proxmox VE 中的 QEMU 总是会尝试加载 KVM 模块。

Proxmox VE 中的 QEMU 以 root 进程运行，因为访问块设备和 PCI 设备需要这样的权限。

10.1 模拟设备和半虚拟化设备

QEMU 模拟的 PC 硬件包括主板、网络控制器、SCSI、IDE 和 SATA 控制器、串口等（完整列表可见 `kvm(1)` 手册页），它们全部由软件模拟。这些设备都是现有硬件设备的精确软件等价物； 如果客户机 这使 QEMU 能够运行_未修改_的操作系统。

不过这会带来性能成本，因为用软件运行原本应由硬件完成的工作，会让主机 CPU 承担大量额外任务。 为缓解这一问题，QEMU 可以向客户操作系统呈现_半虚拟化设备_，使客户操作系统知道自己运行在 QEMU 内部，并与虚拟化管理程序协作。

QEMU 依赖 virtio 虚拟化标准，因此能够呈现半虚拟化的 virtio 设备，包括半虚拟化通用磁盘控制器、半虚拟化网卡、半虚拟化串口、半虚拟化 SCSI 控制器等。

Tip
只要条件允许，*强烈建议*使用 virtio 设备，因为它们能显著提升性能，并且通常维护得更好。根据 bonnie++(8) 测量，与模拟 IDE 控制器相比，使用 virtio 通用磁盘控制器可使顺序写入吞吐量翻倍。 根据 iperf(1) 测量，使用 virtio 网络接口最多可达到模拟 Intel E1000 网卡三倍的吞吐量。 [a]

[a] 参见 KVM wiki 上的此基准测试 https://www.linux-kvm.org/page/Using_VirtIO_NIC

10.2 虚拟机设置

一般而言，Proxmox VE 会尝试为虚拟机（VM）选择合理默认值。请确保理解所修改设置的含义，因为错误设置可能导致性能下降，或使数据面临风险。

10.2.1 常规设置

虚拟机的常规设置包括：

- **Node**：运行该 VM 的物理服务器
- **VM ID**：此 Proxmox VE 安装中用于标识该 VM 的唯一编号
- **Name**：可用于描述该 VM 的自由格式文本字符串
- **Resource Pool**：VM 的逻辑分组

10.2.2 操作系统设置

创建虚拟机（VM）时，设置正确的操作系统（OS）可让 Proxmox VE 优化一些低层参数。 例如，Wir
操作系统期望 BIOS 时钟使用本地时间，而基于 Unix 的操作系统期望 BIOS 时钟使用 UTC 时间。

10.2.3 系统设置

创建 VM 时，可以更改新 VM 的一些基础系统组件。可以指定要使用的 显示类型。

此外，也可以更改 SCSI 控制器。 如果计划安装 QEMU Guest Agent，或者所选 ISO 镜像已经附带并会
可以勾选 *QEMU Agent* 选项，让 Proxmox VE 知道可以使用其功能显示更多信息，并更智能地完成某

Proxmox VE 允许使用不同的固件和机器类型启动 VM，即 SeaBIOS 和 OVMF。 在大多数情况下，只有
PCIe 直通时，才需要从默认 SeaBIOS 切换到 OVMF。

机器类型

VM 的 *Machine Type* 定义了 VM 虚拟主板的硬件布局。可以在默认的 Intel 440FX 与 Q35 芯片组之间选择；Q35 还提供虚拟 PCIe 总线，因此在需要直通 PCIe 硬件时可能更适合。 此外，还可以 vIOMMU 实现。

每种机器类型在 QEMU 中都有版本，一个给定的 QEMU 二进制文件支持多个机器版本。 新版本可能带 为了避免从客户机角度发生突然变化，并确保 VM 状态、在线迁移以及带 RAM 快照的兼容性， 新的 QEMU 实例会继续使用相同机器版本。

对于 Windows 客户机，机器版本会在创建期间固定，因为 Windows 对虚拟硬件变化很敏感， 即使是在冷启动之间也是如此。例如，不同机器版本下网络设备枚举可能不同。 Linux 等其他操作系统 *Latest* 机器版本。 这意味着在全新启动后，会使用 QEMU 二进制文件支持的最新机器版本（例如 QEMU 8.1 支持的最新机器版本是每种机器类型的 8.1 版本）。

在实现会改变硬件布局的新功能或修复时，机器版本也作为一种保护机制，用于确保向后兼容性。 对于正在运行的 VM 上的操作（例如在线迁移），会保存运行中的机器版本，以确保 VM 能够精确恢复 这不仅从 QEMU 虚拟化角度如此，也包括 Proxmox VE 如何创建 QEMU 虚拟机实例。

PVE 机器修订版

有时 Proxmox VE 需要在不等待新 QEMU 发布的情况下更改硬件布局或修改选项。 为此，Proxmox VE 添加了一个额外的下游修订版，形式为 +pveX。 在这些修订版中，每个新的 QEMU 机器版本的 X 为 0，并在这种情况下省略； 例如机器版本 `pc-q35-9.2` 与机器版本 `pc-q35-9.2+pve0` 相同。

如果 Proxmox VE 要更改硬件布局或默认选项，则会递增修订版，并用于新创建的客户机， 或用于始终 VM 在重启后生效。

从 QEMU 10.1 开始，机器版本会在 6 年后从上游 QEMU 中移除。在 Proxmox VE 中，主版本大约每 2 年发布一次， 因此一个 Proxmox VE 主版本将支持大约前两个 Proxmox VE 主版本中的机器版本。

在升级到新的 Proxmox VE 主版本之前，应更新 VM 配置，避免使用将在下一个 Proxmox VE 主版本中被移除的所有机器版本。 这可确保客户机在整个该版本周期内仍然可用。参见 更新到较新的机

该移除策略尚未对 Proxmox VE 8 生效，因此受支持机器版本的基线为 2.4。 预计 Proxmox VE 9 发布的最后一个 QEMU 二进制版本为 QEMU 11.2。 该 QEMU 二进制文件将移除对早于 6.0 的机器版本支持，因此 6.0 是 Proxmox VE 9 发布生命周期的基线。 预计每个 Proxmox VE 主版本的基线都会提高 2 个主版本，例如 Proxmox VE 10 的基线为 8.0。

如果看到弃用警告，应将机器版本更改为较新的版本。请先确保有可用备份，并准备好处理客户机看到 在某些场景中，可能需要重新安装特定驱动。还应检查是否存在使用这些机器版本创建的带 RAM 快照（即 `runningmachine` 配置项）。遗憾的是，无法更改快照的机器版本，因此需要加载该快照来

10.2.4 硬盘

总线/控制器

QEMU 可以模拟多种存储控制器：

Tip
出于性能原因，也因为维护状况更好，强烈建议使用 **VirtIO SCSI** 或 **VirtIO Block** 控制器。

- **IDE** 控制器的设计可追溯到 1984 年的 PC/AT 磁盘控制器。即使该控制器已被较新的设计取代，你能想到的几乎所有操作系统都支持它，因此如果要运行 2003 年之前发布的操作系统，它是一个很该控制器最多可连接 4 个设备。
- **SATA**（Serial ATA）控制器出现于 2003 年，设计更现代，允许更高吞吐量并可连接更多设备。该控制器最多可连接 6 个设备。
- **NVME** 控制器。该控制器最多可连接 6 个设备。
- **SCSI** 控制器设计于 1985 年，常见于服务器级硬件，最多可连接 14 个存储设备。 Proxmox VE 默认模拟 LSI 53C895A 控制器。

  如果追求性能，建议使用 *VirtIO SCSI single* 类型的 SCSI 控制器，并为附加磁盘启用 IO Thread 设置。从 Proxmox VE 7.3 起，这是新建 Linux VM 的默认设置。 每个磁盘都会拥有自己的 *VirtIO SCSI* 控制器，QEMU 会在专用线程中处理磁盘 IO。 Linux 发行版自 2012 年起支持该控制器自 2014 年起支持。 对于 Windows 操作系统，在安装期间需要额外提供包含驱动程序的 ISO。
- **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。 从功能角度 VirtIO SCSI 控制器取代。

镜像格式

在每个控制器上，可以附加若干模拟硬盘；这些硬盘由位于已配置存储中的文件或块设备提供后端。存储类型的选择将决定硬盘镜像格式。呈现块设备的存储（LVM、ZFS、Ceph）需要使用 **raw disk image format**，而基于文件的存储（Ext4、NFS、CIFS、GlusterFS）可在 **raw disk image format** 和 **QEMU image format** 之间选择。

- **QEMU image format** 是一种写时复制格式，支持快照和磁盘镜像精简配置。
- **raw disk image** 是硬盘的逐位镜像，类似于在 Linux 中对块设备执行 dd 命令得到的结果。该格式本身不支持精简配置或快照，需要存储层配合完成这些任务。不过，它可能比 **QEMU image format** 快最多 10%。[1]
- 只有在打算将磁盘镜像导入/导出到其他虚拟化管理程序时，**VMware image format** 才有意义。

缓存模式

设置硬盘的 **Cache** 模式会影响主机系统如何向客户系统通知块写入完成。 默认的 **No cache** 表示当每个块到达物理存储写入队列时，客户系统会被通知写入完成，同时忽略主机页缓存。这在安全

如果希望 Proxmox VE 备份管理器在备份 VM 时跳过某个磁盘，可以在该磁盘上设置 **No backup** 选项。

如果希望 Proxmox VE 存储复制机制在启动复制作业时跳过某个磁盘，可以在该磁盘上设置 **Skip replication** 选项。从 Proxmox VE 5.0 起，复制要求磁盘镜像位于 `zfspool` 类型的存储上；因此，在 VM 已配置复制的情况下，如果向其他存储添加磁盘镜像，就需要为该磁盘镜像跳过复制。

[1] 详见此基准测试 https://events.static.linuxfound.org/sites/events/files/slides/-CloudOpen2013_Khoa_Huynh_v3.pdf

Trim/Discard

如果存储支持_精简配置_（参见 Proxmox VE 指南中的存储章节），可以在驱动器上启用 **Discard** 选项。 设置 **Discard** 并使用支持 *TRIM* 的客户操作系统 [2] 时，当 VM 文件系统在删除文件后将块控制器会将此信息转发给存储，随后存储会相应缩小磁盘镜像。 为了让客户机能够发出 *TRIM* 命令，必 **Discard** 选项。 某些客户操作系统可能还要求设置 **SSD Emulation** 标志。 请注意，**VirtIO Block** 驱动器上的 **Discard** 仅支持使用 Linux Kernel 5.0 或更高版本的客户机。

如果希望某个驱动器在客户机中呈现为固态硬盘而不是旋转硬盘，可以在该驱动器上设置 **SSD emulation** 选项。底层存储不要求实际由 SSD 支撑；此功能可用于任何类型的物理介质。 请注意，**Virt Block** 驱动器不支持 **SSD emulation**。

IO Thread

选项 **IO Thread** 只能在使用 **VirtIO** 控制器的磁盘时使用；或者在使用 **SCSI** 控制器且模拟控制器类型 **VirtIO SCSI single** 时使用。启用 **IO Thread** 后，QEMU 会为每个存储控制器创建一个 I/O 线程，而不是在主事件循环或 vCPU 线程中处理所有 I/O。其好处之一是更好地分配工作并利用底层存 另一个好处是在主机工作负载 I/O 非常密集时，减少客户机中的延迟（卡顿），因为主线程和 vCPU 线程都不会被磁盘 I/O 阻塞。

10.2.5 CPU

CPU socket 是 PC 主板上可插入 CPU 的物理插槽。该 CPU 可以包含一个或多个 **core**，这些核心是 4 个核心的 CPU 插槽，和两个各包含 2 个核心的 CPU 插槽通常差别不大。不过，某些软件许可证依赖 将插槽数量设置为许可证允许的数量是有意义的。

增加虚拟 CPU（核心和插槽）数量通常会提升性能，但这高度依赖 VM 的用途。 多线程应用当然会受益 CPU，因为每增加一个虚拟 CPU，QEMU 都会在主机系统上创建一个新的执行线程。 如果不确定 VM 的工作负载，通常将 **Total cores** 设置为 2 是比较稳妥的选择。

Note
如果所有 VM 的_总_核心数大于服务器核心数，这是完全安全的（例如，在只有 8 个核心的机器上运行 4 个 VM，每个 VM 4 个核心，总计 16 个核心）。 在这种情况下，主机系统会在服务器核心之间平衡 QEMU 执行线程，就像运行标准多线程应用一样。 不过，Proxmox VE 会阻止启动虚拟 CPU 核心数超过物理可用核心数的 VM， 因为这只会由于上下文切换成本而降低性能。

资源限制

cpulimit

除了虚拟核心数量之外，还可以通过 **cpulimit** 选项设置 VM 可用的总 “Host CPU Time”。 这是一个表 CPU 时间百分比的浮点值，因此 `1.0` 等于 `100%`，`2.5` 等于 `250%`，依此类推。 如果单个进程完全使用 CPU Time 使用率为 `100%`。 如果一个具有四个核心的 VM 完全使用所有核心，理论上会使用 `400%`。 实际使用率可能还会略高，因为除 vCPU 核心线程外，QEMU 还可能为 VM 外设创建额外线程。

[2] TRIM, UNMAP, and discard https://en.wikipedia.org/wiki/Trim_%28computing%29

当 VM 因并行运行某些进程而需要多个 vCPU，但整体上不应同时让所有 vCPU 以 100% 运行时，此设

例如，假设某个虚拟机拥有 8 个虚拟 CPU 会受益，但你不希望它能够在满负载下占满全部 8 个核心，因为这会使服务器过载，并让其他虚拟机和容器获得过少 CPU 时间。 为解决此问题，可以将 **cpulimi** 设置为 `4.0`（=400%）。 这意味着如果 VM 通过同时运行 8 个进程完全利用全部 8 个虚拟 CPU，每个 vCPU 最多会从物理核心获得 50% CPU 时间。 但是，如果 VM 工作负载只完全利用 4 个虚拟 CPU，它仍可从物理核心获得最多 100% CPU 时间，总计 400%。

> **Note**
> 根据配置不同，VM 可能会使用额外线程，例如用于网络或 IO 操作，也可能用于在线迁移。因此，VM 显示的 CPU 时间可能超过其虚拟 CPU 本身可使用的时间。要确保 VM 使用的 CPU 时间永不超过分配的 vCPU， 请将 **cpulimit** 设置为与总核心数相同的值。

cpuunits

通过 **cpuunits** 选项（现在通常称为 CPU shares 或 CPU weight），可以控制某个 VM 相对于其他正 VM 获得多少 CPU 时间。它是一个相对权重，默认值为 `100`（如果主机使用旧版 cgroup v1，则为 `1024`）。 如果为某个 VM 增大该值，调度器会相对于权重较低的其他 VM 优先调度它。

例如，如果 VM 100 设置为默认值 `100`，而 VM 200 改为 `200`，后者 VM 200 将获得前者 VM 100 两倍的 CPU 带宽。

更多信息请参见 `man systemd.resource-control`，其中 `CPUQuota` 对应 `cpulimit`，`CPUWeigh` 对应这里的 `cpuunits` 设置。参见其 Notes 章节以了解参考资料和实现细节。

affinity

通过 **affinity** 选项，可以指定用于运行 VM vCPU 的物理 CPU 核心。 VM 外围进程（例如用于 I/O 的进程）不受此设置影响。请注意，**CPU affinity** 不是安全功能。

在某些情况下强制 CPU **affinity** 是有意义的，但会增加复杂性和维护工作量。 例如，稍后想要添加更多 VM，或将 VM 迁移到 CPU 核心更少的节点时。 如果某些 CPU 被完全占用而其他 CPU 几乎空闲，也很

affinity 通过 `taskset` CLI 工具设置。它接受主机 CPU 编号（见 `lscpu`），格式为 `man cpuset` 中的 `List Format`。这个 ASCII 十进制列表可以包含数字，也可以包含数字范围。 例如，**affinity** `0-1,8-11`（展开为 `0, 1, 8, 9, 10, 11`）将只允许 VM 在这六个特定主机核心上运行。

CPU 类型

QEMU 可以模拟从 486 到最新 Xeon 处理器的多种 **CPU types**。 每一代新处理器都会增加新功能，例 3D 渲染、随机数生成、内存保护等。 当前代处理器也可以通过 微码更新获得缺陷修复或安全修复。

通常应为 VM 选择与主机系统 CPU 非常接近的处理器类型，因为这意味着主机 CPU 功能（也称为 *CPU flags*）将在 VM 中可用。如果想要精确匹配，可以将 CPU 类型设置为 **host**，此时 VM 将拥有与主 CPU flags。

不过这也有缺点。如果要在不同主机之间对 VM 执行在线迁移，VM 可能最终运行在具有不同 CPU 类型或不同微码版本的新系统上。 如果传递给客户机的 CPU flags 缺失，QEMU 进程将停止。为解决此 也提供自己的虚拟 CPU 类型， Proxmox VE 默认使用这些类型。

后端默认值为 *kvm64*，它基本适用于所有 x86_64 主机 CPU；在 UI 中创建新 VM 时的默认值为 *x86-64-v2-AES*，该值要求主机 CPU 对 Intel 至少为 Westmere，对 AMD 至少为第四代 Opteron。

简而言之：

如果不关心在线迁移，或者拥有同构集群，其中所有节点具有相同 CPU 和相同微码版本，则将 CPU 类型设置为 host； 理论上这会为客户机提供最高性能。

如果关心在线迁移和安全性，并且只有 Intel CPU 或只有 AMD CPU，则选择集群中最低代的 CPU 型号。

如果关心在线迁移但不关心安全性，或者拥有混合 Intel/AMD 集群，则选择最低兼容的虚拟 QEMU CPU 类型。

Note
Intel 和 AMD 主机 CPU 之间的在线迁移不保证可以工作。

另请参见 QEMU 中定义的 AMD 和 Intel CPU 类型列表。

QEMU CPU 类型

QEMU 还提供虚拟 CPU 类型，同时兼容 Intel 和 AMD 主机 CPU。

Note
要为虚拟 CPU 类型缓解 Spectre 漏洞，需要添加相关 CPU flags，参见 Meltdown / Spectre 相关 CPU flags。

历史上，Proxmox VE 使用 *kvm64* CPU 型号，启用的 CPU flags 处于 Pentium 4 水平，因此对某些工作负载而言性能并不理想。

2020 年夏季，AMD、Intel、Red Hat 和 SUSE 协作，在 x86-64 基线之上定义了三个 x86-64 微架构级别，并启用现代 flags。详情参见 x86-64-ABI specification。

Note
一些较新的发行版（如 CentOS 9）现在以 *x86-64-v2* flags 作为最低构建要求。

- *kvm64 (x86-64-v1)*：兼容 Intel CPU >= Pentium 4、AMD CPU >= Phenom。
- *x86-64-v2*：兼容 Intel CPU >= Nehalem、AMD CPU >= Opteron_G3。 与 *x86-64-v1* 相比新增 CPU flags：+*cx16*、+*lahf-lm*、+*popcnt*、+*pni*、 +*sse4.1*、+*sse4.2*、+*ssse3*。
- *x86-64-v2-AES*：兼容 Intel CPU >= Westmere、AMD CPU >= Opteron_G4。 与 *x86-64-v2* 相比新增 CPU flags：+*aes*。
- *x86-64-v3*：兼容 Intel CPU >= Broadwell、AMD CPU >= EPYC。 与 *x86-64-v2-AES* 相比新增 CPU flags：+*avx*、+*avx2*、+*bmi1*、+*bmi2*、 +*f16c*、+*fma*、+*movbe*、+*xsave*。
- *x86-64-v4*：兼容 Intel CPU >= Skylake、AMD CPU >= EPYC v4 Genoa。 与 *x86-64-v3* 相比新增 CPU flags：+*avx512f*、+*avx512bw*、+*avx512cd*、 +*avx512dq*、+*avx512vl*。

自定义 **CPU** 类型

可以指定包含可配置功能集的自定义 CPU 类型。这些类型由管理员在配置文件 /etc/pve/virtual-g 中维护。格式详情参见 man cpu-models.conf。

任何在 /nodes 上具有 Sys.Audit 权限的用户都可以选择已指定的自定义类型。 通过 CLI 或 API 为 VM 配置自定义 CPU 类型时，名称需要带有 *custom-* 前缀。

Meltdown / Spectre 相关 **CPU flags**

有几个 CPU flags 与 Meltdown 和 Spectre 漏洞 [3] 相关；除非 VM 所选 CPU 类型已经默认启用这些 flags，否则需要手动设置。

要使用这些 CPU flags，需要满足两个要求：

- 主机 CPU 必须支持该功能，并将其传播给客户机的虚拟 CPU
- 客户操作系统必须更新到能够缓解这些攻击并使用该 CPU 功能的版本

否则，需要设置虚拟 CPU 所需的 CPU flag，可以通过在 Web UI 中编辑 CPU 选项， 或在 VM 配置文件中设置 *cpu* 选项的 *flags* 属性。

对于 Spectre v1、v2、v4 修复，CPU 或系统厂商还需要为你的 CPU 提供所谓的 “microcode update”，参见 固件更新章节。 请注意，并非所有受影响 CPU 都可以更新到支持 spec-ctrl。

要检查 Proxmox VE 主机是否存在漏洞，请以 root 身份执行以下命令：

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

也可以使用社区脚本检测主机是否仍然存在漏洞。 [4]

Intel 处理器

- *pcid*

  这会降低名为 *Kernel Page-Table Isolation (KPTI)* 的 Meltdown (CVE-2017-5754) 缓解措施带来会有效地向用户空间隐藏内核内存。 没有 PCID 时，KPTI 是一种成本相当高的机制 [5]。

  要检查 Proxmox VE 主机是否支持 PCID，请以 root 身份执行以下命令：

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

  如果返回结果不为空，则主机 CPU 支持 *pcid*。

- *spec-ctrl*

  在 retpolines 不足的情况下，需要该标志以启用 Spectre v1 (CVE-2017-5753) 和 Spectre v2 (CVE-2017-5715) 修复。 带 -IBRS 后缀的 Intel CPU 型号默认包含该标志。 对于不带 -IBRS 后缀的 Intel CPU 型号，必须显式启用。 需要更新后的主机 CPU 微码（intel-microcode >= 20180425）。

[3] Meltdown Attack https://meltdownattack.com/
[4] spectre-meltdown-checker https://meltdown.ovh/
[5] PCID is now a critical performance/security feature on x86 https://groups.google.com/forum/m/-#!topic/mechanical-sympathy/L9mHTbeQLNU

- *ssbd*

  需要该标志以启用 Spectre V4 (CVE-2018-3639) 修复。任何 Intel CPU 型号默认都不包含它。所有 Intel CPU 型号都必须显式启用。 需要更新后的主机 CPU 微码（intel-microcode >= 20180703）。

AMD 处理器

- *ibpb*

  在 retpolines 不足的情况下，需要该标志以启用 Spectre v1 (CVE-2017-5753) 和 Spectre v2 (CVE-2017-5715) 修复。 带 -IBPB 后缀的 AMD CPU 型号默认包含该标志。 对于不带 -IBPB 后缀的 AMD CPU 型号，必须显式启用。 在可供客户 CPU 使用前，需要主机 CPU 微码支持此功

- *virt-ssbd*

  需要该标志以启用 Spectre v4 (CVE-2018-3639) 修复。 任何 AMD CPU 型号默认都不包含它。所有 AMD CPU 型号都必须显式启用。 为了最大化客户机兼容性，即使也提供了 amd-ssbd，也应向请注意，使用 "host" CPU 型号时必须显式启用它，因为这是一个不存在于物理 CPU 中的虚拟功能。

- *amd-ssbd*

  需要该标志以启用 Spectre v4 (CVE-2018-3639) 修复。 任何 AMD CPU 型号默认都不包含它。所有 AMD CPU 型号都必须显式启用。 它提供的性能高于 virt-ssbd，因此支持该功能的主机应尽可能始终 尽管如此，为了最大化客户机兼容性，也应暴露 virt-ssbd，因为某些内核只识别 virt-ssbd。

- *amd-no-ssb*

  建议使用该标志来表明主机不受 Spectre V4 (CVE-2018-3639) 影响。 任何 AMD CPU 型号默认都不 未来硬件代际的 CPU 将不受 CVE-2018-3639 影响，因此应通过暴露 amd-no-ssb 告知客户机不要 它与 virt-ssbd 和 amd-ssbd 互斥。

NUMA

也可以选择在 VM 中模拟 **NUMA**[6] 架构。 NUMA 架构的基本含义是，内存不是作为所有核心都可访问的 而是分布在靠近每个插槽的本地内存区中。 由于内存总线不再是瓶颈，这可以带来速度提升。如果系统 NUMA 架构[7]，建议启用该选项，因为这允许在主机系统上合理分配 VM 资源。 在 VM 中热插拔核心或 RAM 时也需要该选项。

如果使用 NUMA 选项，建议将插槽数量设置为主机系统的节点数量。

vCPU 热插拔

现代操作系统引入了在运行中系统内热插拔 CPU，并在一定程度上热移除 CPU 的能力。 虚拟化让我们 不过，这仍是一个相当新且复杂的功能，因此应仅限于确实需要的场景。 大部分功能都可以用其他经过 资源限制。

在 Proxmox VE 中，可插入 CPU 的最大数量始终为 `cores * sockets`。 要以少于该总核心数的 CPU 启动 VM，可以使用 **vcpus** 设置； 它表示 VM 启动时应插入多少个 vCPU。

目前此功能仅在 Linux 上受支持，需要高于 3.10 的内核，建议使用高于 4.7 的内核。

可以使用如下 udev 规则，在客户机中自动将新 CPU 设置为 online：

[6] https://en.wikipedia.org/wiki/Non-uniform_memory_access

[7] 如果命令 `numactl --hardware | grep available` 返回多个节点，则主机系统具有 NUMA 架构

```
SUBSYSTEM==”cpu”, ACTION==”add”, TEST==”online”, ATTR{online}==”0”, ATTR{ ↩
   online}=”1”
```

将其保存在 /etc/udev/rules.d/ 下，并使用以 .rules 结尾的文件名。

注意：CPU 热移除依赖机器类型，并需要客户机配合。删除命令并不保证 CPU 移除实际发生；通常它
x86/amd64 上的 ACPI) 转发给客户操作系统的请求。

10.2.6 内存

对于每个 VM，可以选择设置固定大小的内存，或让 Proxmox VE 根据主机当前 RAM 使用情况动态分

固定内存分配

当将 memory 和 minimum memory 设置为相同数值时，Proxmox VE 会直接为 VM 分配指定内存。

即使使用固定内存大小，ballooning 设备也会被添加到 VM 中，因为它会提供有用信息，例如客户机
一般来说，应保持 **ballooning** 启用；但如果想要禁用它（例如用于调试），只需取消勾选 **Balloning Device** 或设置

```
balloon: 0
```

到配置中。

自动内存分配

当 minimum memory 设置为低于 memory 时，Proxmox VE 会确保指定的最低内存始终可供 VM 使用；如果主机 RAM 使用率低于某个目标百分比，则会动态向客户机添加内存，直到达到指定的最
目标百分比默认为 80%，可以 在节点选项中配置。

当主机 RAM 不足时，VM 会将部分内存释放回主机；必要时会交换正在运行的进程，并在最后手段下启
oom killer。主机和客户机之间的内存传递通过运行在客户机内部的特殊 balloon 内核驱动完成，该驱动会从主机获取或释放内存页。[8]

当多个 VM 使用自动分配功能时，可以设置 **Shares** 系数，用于表示每个 VM 应获取的空闲主机内存相
例如，假设有四个 VM，其中三个运行 HTTP 服务器，最后一个运行数据库服务器。主机配置为目标 RAM 使用率 80%。为了在有空闲 RAM 时让数据库服务器 RAM 缓存更多数据库块，希望优先考虑数据
VM。为此，可以为数据库 VM 分配 3000 的 Shares 属性，其他 VM 保持默认 Shares 设置 1000。主机服务器有 32GB RAM，当前使用 16GB，因此在已配置最低内存之外，还有 32 * 80/100 - 16 = 9GB RAM 可分配给 VM。数据库 VM 将获得 9 * 3000 / (3000 + 1000 + 1000 + 1000) = 4.5 GB 额外 RAM，每个 HTTP 服务器获得 1.5 GB。

2010 年之后发布的所有 Linux 发行版都包含 balloon 内核驱动。对于 Windows 操作系统，需要手动
balloon 驱动，并且可能导致客户机变慢，因此不建议在关键系统上使用它。

为 VM 分配 RAM 时，一个良好的经验规则是始终为主机保留 1GB 可用 RAM。

[8]这里有一篇对 balloon 驱动内部工作原理的良好说明 https://rwmj.wordpress.com/2010/07/17/virtio-balloon/

10.2.7 内存加密

AMD SEV

SEV（Secure Encrypted Virtualization）使用 AES-128 加密和 AMD Secure Processor 为每个 VM 启用内存加密。

SEV-ES（Secure Encrypted Virtualization - Encrypted State）还会加密所有 CPU 寄存器内容，以防止信息泄露给虚拟化管理程序。

SEV-SNP（Secure Encrypted Virtualization - Secure Nested Paging）还尝试防止基于软件的完整更多信息参见 AMD SEV SNP white paper。

主机要求：

- AMD EPYC CPU
- SEV-ES 仅支持 AMD EPYC 7002 系列及更新的 EPYC CPU
- SEV-SNP 仅支持 AMD EPYC 7003 系列及更新的 EPYC CPU
- SEV-SNP 要求主机内核版本为 6.11 或更高。
- 在主机 BIOS 设置中配置 AMD memory encryption
- 如果默认未启用，请向内核参数添加 ”kvm_amd.sev=1”
- 如果希望在主机上加密内存（SME），请向内核参数添加 ”mem_encrypt=on”，参见 https://ww doc/Documentation/x86/amd-memory-encryption.txt
- 可能需要增大 SWIOTLB，参见 https://github.com/AMDESE/AMDSEV#faq-4

要检查主机上是否启用了 SEV，请在 dmesg 中搜索 sev，并打印 kvm_amd 的 SEV 内核参数：

```
# dmesg | grep -i sev
[...] ccp 0000:45:00.1: sev enabled
[...] ccp 0000:45:00.1: SEV API: <buildversion>
[...] SEV supported: <number> ASIDs
[...] SEV-ES supported: <number> ASIDs
# cat /sys/module/kvm_amd/parameters/sev
Y
```

客户机要求：

- edk2-OVMF
- 建议使用 Q35
- 客户操作系统必须包含 SEV 支持。

限制：

- 由于内存已加密，主机上的内存使用量始终不准确。
- 涉及保存或还原内存的操作（如快照和在线迁移）尚不能工作，或是 可被攻击的。

- 不支持 PCI 直通。
- SEV-ES 和 SEV-SNP 都非常实验性。
- SEV-SNP 不支持 EFI 磁盘。
- 使用 SEV-SNP 时，VM 内部的 reboot 命令只会关闭 VM。

示例配置 **(SEV)**：

```
# qm set <vmid> -amd-sev type=std,no-debug=1,no-key-sharing=1,kernel-hashes ↩
   =1
```

type 定义加密技术（"type=" 不是必需的）。可用选项为 std、es 和 snp。

QEMU **policy** 参数通过 **no-debug** 和 **no-key-sharing** 参数计算。 这些参数对应 policy-bit 0 和 1。如果 **type** 为 **es**，则 policy-bit 2 设置为 1，从而启用 SEV-ES。 Policy-bit 3（nosend）始终 1，以防止迁移攻击。 有关如何计算 policy 的更多信息，请参见： AMD SEV API Specification Chapter 3

为了与较旧 OVMF 镜像以及不测量 kernel/initrd 的客户机保持向后兼容，**kernel-hashes** 选项默认 参见 https://lists.gnu.org/archive/html/qemu-devel/2021-11/msg02598.html

检查 **VM** 中 **SEV** 是否工作

方法 1 - dmesg：

输出应类似如下：

```
# dmesg | grep -i sev
AMD Memory Encryption Features active: SEV
```

方法 2 - MSR 0xc0010131 (MSR_AMD64_SEV)：

输出应为 1。

```
# apt install msr-tools
# modprobe msr
# rdmsr -a 0xc0010131
1
```

示例配置 **(SEV-SNP)**：

```
# qm set <vmid> -amd-sev type=snp,allow-smt=1,no-debug=1,kernel-hashes=1
```

allow-smt policy-bit 默认已设置。如果通过将 allow-smt 设置为 0 来禁用它， 则必须在主机上禁用 SMT，VM 才能运行。

检查 **VM** 中 **SEV-SNP** 是否工作

```
# dmesg | grep -i snp
Memory Encryption Features active: AMD SEV SEV-ES SEV-SNP
SEV: Using SNP CPUID table, 29 entries present.
SEV: SNP guest platform device initialized.
```

链接：

- https://developer.amd.com/sev/

- https://github.com/AMDESE/AMDSEV
- https://www.qemu.org/docs/master/system/i386/amd-memory-encryption.html
- https://www.amd.com/system/files/TechDocs/55766_SEV-KM_API_Specification.pdf
- https://documentation.suse.com/sles/15-SP1/html/SLES-amd-sev/index.html
- SEV Secure Nested Paging Firmware ABI Specification

10.2.8 网络设备

每个 VM 可以有多个_网络接口控制器_（NIC），共有四种不同类型：

- **Intel E1000** 是默认值，模拟 Intel 千兆网卡。
- 如果追求最高性能，应使用 **VirtIO** 半虚拟化 NIC。与所有 VirtIO 设备一样，客户操作系统应安装相
- **Realtek 8139** 模拟较旧的 100 MB/s 网卡，仅应在模拟较旧操作系统（2002 年之前发布）时使用
- **vmxnet3** 是另一种半虚拟化设备，只应在从其他虚拟化管理程序导入 VM 时使用。

Proxmox VE 会为每个 NIC 生成随机 **MAC address**，使 VM 可以在以太网网络中被寻址。

添加到 VM 的 NIC 可以采用以下两种模式之一：

- 在默认 **Bridged mode** 下，每个虚拟 NIC 在主机上由一个 *tap device*（模拟以太网 NIC 的软件回环设备）提供后端。该 tap 设备会加入一个网桥，在 Proxmox VE 中默认为 vmbr0。在此模式下，VM 可直接访问主机所在的以太网 LAN。
- 在替代的 **NAT mode** 下，每个虚拟 NIC 只会与 QEMU 用户态网络栈通信；其中内置路由器和 DHCP 服务器可以提供网络访问。该内置 DHCP 会在私有 10.0.2.0/24 范围内提供地址。NAT 模式比桥接模式慢得多，应仅用于测试。此模式只能通过 CLI 或 API 使用，不能通过 Web UI 使用。

创建 VM 时，也可以选择 **No network device** 来跳过添加网络设备。

可以覆盖每个 VM 网络设备的 **MTU** 设置。选项 `mtu=1` 表示一种特殊情况，其中 MTU 值会从底层网桥 **VirtIO** 网络设备。

多队列

如果使用 VirtIO 驱动，可以选择启用 **Multiqueue** 选项。该选项允许客户操作系统使用多个虚拟 CPU 处理网络数据包，从而增加传输的数据包总量。

在 Proxmox VE 中使用 VirtIO 驱动时，每个 NIC 网络队列都会传递给主机内核，该队列会由 vhost 驱动生成的内核线程处理。启用此选项后，可以为每个 NIC 向主机内核传递_多个_网络队列。

使用 Multiqueue 时，建议将其设置为等于客户机 vCPU 数量的值。请记住，vCPU 数量等于为 VM 配置的插槽数乘以核心数。还需要使用以下 ethtool 命令，在 VM 中为每个 VirtIO NIC 设置多用途

```
ethtool -L ens1 combined X
```

其中 X 是 VM 的 vCPU 数量。

要为 Windows 客户机配置 Multiqueue，请安装 Redhat VirtIO Ethernet Adapter drivers，然后按如
NIC 配置。打开设备管理器，右键点击 ”Network adapters” 下的 NIC，并选择 ”Properties”。然后打
”Advanced” 选项卡，在左侧列表中选择 ”Receive Side Scaling”。确保它设置为 ”Enabled”。
接着在列表中找到 ”Maximum number of RSS Queues”，并将其设置为 VM 的 vCPU 数量。
确认设置正确后，点击 ”OK” 进行确认。

请注意，将 Multiqueue 参数设置为大于 1 的值时，随着流量增加，主机和客户系统的 CPU 负载也会增
建议仅在 VM 必须处理大量传入连接时设置此选项，例如 VM 用作路由器、反向代理，或作为繁忙的
HTTP 服务器执行长轮询时。

10.2.9 显示

QEMU 可以虚拟化几种 VGA 硬件。示例如下：

- **std**，默认值，模拟带有 Bochs VBE 扩展的显卡。
- **cirrus**，曾经是默认值，它模拟一个非常旧且带有各种问题的硬件模块。此显示类型仅应在确有必要[9]，例如使用 Windows XP 或更早版本时。
- **vmware**，兼容 VMWare SVGA-II 的适配器。
- **ramfb**，qemu ramfb 适配器。
- **mdev**，由 vfio-pci 提供的 VGA 显示功能。选择它会启用 vGPU 设备显示其 framebuffer。支持 NVIDIA 和 mthreads vGPU。
- **qxl**，QXL 半虚拟化显卡。选择它还会为 VM 启用 SPICE（一种远程查看器协议）。
- **virtio-gl**，通常称为 VirGL，是供 VM 内部使用的虚拟 3D GPU；它可以将工作负载卸载到主机 GPU，而不需要特殊（昂贵）的型号和驱动，也不会完全绑定主机 GPU，从而允许在多个客户机和/

> **Note**
> VirGL 支持需要一些额外库；由于这些库相对较大，并且并非所有 GPU 型号/厂商都提供开源版本，因此默认不会安装。对于大多数设置，只需执行：`apt install libgl1 libegl1`

可以通过设置 *memory* 选项来编辑分配给虚拟 GPU 的内存量。这可以在 VM 内启用更高分辨率，尤其
SPICE/QXL 时。

由于内存由显示设备预留，为 SPICE 选择 Multi-Monitor 模式（例如用于双显示器的 `qxl2`）会带来一

- Windows 需要为每个显示器提供一个设备，因此如果 *ostype* 是某个 Windows 版本，Proxmox VE 会为 VM 的每个显示器额外提供一个设备。每个设备都会获得指定内存量。
- Linux VM 始终可以启用更多虚拟显示器，但选择 Multi-Monitor 模式会将分配给设备的内存乘以显

选择 `serialX` 作为显示 *type* 会禁用 VGA 输出，并将 Web Console 重定向到所选串口。在这种情况下
memory 设置会被忽略。

[9] https://www.kraxel.org/blog/2014/10/qemu-using-cirrus-considered-harmful/ qemu: using cirrus considered harmful

VNC 剪贴板

可以通过将 clipboard 设置为 vnc 来启用 VNC 剪贴板。

```
# qm set <vmid> -vga <displaytype>,clipboard=vnc
```

要使用剪贴板功能，必须先安装 SPICE guest tools。在基于 Debian 的发行版上， 可以通过安装 spice-vdagent 实现。对于其他操作系统，请在官方仓库中搜索它， 或参见：https://www.spice-space.org/download.html

安装 spice guest tools 后，就可以使用 VNC 剪贴板功能（例如在 noVNC 控制台面板中）。 不过，如果使用 SPICE、virtio 或 virgl，需要选择要使用的剪贴板。 这是因为当 clipboard 设置为 vnc 时，默认 **SPICE** 剪贴板会被 **VNC** 剪贴板替换。

10.2.10 USB 直通

USB 直通设备有两种不同类型：

- 主机 USB 直通
- SPICE USB 直通

主机 USB 直通通过将主机的 USB 设备提供给 VM 来工作。 这既可以通过 vendor-id 和 product-id 完成，也可以通过主机总线和端口完成。

vendor/product-id 形如 **0123:abcd**，其中 **0123** 是厂商 ID，**abcd** 是产品 ID； 这意味着同一款 USB 设备的两个实体具有相同 ID。

bus/port 形如 **1-2.3.4**，其中 **1** 是总线，**2.3.4** 是端口路径。 它表示主机的物理端口（取决于 USB 控制器的内部顺序）。

如果 VM 启动时某个设备存在于 VM 配置中，但该设备不存在于主机上，VM 仍可正常启动。 一旦该设备 VM。

Warning
使用这种 USB 直通意味着无法将 VM 在线迁移到另一台主机， 因为该硬件只在 VM 当前所在主机上可用。

第二种直通类型是 SPICE USB 直通。如果向 VM 添加一个或多个 SPICE USB 端口， 就可以从 SPICE 客户端动态地将本地 USB 设备直通给 VM。 这对于临时重定向输入设备或硬件加密狗很有用。

也可以在集群级别映射设备，使其能够与 HA 正确配合使用，检测硬件变化，并允许非 root 用户配置。详情参见 资源映射。

10.2.11 BIOS 和 UEFI

为了正确模拟计算机，QEMU 需要使用固件。在常见 PC 上，这通常称为 BIOS 或 (U)EFI， 并在 VM 启动初期执行。它负责执行基本硬件初始化，并为操作系统提供访问固件和硬件的接口。 默认情况使用 **SeaBIOS** 完成此任务；SeaBIOS 是一种开源 x86 BIOS 实现。 对于大多数标准设置，SeaBIOS 是不错的选择。

某些操作系统（例如 Windows 11）可能要求使用兼容 UEFI 的实现。 在这种情况下，必须改用 **OVMF**，它是开源 UEFI 实现。[10]

对于 arm64/riscv64/loongarch64 设备，必须使用 **OVMF** 启动操作系统。

还有其他场景中 SeaBIOS 可能不是理想的启动固件，例如想要进行 VGA 直通时。[11]

如果想要使用 OVMF，需要考虑几件事：

为了保存 **boot order** 等内容，需要有一个 EFI Disk。 该磁盘会包含在备份和快照中，并且只能有一个

可以使用以下命令创建这样的磁盘：

```
# qm set <vmid> -efidisk0 <storage>:1,format=<format>,efitype=4m,pre- ↩
   enrolled-keys=1
```

其中 **<storage>** 是要放置磁盘的存储，**<format>** 是该存储支持的格式。 也可以通过 Web 界面，在 VM 硬件部分使用 *Add → EFI Disk* 创建此类磁盘。

efitype 选项指定应使用哪个版本的 OVMF 固件。对于新 VM，应始终使用 *4m*， 因为它支持 Secure Boot，并分配了更多空间以支持未来发展（这也是 GUI 中的默认值）。

pre-enroll-keys 指定 efidisk 是否应预加载发行版专用和 Microsoft Standard Secure Boot 密钥。 它还会默认启用 Secure Boot（不过仍可在 VM 内的 OVMF 菜单中禁用）。

Note
如果想在现有 VM 中开始使用 Secure Boot（该 VM 仍使用 *2m* efidisk），需要重新创建 efidisk。 为此，请删除旧磁盘（`qm set <vmid> -delete efidisk0`），并按上文所述添加新磁盘。 这会重置你在 OVMF 菜单中进行的所有自定义配置！

使用 OVMF 和虚拟显示（不使用 VGA 直通）时，需要在 OVMF 菜单中设置客户端分辨率（启动期间按 ESC 键可进入该菜单），或者必须选择 SPICE 作为显示类型。

10.2.12 可信平台模块 (TPM)

Trusted Platform Module 是一种安全存储密钥等秘密数据的设备，并提供用于验证系统启动的防

某些操作系统（例如 Windows 11）要求机器（无论物理机还是虚拟机）附加此类设备。

通过指定 **tpmstate** 卷来添加 TPM。其工作方式类似 efidisk，一旦创建就不能更改（只能移除）。可以通过以下命令添加：

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

其中 **<storage>** 是要存放状态的存储，**<version>** 为 *v1.2* 或 *v2.0*。 也可以通过 Web 界面，在 VM 硬件部分选择 *Add → TPM State* 添加。

v2.0 TPM 规范更新且支持更好，因此除非有特定实现要求 *v1.2* TPM，否则应优先使用它。

Note
与物理 TPM 相比，模拟 TPM *不会*提供任何真正的安全收益。 TPM 的意义在于其上的数据不能被轻易修改，除非通过 TPM 规范指定的命令。 而对于模拟设备，数据存储在普通卷上，因此任何有访问权限的人都可能编辑它。

---

[10] 参见 OVMF 项目 https://github.com/tianocore/tianocore.github.io/wiki/OVMF

[11] Alex Williamson 有一篇关于此主题的优秀博客文章 https://vfio.blogspot.co.at/2014/08/primary-graphics-assignment-without-vga.html

10.2.13 VM 间共享内存

可以添加 VM 间共享内存设备（ivshmem），它允许在主机和客户机之间共享内存，也允许在多个客户

可以使用 qm 添加此类设备：

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

其中大小单位为 MiB。文件将位于 /dev/shm/pve-shm-$name 下（默认名称为 vmid）。

> **Note**
> 目前，一旦任何使用该设备的 VM 关闭或停止，该设备就会被删除。已打开的连接仍会保留，但无法再建立到同一设备的新连接。

此类设备的一个用例是 Looking Glass [12] 项目，它支持主机与客户机之间的高性能、低延迟显示镜像。

10.2.14 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，模拟 ICH9
- intel-hda：Intel HD Audio Controller，模拟 ICH6
- AC97：Audio Codec '97，适用于 Windows XP 等较旧操作系统

有两个可用后端：

- *spice*
- *none*

spice 后端可以与 SPICE 结合使用；如果某些软件需要 VM 中存在音频设备才能工作，*none* 后端也可（参见 PCI 直通 和 USB 直通）。 Microsoft RDP 等远程协议也提供播放声音的选项。

10.2.15 VirtIO RNG

RNG（Random Number Generator）是一种向系统提供熵（*randomness*）的设备。 虚拟硬件 RNG 可用于从主机系统向客户 VM 提供这种熵。 这有助于避免客户机中的熵耗尽问题（即可用熵不足

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]
```

[12] Looking Glass: https://looking-glass.io/

source 指定在主机上从何处读取熵，必须是以下之一：

- /dev/urandom：非阻塞内核熵池（首选）
- /dev/random：阻塞内核池（不推荐，可能导致主机系统熵耗尽）
- /dev/hwrng：直通附加到主机的硬件 RNG（如果有多个可用，将使用 /sys/devices/virtual/
  中选择的一个）

可以通过 max_bytes 和 period 参数指定限制，它们表示每 period 毫秒读取 max_bytes。
不过，这并不代表线性关系：1024B/1000ms 表示最多 1 KiB 数据会按 1 秒定时器变为可用，
而不是在一秒内向客户机流式传输 1 KiB。因此，减少 period 可用于以更快速度向客户机注入熵。

默认情况下，限制设置为每 1000 ms 1024 字节（1 KiB/s）。 建议始终使用限制器，以避免客户机使
max_bytes 设置为 *0* 以禁用所有限制。

10.2.16 Virtiofs

Virtiofs 是为虚拟环境设计的共享文件系统。它允许通过在 VM 内挂载主机上可用的目录树来共享该目录
它不使用网络栈，目标是提供与源文件系统相近的性能和语义。

要使用 virtiofs，需要在后台运行 virtiofsd 守护进程。 在 Proxmox VE 中，启动使用 virtiofs
挂载的 VM 时会自动完成此操作。

使用 kernel >=5.4 的 Linux VM 默认支持 virtiofs（virtiofs kernel module），但某些功能需要更新

要使用 virtiofs，请确保 Proxmox VE 主机上已安装 virtiofsd：

```
apt install virtiofsd
```

有一份 guide 说明如何在 Windows VM 中使用 virtiofs。

已知限制

- 如果 virtiofsd 崩溃，其挂载点会在 VM 中挂起，直到 VM 完全停止。
- virtiofsd 无响应可能导致 VM 中的挂载挂起，类似不可达的 NFS。
- 内存热插拔不能与 virtiofs 组合使用（也会导致访问挂起）。
- 与内存相关的功能，例如在线迁移、快照和休眠，不适用于 virtiofs 设备。
- Windows 无法理解 virtiofs 上下文中的 ACL。因此，不要为 Windows VM 暴露 ACL， 否则
  virtiofs 设备在 VM 内不可见。

为共享目录添加映射

要为共享目录添加映射，可以按 资源映射一节所述，直接使用 pvesh 调用 API：

```
pvesh create /cluster/mapping/dir --id dir1 \
    --map node=node1,path=/path/to/share1 \
    --map node=node2,path=/path/to/share2 \
```

向 **VM** 添加 **virtiofs**

要使用 virtiofs 共享目录，请向 VM 配置添加参数 `virtiofs<N>`（N 可以是 0 到 9 之间的任意值），并使用已在资源映射中配置的目录 ID（dirid）。此外，可以根据需求将 cache 选项设置为 `always`、`n` 或 `auto`（默认值：`auto`）。不同缓存模式的行为可在 这里的 "Caching Modes" 章节阅读。

`virtiofsd` 支持 ACL 和 xattr 直通（可通过 `expose-acl` 和 `expose-xattr` 选项启用），如果底层主机文件系统支持它们，则允许客户机访问 ACL 和 xattr；但它们也必须与客户文件系统兼容（例如，大多数 Linux 文件系统支持 ACL，而 Windows 文件系统不支持）。

`expose-acl` 选项会自动隐含 `expose-xattr`，也就是说，如果 `expose-acl` 设置为 `1`， 再将 `expose-xattr` 设置为 `0` 不会产生差异。

如果希望 virtiofs 遵守 `O_DIRECT` 标志，可以将 `direct-io` 参数设置为 `1`（默认值：`0`）。 这会降低性

```
qm set <vmid> -virtiofs0 dirid=<dirid>,cache=always,direct-io=1
qm set <vmid> -virtiofs1 <dirid>,cache=never,expose-xattr=1
qm set <vmid> -virtiofs2 <dirid>,expose-acl=1
```

要在带有 Linux 内核 virtiofs 驱动的客户 VM 中临时挂载 virtiofs，请在客户机内运行以下命令：

```
mount -t virtiofs <dirid> <mount point>
```

要持久挂载 virtiofs，可以创建 fstab 条目：

```
<dirid> <mount point> virtiofs rw,relatime 0 0
```

与当前节点路径关联的 dirid 也会用作挂载标签（客户机上用于挂载设备的名称）。

有关可用 virtiofsd 参数的更多信息，请参见 GitLab virtiofsd 项目页面。

10.2.17 设备启动顺序

QEMU 可以告知客户机应从哪些设备启动以及启动顺序。 这可以通过配置中的 `boot` 属性指定，例如：

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

这样，客户机会首先尝试从磁盘 `scsi0` 启动；如果失败，则继续尝试从 `net0` 网络启动； 如果也失败，PCIe 设备启动（如果是 NVMe，则视为磁盘；否则尝试进入 option ROM）。

在 GUI 中，可以使用拖放编辑器指定启动顺序，并使用复选框整体启用或禁用某些设备的启动能力。

Note
如果客户机使用多个磁盘启动操作系统或加载引导加载器，则所有这些磁盘都必须标记为 *bootable* （即必须启用复选框，或出现在配置列表中），客户机才能启动。 这是因为较新的 SeaBIOS 和 OVMF 版本只会初始化标记为 *bootable* 的磁盘。

无论如何，即使设备未出现在列表中或复选框被禁用，一旦客户机操作系统启动并初始化它们， 这些设标志只影响客户机 BIOS 和引导加载器。

10.2.18 虚拟机自动启动和关闭

创建 VM 后，可能希望它们在主机系统启动时自动启动。 为此，需要在 Web 界面中 VM 的 *Options* 选项卡选择 *Start at boot*，或使用以下命令设置：

```
# qm set <vmid> -onboot 1
```

启动和关闭顺序

在某些情况下，需要精细调整 VM 的启动顺序，例如某个 VM 为其他客户系统提供防火墙或 DHCP。为此，可以使用以下参数：

- **Start/Shutdown order**：定义启动顺序优先级。例如，如果希望 VM 第一个启动，请将其设置为 1。（关闭时使用相反的启动顺序，因此启动顺序为 1 的机器会最后关闭。）如果一台主机上的多个 VM 定义了相同顺序，它们还会按 *VMID* 升序排序。
- **Startup delay**：定义此 VM 启动与后续 VM 启动之间的间隔。例如，如果希望在启动其他 VM 前等待 240 秒，请将其设置为 240。
- **Shutdown timeout**：定义 Proxmox VE 在发出关闭命令后等待 VM 离线的秒数。 默认值为 180，这意味着 Proxmox VE 会发出关闭请求，并等待机器离线 180 秒。 如果超时后机器仍在线，将

Note
目前，由 HA 栈管理的 VM 不遵循 *start on boot* 和 *boot order* 选项。 这些 VM 会被启动和关闭算法跳过，因为 HA 管理器本身会确保 VM 启动和停止。

请注意，未设置 Start/Shutdown order 参数的机器始终会在已设置该参数的机器之后启动。 此外，该

如果需要在主机启动和第一个 VM 启动之间设置延迟，请参见 Proxmox VE 节点管理章节。

10.2.19 QEMU Guest Agent

QEMU Guest Agent 是运行在 VM 内部的服务，在主机和客户机之间提供通信通道。 它用于交换信息

例如，VM 摘要面板中的 IP 地址就是通过 guest agent 获取的。

又如，在启动备份时，会通过 guest agent 告知客户机使用 *fs-freeze* 和 *fs-thaw* 命令同步未完成写入

要让 guest agent 正常工作，必须执行以下步骤：

- 在客户机中安装 agent，并确保它正在运行
- 在 Proxmox VE 中启用通过 agent 的通信

安装 **Guest Agent**

对于大多数 Linux 发行版，都可以使用 guest agent。软件包通常名为 `qemu-guest-agent`。

对于 Windows，可以从 Fedora VirtIO 驱动 ISO 安装。

启用 **Guest Agent** 通信

可以在 VM 的 **Options** 面板中启用 Proxmox VE 与 guest agent 的通信。 需要重新启动 VM，变更才会生效。

使用 **QGA** 自动 **TRIM**

可以启用 *Run guest-trim* 选项。启用后，在以下可能向存储写出零的操作之后， Proxmox VE 会向客户机发出 trim 命令：

- 将磁盘移动到另一个存储
- 将带本地存储的 VM 在线迁移到另一个节点

在精简配置存储上，这有助于释放未使用空间。

Note
Linux 上的 ext4 有一个注意事项，因为它使用内存中优化来避免发出重复的 TRIM 请求。 由于客户机不知道底层存储变化，只有第一次 guest-trim 会按预期运行。 在下次重启之前，后续操作只会考虑此后发生变化的文件系统部分。

备份时文件系统 **Freeze** 和 **Thaw**

默认情况下，执行备份时会通过 *fs-freeze* QEMU Guest Agent 命令同步客户文件系统，以提供一致性

在 Windows 客户机上，某些应用程序可能通过接入 Windows VSS（Volume Shadow Copy Service）层自行处理一致性备份； 此时 *fs-freeze* 可能会干扰它们。例如，已观察到在某些 SQL Server 上调用 *fs-freeze* 会触发 VSS 以破坏 SQL Server 差异备份链的模式调用 SQL Writer VSS 模块。

处理这种情况有两个选项。

1. 配置 QEMU Guest Agent 使用不会干扰其他 VSS 用户的不同 VSS 变体。 Proxmox VE wiki 中有更多详情。
2. 或者，可以通过将 `freeze-fs-on-backup` QGA 选项设置为 `0`， 配置 Proxmox VE 在备份时不 freeze-and-thaw 周期。 也可以通过 GUI 中的 *Freeze/thaw guest filesystems on backup for consistency* 选项完成。

Important
禁用此选项可能导致备份中的文件系统不一致。因此，首选方式是在客户机中调整 QEMU Guest Agent 配置。

故障排查

VM 无法关闭

确保 guest agent 已安装并正在运行。

一旦启用 guest agent，Proxmox VE 会通过 guest agent 发送 *shutdown* 等电源命令。 如果 guest agent 未运行，命令无法正确执行，关闭命令会超时。

10.2.20 SPICE 增强

SPICE 增强是可选功能，可改善远程查看器体验。

要通过 GUI 启用它们，请进入虚拟机的 **Options** 面板。 要通过 CLI 启用它们，请运行以下命令：

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

Note
要使用这些功能，虚拟机的 Display 必须设置为 SPICE (qxl)。

文件夹共享

与客户机共享本地文件夹。客户机中需要安装 `spice-webdavd` 守护进程。 它会通过位于 http://local 的本地 WebDAV 服务器提供共享文件夹。

对于 Windows 客户机，可以从 官方 SPICE 网站 下载 *Spice WebDAV daemon* 安装程序。

大多数 Linux 发行版都有名为 `spice-webdavd` 的可安装软件包。

要在 Virt-Viewer（Remote Viewer）中共享文件夹，请转到 *File → Preferences*。 选择要共享的文件

Note
文件夹共享目前仅适用于 Linux 版本的 Virt-Viewer。

Caution
实验性功能！目前此功能工作并不可靠。

视频流

快速刷新的区域会被编码为视频流。有两个选项：

- **all**：任何快速刷新的区域都会被编码为视频流。
- **filter**：使用额外过滤器决定是否应使用视频流（目前只跳过小窗口表面）。

无法给出是否应启用视频流以及应选择哪个选项的一般性建议。实际效果会因具体情况而异。

故障排查

共享文件夹未显示

确保客户机中的 WebDAV 服务已启用并正在运行。在 Windows 上，它名为 *Spice webdav proxy*。在 Linux 中，名称为 *spice-webdavd*，但可能因发行版而异。

如果服务正在运行，请在客户机浏览器中打开 http://localhost:9843 来检查 WebDAV 服务器。

重启 SPICE 会话可能会有所帮助。

10.3 迁移

如果有集群，可以使用以下命令将 VM 迁移到另一台主机：

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

通常有两种机制：

- 在线迁移（也称 Live Migration）
- 离线迁移

10.3.1 在线迁移

如果 VM 正在运行且未配置本地绑定资源（例如直通设备），可以在 `qm migration` 命令调用中使用 `--online` 标志发起在线迁移。VM 运行时，Web 界面默认使用在线迁移。

工作原理

在线迁移首先在目标主机上以 *incoming* 标志启动新的 QEMU 进程。 该进程只执行基本初始化，客户 vCPU 仍保持暂停，然后等待源虚拟机的客户内存和设备状态数据流。 所有其他资源（例如磁盘）要么 VM 运行时状态迁移开始前发送完成； 因此剩下需要传输的只有内存内容和设备状态。

建立此连接后，源端开始异步向目标端发送内存内容。如果源端的客户内存发生变化， 这些区域会被标 dirty，并再次发送客户内存数据。 该循环会重复执行，直到正在运行的源 VM 与 incoming 目标 VM 之间的数据差异足够小， 可以在几毫秒内发送完成；此时源 VM 可以完全暂停，且用户或程序不会 随后将剩余数据发送到目标端，并恢复目标 VM 的 CPU，使其在远低于一秒的时间内成为新的运行 VM。

要求

要使 Live Migration 正常工作，需要满足以下条件：

- VM 没有无法迁移的本地资源。例如，当前直通的 PCI 或 USB 设备会阻止在线迁移。 另一方面，本地
- 主机位于同一个 Proxmox VE 集群中。

- 主机之间具有可用且可靠的网络连接。
- 目标主机必须具有相同或更高版本的 Proxmox VE 软件包。 虽然有时反向也能工作，但无法保证。
- 主机 CPU 来自同一厂商，并具有相似能力。不同厂商的 CPU 根据实际型号和配置的 VM CPU 类型 可能 可以工作，但无法保证，因此在将此类设置部署到生产环境之前请先测试。

10.3.2 离线迁移

如果有本地资源，只要所有磁盘都位于两台主机都定义的存储上，仍可离线迁移 VM。 迁移随后会像在
请注意，任何硬件直通配置都可能需要适配目标主机上的设备位置。

10.4 复制和克隆

VM 安装通常使用操作系统厂商提供的安装介质（CD-ROM）完成。 根据操作系统不同，这可能是一个

部署大量同类型 VM 的一种简单方法是复制现有 VM。 我们使用术语 *clone* 表示这类副本，并区分 *linked* 克隆和 *full* 克隆。

完整克隆
: 这种复制的结果是一个独立 VM。新 VM 不与原始 VM 共享任何存储资源。

  可以选择 **Target Storage**，因此可利用它将 VM 迁移到完全不同的存储。 如果存储驱动支持多 **Format**。

  > **Note**
  > 完整克隆需要读取并复制所有 VM 镜像数据。这通常比创建链接克隆慢得多。

  某些存储类型允许复制特定 **Snapshot**，默认是 *current* VM 数据。 这也意味着最终副本不会包 VM 的任何额外快照。

链接克隆
: 现代存储驱动支持生成快速链接克隆的方式。这种克隆是一个可写副本，其初始内容与原始数据相
  创建链接克隆几乎是瞬时完成的，并且最初不消耗额外空间。

  它们称为 *linked*，是因为新镜像仍引用原始镜像。未修改的数据块从原始镜像读取， 而修改会写
  Copy-on-write。

  这要求原始卷为只读。在 Proxmox VE 中，可以将任何 VM 转换为只读 模板。 之后可使用此类模

  > **Note**
  > 当链接克隆存在时，不能删除原始模板。

  无法更改链接克隆的 **Target storage**，因为这是存储内部功能。

Target node 选项允许在不同节点上创建新 VM。唯一限制是 VM 位于共享存储上， 并且该存储在目

为避免资源冲突，所有网络接口 MAC 地址都会随机化，并且会为 VM BIOS（smbios1）设置生成新的 *UUID*。

10.5 虚拟机模板

可以将 VM 转换为模板。此类模板是只读的，可用于创建链接克隆。

PXVIRT 有两种模板：pxvditemplate 和 template。Pxvditemplate 类型可以启动，常用于 VDI 场景。

Note
不能启动模板，因为这会修改磁盘镜像。如果想更改模板，请创建链接克隆并修改该克隆。

10.6 VM Generation ID

Proxmox VE 支持虚拟机 Generation ID（*vmgenid*）[13]。客户操作系统可以使用它检测导致时间偏移

创建新 VM 时，会自动生成 *vmgenid* 并保存到其配置文件中。

要为已存在 VM 创建并添加 *vmgenid*，可以传递特殊值 '1' 让 Proxmox VE 自动生成，也可以手动设置 *UUID* [14] 作为其值，例如：

```
# qm set VMID -vmgenid 1
# qm set VMID -vmgenid 00000000-0000-0000-0000-000000000000
```

Note
首次向现有 VM 添加 *vmgenid* 设备，可能产生与快照回滚、备份还原等变更相同的效果，因为 VM 可能将其解释为代际变化。

在少数不需要 *vmgenid* 机制的情况下，可以在创建 VM 时为其值传递 '0'，或事后使用以下命令从配置

```
# qm set VMID -delete vmgenid
```

vmgenid 最突出的用例是较新的 Microsoft Windows 操作系统；它们使用该机制避免在快照回滚、备份还原或整个 VM 克隆操作中，对时间敏感或复制服务（例如数据库或域控制器 [15]）产生问题。

10.7 导入虚拟机

可以通过多种方法从外部虚拟化管理程序或其他 Proxmox VE 集群导入现有虚拟机，最常见的方法包括

- 使用原生导入向导，它利用 *import* 内容类型，例如由 ESXi 特殊存储提供的内容类型。
- 在源端执行备份，然后在目标端还原。该方法在从另一个 Proxmox VE 实例迁移时效果最好。
- 使用 qm 命令行工具中专用于 OVF 的导入命令。

如果要从其他虚拟化管理程序将 VM 导入 Proxmox VE，建议先熟悉 Proxmox VE 的概念。

[13] 官方 *vmgenid* 规范 https://docs.microsoft.com/en-us/windows/desktop/hyperv_v2/virtual-machine-generation-identifier

[14] 在线 GUID 生成器 http://guid.one/

[15] https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/-virtualized-domain-controller-architecture

10.7.1 导入向导

Proxmox VE 提供了一个集成式 VM 导入器，使用存储插件系统原生集成到 API 和基于 Web 的用户界面
可以使用它整体导入 VM，并将大部分配置映射到 Proxmox VE 配置模型，同时减少停机时间。

Note
导入向导是在 Proxmox VE 8.2 开发周期中添加的，目前处于技术预览状态。
虽然它已经很有前景且工作稳定，但仍在积极开发中。

要使用导入向导，必须先为导入源设置一个新存储；可以在 Web 界面中通过 *Datacenter → Storage → Add* 完成。

然后，可以在资源树中选择新存储，并使用 *Virtual Guests* 内容选项卡查看所有可导入的客户机。

选择其中一个，并使用 *Import* 按钮（或双击）打开导入向导。 可以在此修改可用选项的子集，然后启
请注意，导入完成后还可以进行更高级的修改。

Tip
ESXi 导入向导已经使用 ESXi 6.5 到 8.0 版本测试。 请注意，使用 vSAN 存储的客户机不能直接导入；必须先将其磁盘移动到其他存储。 虽然可以使用 vCenter 作为导入源，但性能会显著下降（慢 5 到 10 倍）。

关于如何让虚拟客户机适配新的虚拟化管理程序的分步指南和提示，请参见我们的 迁移到 Proxmox VE wiki 文章。

OVA/OVF 导入

要导入 OVA/OVF 文件，首先需要一个具有 *import* 内容类型的基于文件的存储。 在该存储上，会有一个
import 文件夹，可将 OVA 文件或带有相应镜像且采用扁平结构的 OVF 文件放入其中。 也可以使用
Web UI 直接上传或下载 OVA 文件。随后可以使用 Web UI 选择这些文件， 并使用导入向导导入客户机

对于 OVA 文件，需要额外空间临时提取镜像。这需要一个配置了 *images* 内容类型的基于文件的存储。
默认选择源存储用于此目的，但也可以指定一个 *Import Working Storage*， 在导入到实际目标存储之

Note
由于 OVA/OVF 文件结构和内容并不总是维护良好或定义清晰，可能需要手动调整某些客户机设置。
例如，SCSI 控制器类型几乎从不会在 OVA/OVF 文件中定义，但默认值无法使用 OVMF (UEFI)
启动， 因此在这些情况下应选择 *Virtio SCSI* 或 *VMware PVSCSI*。

10.7.2 通过 CLI 导入 OVF/OVA

来自外部虚拟化管理程序的 VM 导出通常采用一个或多个磁盘镜像的形式，并附带描述 VM 设置
（RAM、核心数量）的配置文件。 + 如果磁盘来自 VMware 或 VirtualBox，磁盘镜像可以是 vmdk
格式；如果磁盘来自 KVM 虚拟化管理程序， 则可以是 qcow2 格式。VM 导出最流行的配置格式是
OVF 标准，但实际互操作性有限， 因为许多设置未在标准本身中实现，虚拟化管理程序会在非标准扩展

除格式问题外，从其他虚拟化管理程序导入磁盘镜像时，如果不同虚拟化管理程序之间模拟硬件变化过
也可能失败。Windows VM 尤其受此影响，因为该操作系统对任何硬件变化都非常敏感。 可以在导出前
MergeIDE.zip 实用工具，并在启动导入的 Windows VM 前选择 **IDE** 硬盘类型，以解决此问题。

最后还有半虚拟化驱动问题；这些驱动可提升模拟系统速度，并且特定于虚拟化管理程序。 GNU/Linux
和其他自由 Unix 操作系统默认已安装所有必要驱动，因此可以在导入 VM 后立即切换到半虚拟化驱动。
对于 Windows VM，需要自行安装 Windows 半虚拟化驱动。

GNU/Linux 和其他自由 Unix 通常可以顺利导入。请注意，由于上述问题，我们无法保证所有情况下都
Windows VM。

Windows OVF 导入分步示例

Microsoft 提供 Virtual Machines downloads 以便开始 Windows 开发。我们将使用其中一个来演示
OVF 导入功能。

了解用户协议后，为 VMware 平台选择 *Windows 10 Enterprise (Evaluation - Build)*，并下载
zip。

使用 `unzip` 工具或任意归档工具解压 zip，并通过 ssh/scp 将 ovf 和 vmdk 文件复制到 Prox-
mox VE 主机。

这会创建一个新虚拟机，使用从 OVF manifest 中读取的核心数、内存和 VM 名称， 并将磁盘导入到
`local-lvm` 存储。必须手动配置网络。

```
# qm importovf 999 WinDev1709Eval.ovf local-lvm
```

VM 已准备好启动。

也可以向 VM 添加现有磁盘镜像，该镜像可以来自外部虚拟化管理程序，也可以是自行创建的镜像。

假设使用 *vmdebootstrap* 工具创建了 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
```

现在可以创建新的目标 VM，将镜像导入到存储 `pvedir`，并将其附加到 VM 的 SCSI 控制器：

```
# qm create 600 --net0 virtio,bridge=vmbr0 --name vm600 --serial0 socket \
   --boot order=scsi0 --scsihw virtio-scsi-pci --ostype l26 \
   --scsi0 pvedir:0,import-from=/path/to/dir/vm600.raw
```

VM 已准备好启动。

10.8 Cloud-Init 支持

Cloud-Init 是事实上的跨发行版软件包，用于处理 虚拟机实例的早期初始化。借助 Cloud-Init，可以在 ssh 密钥。当 VM 首次启动时，VM 内部的 Cloud-Init 软件会应用这些设置。

许多 Linux 发行版都提供可直接使用的 Cloud-Init 镜像，大多面向 *OpenStack* 设计。 这些镜像同样可 Proxmox VE 中使用。虽然获取这类即用镜像看起来很方便，但通常建议自行 准备镜像。这样做的优势

创建好 Cloud-Init 镜像后，建议将其转换为 VM 模板。通过 VM 模板可以快速创建链接克隆，因此这是快速部署新 VM 实例的方法。启动新 VM 前，只需配置网络（以及可能需要的 ssh 密钥）。

建议使用基于 SSH 密钥的认证登录由 Cloud-Init 置备的 VM。也可以设置密码，但这不如 基于 SSH 密钥的认证安全，因为 Proxmox VE 需要在 Cloud-Init 数据中保存该密码的加密版本。

Proxmox VE 会生成一个 ISO 镜像，用于将 Cloud-Init 数据传递给 VM。因此，所有 Cloud-Init VM 都需要分配一个 CD-ROM 驱动器。通常还应添加串口控制台并将其用作显示设备。许多 Cloud-Init 镜像依赖这一点，这也是 OpenStack 的要求。不过，其他镜像可能不适用于这种 配置。如

10.8.1 准备 Cloud-Init 模板

第一步是准备 VM。基本上可以使用任意 VM。只需在要准备的 VM *内部*安装 Cloud-Init 软件包即可。 Debian/Ubuntu 的系统中，操作如下：

```
apt-get install cloud-init
```

Warning
该命令*不应*在 Proxmox VE 主机上执行，只应在 VM 内部执行。

许多发行版已经提供可直接使用的 Cloud-Init 镜像（以 `.qcow2` 文件形式提供），因此也可以 直接下载 Ubuntu 在 https://cloud-images.ubuntu.com 提供的 cloud image。

```
# download the image
wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg- ↩
   amd64.img

# create a new VM with VirtIO SCSI controller
qm create 9000 --memory 2048 --net0 virtio,bridge=vmbr0 --scsihw virtio- ↩
   scsi-pci

# import the downloaded disk to the local-lvm storage, attaching it as a  ↩
   SCSI drive
qm set 9000 --scsi0 local-lvm:0,import-from=/path/to/bionic-server-cloudimg ↩
   -amd64.img
```

Note
Ubuntu Cloud-Init 镜像要求 SCSI 驱动器使用 `virtio-scsi-pci` 控制器类型。

添加 **Cloud-Init CD-ROM** 驱动器

下一步是配置 CD-ROM 驱动器，它将用于向 VM 传递 Cloud-Init 数据。

```
qm set 9000 --ide2 local-lvm:cloudinit
```

为能够直接从 Cloud-Init 镜像启动，将 boot 参数设置为 order=scsi0，限制 BIOS 仅从此磁盘启动。VM BIOS 会跳过对可引导 CD-ROM 的检测。

```
qm set 9000 --boot order=scsi0
```

许多 Cloud-Init 镜像要求配置串口控制台并将其用作显示设备。但如果某个镜像无法使用该配置，请改回默认显示配置。

```
qm set 9000 --serial0 socket --vga serial0
```

最后一步，将 VM 转换为模板会很有帮助。之后可以从该模板快速创建链接克隆。从 VM 模板部署比创建完整克隆（副本）快得多。

```
qm template 9000
```

10.8.2 部署 Cloud-Init 模板

可以通过克隆轻松部署此类模板：

```
qm clone 9000 123 --name ubuntu2
```

然后配置用于认证的 SSH 公钥，并配置 IP 设置：

```
qm set 123 --sshkey ~/.ssh/id_rsa.pub
qm set 123 --ipconfig0 ip=10.0.10.123/24,gw=10.0.10.1
```

也可以只用一条命令配置所有 Cloud-Init 选项。上面的示例只是为了缩短行长度而拆分为多条 命令。还 IP 设置。

10.8.3 自定义 Cloud-Init 配置

Cloud-Init 集成也允许使用自定义配置文件替代自动生成的配置。这可通过命令行中的 cicustom 选项完成：

```
qm set 9000 --cicustom ”user=<volume>,network=<volume>,meta=<volume>”
```

自定义配置文件必须位于支持 snippets 的存储上，并且必须在 VM 可能迁移到的所有节点上可用。否则 VM 将无法启动。例如：

```
qm set 9000 --cicustom ”user=local:snippets/userconfig.yaml”
```

Cloud-Init 有三类配置。第一类是上例中的 user 配置，第二类是 network 配置，第三类是 meta 配置。它们可以一起指定，也可以按需组合。未指定自定义配置文件的部分，将使用自动 生成的配置。

可以导出生成的配置，作为自定义配置的基础：

```
qm cloudinit dump 9000 user
```

network 和 meta 也支持同样的命令。

常用命令示例

在修改自定义配置前，可以先查看 Proxmox VE 当前生成的 Cloud-Init 内容：

```
qm cloudinit dump 9000 user
qm cloudinit dump 9000 network
qm cloudinit dump 9000 meta
```

10.8.4 Windows 上的 Cloud-Init

Windows 有一个 Cloud-Init 的重新实现，名为 cloudbase-init。 Cloudbase-Init 并不具备 Cloud-Init 的所有功能，部分功能也与 Cloud-Init 有所不同。

Cloudbase-Init 要求将 ostype 设置为任意 Windows 版本，并将 citype 设置为 configdrive2；对 Windows ostype，后者都是默认值。

目前没有免费的 Windows 现成 cloud image。使用 Cloudbase-Init 需要手动安装并配置 Windows 客户机。

10.8.5 准备 Cloudbase-Init 模板

第一步是在 VM 中安装 Windows。然后在客户机中下载并安装 Cloudbase-Init。可能需要安装 Beta 版本。安装结束时不要运行 Sysprep，而应先配置 Cloudbase-Init。

常见需要设置的选项包括：

- *username*: 设置管理员用户名
- *groups*: 允许将用户添加到 Administrators 组
- *inject_user_password*: 将其设置为 true，以允许在 VM 配置中设置密码
- *first_logon_behaviour*: 将其设置为 no，登录时就不会要求设置新密码
- *rename_admin_user*: 将其设置为 true，以允许将默认 Administrator 用户重命名为 usernam 指定的用户名
- *metadata_services*: 将其设置为 cloudbaseinit.metadata.services.configdrive.Conf Cloudbase-Init 优先检查此服务。否则，Cloudbase-Init 在启动后可能需要几分钟才能完成 系统配

某些插件（例如 SetHostnamePlugin）需要重启，并会自动执行重启。要禁用 Cloudbase-Init 自动重启，可以将 allow_reboot 设置为 false。

完整配置选项可在 官方 cloudbase-init 文档 中查看。

配置完成后创建一个快照通常很有意义，以便在部分配置仍需调整时回退。配置 Cloudbase-Init 之后，就可以开始创建模板。关闭 Windows 客户机，添加 Cloud-Init 磁盘，并将其转换为模板。

```
qm set 9000 --ide2 local-lvm:cloudinit
qm template 9000
```

将模板克隆为新的 VM：

```
qm clone 9000 123 --name windows123
```

然后设置密码、网络配置和 SSH 密钥：

```
qm set 123 --cipassword <password>
qm set 123 --ipconfig0 ip=10.0.10.123/24,gw=10.0.10.1
qm set 123 --sshkey ~/.ssh/id_rsa.pub
```

设置密码前，请确保 ostype 已设置为任意 Windows 版本。否则密码会被加密，而 Cloudbase-Init 会把加密后的密码当作明文密码使用。

全部设置完成后，启动克隆出的客户机。首次启动时登录不会成功，系统会因主机名变更而自动 重启。

10.8.6 Cloudbase-Init 与 Sysprep

Sysprep 是用于重置 Windows 配置并提供一个 new 系统的功能。它可以与 Cloudbase-Init 配合使用

使用 Sysprep 时，需要调整 2 个配置文件。第一个是常规配置文件，第二个是以 -unattend.conf 结尾的文件。

Cloudbase-Init 分 2 步运行：首先是使用 -unattend.conf 的 Sysprep 阶段，然后是使用主配置文件的常规阶段。

对于 Windows Server，使用随附的 Unattend.xml 文件运行 Sysprep 应该可以直接工作。普通 Windows 版本则需要额外步骤：

1. 打开一个 PowerShell 实例
2. 启用 Administrator 用户：

   ```
   net user Administrator /active:yes`
   ```

3. 使用 Administrator 用户安装 Cloudbase-Init
4. 修改 Unattend.xml，加入在 sysprep 后首次启动时启用 Administrator 用户的命令：

   ```
   <RunSynchronousCommand wcm:action="add">
     <Path>net user administrator /active:yes</Path>
     <Order>1</Order>
     <Description>Enable Administrator User</Description>
   </RunSynchronousCommand>
   ```

   确保 <Order> 不与其他同步命令冲突。将 Cloudbase-Init 命令的 <Order> 修改为更大的值，使其在该命令之后运行：<Order>2</Order>

5. （仅 Windows 11）移除冲突的 Microsoft.OneDriveSync 软件包：

   ```
   Get-AppxPackage -AllUsers Microsoft.OneDriveSync | Remove-AppxPackage ↩
      -AllUsers
   ```

6. cd 进入 Cloudbase-Init 配置目录：

   ```
   cd 'C:\Program Files\Cloudbase Solutions\Cloudbase-Init\conf'
   ```

7. （可选）在 Sysprep 前创建 VM 快照，以防配置错误
8. 运行 Sysprep：

```
C:\Windows\System32\Sysprep\sysprep.exe /generalize /oobe /unattend: ↩
   Unattend.xml
```

完成上述步骤后，VM 应因 Sysprep 而处于关机状态。现在可以将其转换为模板，克隆后再按需配置。

10.8.7 Cloud-Init 专用选项

cicustom: [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>]
: 指定自定义文件，用于在启动时替换自动生成的文件。

  meta=<volume>
  : 指定一个自定义文件，其中包含通过 cloud-init 传递给虚拟机的所有元数据。此内容与 provider 相关，意味着 configdrive2 和 nocloud 的格式不同。

  network=<volume>
  : 通过 cloud-init 向虚拟机传递包含所有网络数据的自定义文件。

  user=<volume>
  : 通过 cloud-init 向虚拟机传递包含所有用户数据的自定义文件。

  vendor=<volume>
  : 通过 cloud-init 向虚拟机传递包含所有 vendor 数据的自定义文件。

cipassword: <string>
: 要分配给用户的密码。通常不建议使用此项，应改用 SSH 密钥。另请注意，较旧版本的 cloud-init 不支持哈希密码。

citype: <configdrive2 | nocloud | opennebula>
: 指定 cloud-init 配置格式。默认值取决于配置的操作系统类型（ostype）。Linux 使用 nocloud 格式，Windows 使用 configdrive2。

ciupgrade: <boolean> (*default* = 1)
: 首次启动后自动执行软件包升级。

ciuser: <string>
: 要修改 SSH 密钥和密码的用户名，用于替代镜像中配置的默认用户。

ipconfig[n]: [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>]
: 为对应接口指定 IP 地址和网关。

  IP 地址使用 CIDR 表示法；网关为可选项，但必须已指定同类型的 IP 地址。

  IP 地址可以使用特殊字符串 *dhcp* 来启用 DHCP；在这种情况下不应提供显式网关。 对于 IPv6，可以使用特殊字符串 *auto* 来启用无状态自动配置。这需要 cloud-init 19.4 或更新版本。

  如果启用了 cloud-init，且既未指定 IPv4 地址也未指定 IPv6 地址，则默认对 IPv4 使用 dhcp。

gw=<GatewayIPv4>
IPv4 流量的默认网关。

> **Note**
> 需要选项：ip

gw6=<GatewayIPv6>
IPv6 流量的默认网关。

> **Note**
> 需要选项：ip6

ip=<IPv4Format/CIDR> (*default* = dhcp)
CIDR 格式的 IPv4 地址。

ip6=<IPv6Format/CIDR> (*default* = dhcp)
CIDR 格式的 IPv6 地址。

nameserver: <string>
设置容器的 DNS 服务器 IP 地址。创建时如果既未设置 searchdomain 也未设置 nameserver，则会自动使用主机上的设置。

searchdomain: <string>
设置容器的 DNS 搜索域。创建时如果既未设置 searchdomain 也未设置 nameserver，则会自动

sshkeys: <string>
设置 SSH 公钥（每行一个密钥，OpenSSH 格式）。

10.9 PCI(e) 直通

PCI(e) 直通是一种机制，可将主机上的 PCI 设备控制权交给虚拟机。相比使用虚拟化 硬件，它可能具备 offloading）。

但是，如果将某个设备直通给虚拟机，就无法再在主机或任何其他虚拟机中使用该设备。

请注意，PCI 直通可用于 i440fx 和 q35 机型，而 PCIe 直通仅可用于 q35 机型。这并 不意味着支持 PCIe 的设备在以 PCI 设备方式直通时只能以 PCI 速度运行。以 PCIe 方式直通设备只是为客户机设置一 PCIe 设备，而不是“非常快 的传统 PCI 设备”。某些客户机应用会因此受益。

10.9.1 通用要求

由于直通是在真实硬件上执行的，因此需要满足一些要求。下面简要概述这些要求；有关 特定设备的更 PCI Passthrough Examples。

硬件

你的硬件需要支持具备 interrupt remapping 能力的 IOMMU（**I/O M**emory **M**anagement **U**nit），
CPU 和主板。

通常，带 VT-d 的 Intel 系统以及带 AMD-Vi 的 AMD 系统支持此功能。但由于硬件实现 不佳、驱动缺失

此外，服务器级硬件通常比消费级硬件提供更好的支持，但即便如此，许多现代系统也 可以支持此功能

请咨询硬件供应商，确认其是否在你的特定配置下支持 Linux 上的此功能。

确定 **PCI** 卡地址

最简单的方法是在虚拟机的硬件选项卡中，通过 GUI 添加类型为 "Host PCI" 的设备。 也可以使用命令

可以使用以下命令定位你的设备：

```
lspci
```

配置

确认硬件支持直通后，需要进行一些配置以启用 PCI(e) 直通。

IOMMU

需要在 BIOS/UEFI 中启用 IOMMU 支持。通常对应设置称为 IOMMU 或 VT-d，但具体 选项名称应以主板

在 AMD CPU 上，IOMMU 默认启用。对于较新的内核（6.8 或更新版本），Intel CPU 也 是如此。在较
内核命令行中添加以下内容，为 Intel CPU 启用该功能：

```
intel_iommu=on
```

IOMMU 直通模式

如果硬件支持 IOMMU 直通模式，启用该模式可能会提升性能。这是因为虚拟机会绕过 hypervisor 通常执行的（默认）DMA 转换，而是将 DMA 请求直接传递给硬件 IOMMU。 要启用这些选项，请

```
iommu=pt
```

到 内核命令行。

在 arm64/loongarch64 平台上，IOMMU 直通默认已启用。如果未使用 pve-port-kernel， 要启用这

```
iommu.passthrough=1
```

内核模块

必须确保加载以下模块。可以通过将它们添加到 ’/etc/modules” 来实现。

> 中介设备直通
> 如果直通 mediated devices（例如 vGPU），则不需要以下内容。在这些情况下，设备会直接由相应主机驱动拥有。

```
vfio
vfio_iommu_type1
vfio_pci
```

更改任何与模块相关的内容后，需要刷新 initramfs。在 Proxmox VE 上可以执行：

```
# update-initramfs -u -k all
```

要检查模块是否已加载，以下命令的输出：

```
# lsmod | grep vfio
```

应包含上面列出的四个模块。

完成配置

最后重启以使变更生效，并检查它确实已启用。

```
# dmesg |grep iommu
```

应显示 Adding to iommu group X。根据硬件和内核不同，确切消息可能有所差异。

有关如何排查或验证 IOMMU 是否按预期工作的说明，请参见我们 Wiki 中的 Verifying IOMMU Parameters 章节。

还需要确保要直通的设备位于*单独*的 IOMMU 组中。可以通过调用 Proxmox VE API 检查：

```
# pvesh get /nodes/{nodename}/hardware/pci --pci-class-blacklist ””
```

如果设备与其功能（例如 GPU 及其 HDMI Audio 设备）、root port 或 PCI(e) bridge 位于同一个 IOMMU 组中，这是可以接受的。

常用命令示例

以下命令常用于确认 PCI 设备、驱动绑定和 IOMMU 分组状态：

```
# lspci -nn
# lspci -nnk
# pvesh get /nodes/{nodename}/hardware/pci --pci-class-blacklist ””
```

> **PCI(e) slots**
> 某些平台对物理 PCI(e) 插槽的处理方式不同。因此，如果没有得到期望的 IOMMU 组隔离效果，有时将卡插入另一个 PCI(e) 插槽会有所帮助。

不安全中断
对于某些平台（arm64/loongarch64），可能需要允许不安全中断。为此，请在 **/etc/modprobe.d/** 中以 '.conf' 结尾的文件里添加以下行：

```
options vfio_iommu_type1 allow_unsafe_interrupts=1
```

请注意，此选项可能使系统不稳定。

GPU 直通说明

无法通过 Proxmox VE Web 界面中的 NoVNC 或 SPICE 显示 GPU 的 framebuffer。

直通整块 GPU 或 vGPU 且需要图形输出时，必须将显示器物理连接到该显卡，或者在 客户机内部配置远
VNC 或 RDP）。

如果只是希望将 GPU 用作硬件加速器，例如用于使用 OpenCL 或 CUDA 的程序，则不 需要这样做。

10.9.2 主机设备直通

PCI(e) 直通最常见的形式是直通整块 PCI(e) 卡，例如 GPU 或网卡。

主机配置

Proxmox VE 会尝试自动使 PCI(e) 设备对主机不可用。不过，如果该机制不起作用，可以采取 两种方式

- 通过添加以下内容，将设备 ID 传递给 *vfio-pci* 模块选项

  ```
  options vfio-pci ids=1234:5678,4321:8765
  ```

  到 **/etc/modprobe.d/** 中的 .conf 文件，其中 1234:5678 和 4321:8765 是通过 以下命令获得的
  ID：

  ```
  # lspci -nn
  ```

- 在主机上完全 blacklist 该驱动，确保它可以自由绑定用于直通，使用

  ```
  blacklist DRIVERNAME
  ```

  写入 **/etc/modprobe.d/** 中的 .conf 文件。

  要查找驱动名称，请执行

  ```
  # lspci -k
  ```

  例如：

  ```
  # lspci -k | grep -A 3 "VGA"
  ```

  会输出类似如下内容：

```
01:00.0 VGA compatible controller: NVIDIA Corporation GP108 [GeForce GT  ↩
   1030] (rev a1)
        Subsystem: Micro-Star International Co., Ltd. [MSI] GP108 [ ↩
           GeForce GT 1030]
        Kernel driver in use: <some-module>
        Kernel modules: <some-module>
```

现在可以将驱动写入 .conf 文件来 blacklist 它们：

```
echo "blacklist <some-module>" >> /etc/modprobe.d/blacklist.conf
```

对于这两种方法，都需要再次 更新 initramfs，然后重启。

如果仍然不起作用，可能需要设置软依赖，使 GPU 模块在加载 *vfio-pci* 之前加载。 可以使用 *softdep* 标志实现；更多信息也可参见 *modprobe.d* 的 manpage。

例如，如果使用名为 <some-module> 的驱动：

```
# echo "softdep <some-module> pre: vfio-pci" >> /etc/modprobe.d/<some- ↩
   module>.conf
```

验证配置

要检查变更是否成功，可以使用

```
# lspci -nnk
```

并检查设备条目。如果其中显示：

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

或者完全没有 *in use* 行，则该设备已准备好用于直通。

中介设备
对于 mediated devices，该行会有所不同，因为设备会直接由主机驱动拥有，而不是 *vfio-pci*。

虚拟机配置

直通 GPU 时，使用 *q35* 作为机型、使用 *OVMF*（虚拟机中的 *UEFI*）而不是 SeaBIOS、使用 PCIe 而不是 PCI，可获得最佳兼容性。请注意，如果要使用 *OVMF* 进行 GPU 直通，GPU 需要具备支持 UEFI 的 ROM；否则请改用 SeaBIOS。要检查 ROM 是否支持 UEFI，请参见 PCI Passthrough Examples Wiki。

此外，在使用 OVMF 时，可能可以禁用 vga arbitration，从而减少启动期间需要运行的 legacy code 数量。要禁用 vga arbitration：

```
 echo "options vfio-pci ids=<vendor-id>,<device-id> disable_vga=1" > /etc/ ↩
    modprobe.d/vfio.conf
```

将 <vendor-id> 和 <device-id> 替换为从以下命令获取的值：

```
# lspci -nn
```

可以在 Web 界面的虚拟机硬件部分添加 PCI 设备。也可以使用命令行；在虚拟机配置 中设置 **host-pciX** 选项，例如执行：

```
# qm set VMID -hostpci0 00:02.0
```

或者向虚拟机配置文件添加一行：

```
 hostpci0: 00:02.0
```

如果设备有多个功能（例如‘00:02.0’和‘00:02.1），可以使用缩写语法 ``*00:02*` 将它们一起直通。Web 界面中勾选 ``All Functions`’ 复选框。

根据设备和客户机操作系统不同，可能需要使用一些选项：

- **x-vga=on|off** 将 PCI(e) 设备标记为虚拟机的主 GPU。启用后，**vga** 配置选项会被 忽略。
- **pcie=on|off** 告诉 Proxmox VE 使用 PCIe 还是 PCI 端口。某些客户机/设备组合需要 PCIe 而不是 PCI。PCIe 仅适用于 *q35* 机型。
- **rombar=on|off** 使 firmware ROM 对客户机可见。默认为 on。某些 PCI(e) 设备需要 禁用此项。
- **ramfb=on|off** 为 vgpu 设备显示 ramfb 设备。默认为 on。
- **romfile=<path>** 是设备使用的 ROM 文件的可选路径。这是 **/usr/share/kvm/** 下的 相对路径。

示例

以下示例展示将 GPU 设置为主设备的 PCIe 直通：

```
# qm set VMID -hostpci0 02:00,pcie=on,x-vga=on
```

PCI ID 覆盖

可以覆盖客户机看到的 PCI vendor ID、device ID 和 subsystem ID。如果设备是某个 变体，其 ID 无法被客户机驱动识别，但你仍希望强制加载这些驱动，这会很有用（例如你知道该设备与受支持变

可用选项包括 `vendor-id`、`device-id`、`sub-vendor-id` 和 `sub-device-id`。可以 设置其中任意 ID。

例如：

```
# qm set VMID -hostpci0 02:00,device-id=0x10f6,sub-vendor-id=0x0000
```

10.9.3 SR-IOV

PCI(e) 设备直通的另一种形式，是在设备可用时使用其硬件虚拟化功能。

启用 **SR-IOV**
要使用 SR-IOV，平台支持尤其重要。可能需要先在 BIOS/UEFI 中启用此功能，或者使用 特定 PCI(e) 端口才能工作。如有疑问，请查阅平台手册或联系供应商。

SR-IOV（**S**ingle-**R**oot **I**nput/**O**utput **V**irtualization）允许单个 设备向系统提供多个 *VF*（**V**irtual **F**unctions）。每个 *VF* 都可以在不同虚拟机 中使用，具备完整硬件功能，并且相比软件虚拟化设备具有

目前最常见的使用场景是支持 SR-IOV 的 NIC（**N**etwork **I**nterface **C**ard），它们可以为每个物理端口 VF。这允许在虚拟机内部使用 checksum offloading 等功能，从而降低（主机）CPU 开销。

主机配置

通常，有两种方法可在设备上启用 virtual functions。

- 有时驱动模块会提供相应选项，例如某些 Intel 驱动

  ```
  max_vfs=4
  ```

  可将其放入 **/etc/modprobe.d/** 下以 *.conf* 结尾的文件中。（之后不要忘记更新 initramfs）

  请参阅驱动模块文档以了解确切参数和选项。

- 第二种更通用的方法是使用 `sysfs`。如果设备和驱动支持此方式，可以动态更改 VF 数量。例如，要 0000:01:00.0 上设置 4 个 VF，请执行：

  ```
  # echo 4 > /sys/bus/pci/devices/0000:01:00.0/sriov_numvfs
  ```

  要使此变更持久化，可以使用 Debian 软件包 ‘sysfsutils`。安装后，通过 **/etc/sysfs.conf** 或 **/etc/sysfs.d/** 中的`FILE.conf’ 进行配置。

虚拟机配置

创建 VF 后，使用 `lspci` 输出时应能看到它们作为独立 PCI(e) 设备出现。获取它们的 ID，并像普通 PCI(e) 设备一样进行直通。

10.9.4 中介设备（**vGPU**、**GVT-g**）

中介设备是另一种让虚拟化硬件复用物理硬件功能和性能的方法。它们最常见 于虚拟化 GPU 部署，例如 Intel 的 GVT-g，以及 NVIDIA 在其 GRID 技术中使用的 vGPU。

通过这种方式，物理卡可以创建虚拟卡，类似于 SR-IOV。区别在于中介设备 不会在主机中显示为 PCI(e) 设备，因此只适合在虚拟机中使用。

主机配置

通常，你的显卡驱动必须支持该功能，否则无法工作。因此，请向供应商了解兼容驱动 以及配置方法。

Intel 的 GVT-g 驱动已集成在内核中，应可与第 5、第 6、第 7 代 Intel Core 处理器， 以及 E3 v4、E3 v5 和 E3 v6 Xeon 处理器配合使用。

要为 Intel Graphics 启用它，必须确保加载 *kvmgt* 模块（例如通过 `/etc/modules`）， 并在内核命令行中启用它，添加以下参数：

```
i915.enable_gvt=1
```

之后记得 更新 initramfs，并重启 主机。

虚拟机配置

要使用 mediated device，只需在 `hostpciX` 虚拟机配置选项上指定 `mdev` 属性。

可以通过 *sysfs* 获取支持的设备。例如，要列出设备 *0000:00:02.0* 支持的类型，只需 执行：

```
# ls /sys/bus/pci/devices/0000:00:02.0/mdev_supported_types
```

每个条目都是一个目录，其中包含以下重要文件：

- *available_instances* 包含该类型仍可用的实例数量；虚拟机中每使用一个 *mdev* 都会 减少该数量。
- *description* 包含该类型能力的简短描述。
- *create* 是创建此类设备的端点；如果配置了带 `mdev` 的 *hostpciX* 选项，Proxmox VE 会自动为你执

使用 `Intel GVT-g vGPU（Intel Skylake 6700k）`的配置示例：

```
# qm set VMID -hostpci0 00:02.0,mdev=i915-GVTg_V5_4
```

设置后，Proxmox VE 会在虚拟机启动时自动创建此类设备，并在虚拟机停止时再次清理它。

10.9.5 在集群中使用

也可以在集群层级映射设备，以便它们可与 HA 正确配合使用、可检测硬件变更，并且 非 root 用户也可
Resource Mapping。

10.9.6 vIOMMU（模拟 IOMMU）

vIOMMU 是在虚拟机内部对硬件 IOMMU 的模拟，可为虚拟化 I/O 设备提供更好的内存 访问控制和安全
vIOMMU 选项还允许通过 Nested Virtualization，在 level-1 虚拟机中将 PCI(e) 设备直通给 level-2 虚拟机。要将物理 PCI(e) 设备从 主机直通到嵌套虚拟机，请遵循 PCI(e) 直通说明。

目前有两种可用的 vIOMMU 实现：Intel 和 VirtIO。

Intel vIOMMU

Intel vIOMMU 特定的虚拟机要求：

- 无论主机使用 Intel 还是 AMD CPU，都必须在虚拟机内核参数中设置 `intel_iommu=on`。
- 要使用 Intel vIOMMU，需要将 **q35** 设置为机型。

如果满足所有要求，可以在应能够直通 PCI 设备的虚拟机配置中，将 `viommu=intel` 添加到 machine 参数。

```
# qm set VMID -machine q35,viommu=intel
```

QEMU VT-d 文档

VirtIO vIOMMU

该 vIOMMU 实现较新，没有 Intel vIOMMU 那么多限制，但目前在生产环境中使用较少， 文档也较少。

使用 VirtIO vIOMMU 时，*不需要*设置任何内核参数。也*不必*使用 q35 作为机型； 但如果要使用 PCIe，建议这样做。

```
# qm set VMID -machine q35,viommu=virtio
```

Michael Zhao 关于 virtio-iommu 的博客文章

10.10 Hook 脚本

可以通过配置属性 `hookscript` 向 VM 添加 hook 脚本。

```
# qm set 100 --hookscript local:snippets/hookscript.pl
```

它会在客户机生命周期的不同阶段被调用。 示例和文档见 `/usr/share/pve-docs/examples/gues` 下的示例脚本。

10.11 休眠

可以通过 GUI 选项 `Hibernate` 将 VM 挂起到磁盘，或使用：

```
# qm suspend ID --todisk
```

这意味着当前内存内容会保存到磁盘，并停止 VM。 下次启动时会加载内存内容，VM 可以从离开的位置

状态存储选择

如果没有为内存指定目标存储，将按以下顺序自动选择第一个可用项：

1. VM 配置中的存储 vmstatestorage。
2. 任意 VM 磁盘中的第一个共享存储。
3. 任意 VM 磁盘中的第一个非共享存储。
4. 作为回退的存储 local。

10.12 资源映射

使用或引用本地资源（例如 PCI 设备地址）时，直接使用原始地址或 ID 有时会产生问题，例如：

- 使用 HA 时，目标节点上可能存在具有相同 ID 或路径的不同设备； 如果将此类客户机分配到 HA 组时不够谨慎，可能会使用错误设备，从而破坏配置。
- 更改硬件可能改变 ID 和路径，因此必须检查所有已分配设备，确认路径或 ID 是否仍然正确。

为了更好地处理此问题，可以定义集群范围资源映射，使资源具有集群唯一且由用户选择的标识符，该标识符可以对应不同主机上的不同设备。这样，HA 不会使用错误设备启动客户机，并且可以检测硬件

可以在 Proxmox VE Web GUI 的 Datacenter 下，通过 Resource Mappings 类别中的相关选项卡创
也可以在 CLI 中使用：

```
# pvesh create /cluster/mapping/<type> <options>
```

其中 <type> 是硬件类型（目前为 pci、usb 或 dir），<options> 是设备映射和其他配置参数。

请注意，选项必须包含带有该硬件所有识别属性的 map 属性，以便验证硬件未发生变化，并且直通的是

例如，要添加一个 PCI 设备作为 device1，它在节点 node1 上的路径为 0000:01:00.0，设备 ID 为 0001，厂商 ID 为 0002，并且在 node2 上为 0000:02:00.0，可以使用：

```
# pvesh create /cluster/mapping/pci --id device1 \
 --map node=node1,path=0000:01:00.0,id=0002:0001 \
 --map node=node2,path=0000:02:00.0,id=0002:0001
```

必须为该设备应具有映射的每个节点重复 map 参数（请注意，目前每个映射中每个节点只能映射一个 USB 设备）。

使用 GUI 会更简单，因为正确属性会被自动选取并发送到 API。

对于 PCI 设备，也可以通过节点的多个 map 属性为每个节点提供多个设备。 如果将此类设备分配给客户
给定路径的顺序也是尝试顺序，因此可以实现任意分配策略。

这对具有 SR-IOV 的设备很有用，因为有时具体直通哪个虚拟功能并不重要。

可以通过 GUI 或以下命令将此类设备分配给客户机：

```
# qm set ID -hostpci0 <name>
```

对于 PCI 设备，或：

```
# qm set <vmid> -usb0 <name>
```

对于 USB 设备。

其中 <vmid> 是客户机 ID，<name> 是为已创建映射选择的名称。 允许使用设备直通的所有常规选项，mdev。

要创建映射，需要在 /mapping/<type>/<name> 上具有 Mapping.Modify （其中 <type> 是设备类型，<name> 是映射名称）。

要使用这些映射，需要在 /mapping/<type>/<name> 上具有 Mapping.Use（此外还需要用于编辑配

附加选项

定义集群范围资源映射时还有一些附加选项。目前包括以下选项：

- mdev (PCI)：这会将 PCI 设备标记为能够提供 mediated devices。 启用后，在客户机上配置时可 PCI 设备， mediated device 会在第一个具有所选类型可用实例的设备上创建。
- live-migration-capable (PCI)：这会将 PCI 设备标记为能够在节点之间在线迁移。 这需要驱动 NVIDIA GPU 支持此功能。 请注意，在线迁移直通设备是实验性功能，可能无法工作或导致问题。

10.13 使用 qm 管理虚拟机

qm 是在 Proxmox VE 上管理 QEMU/KVM 虚拟机的工具。可以创建和销毁虚拟机，并控制其执行（start/stop/suspend/resume）。此外，还可以使用 qm 在关联配置文件中设置参数。 也可以创建和

10.13.1 CLI 使用示例

使用上传到 *local* 存储的 iso 文件，创建一个在 *local-lvm* 存储上具有 4 GB IDE 磁盘的 VM：

```
# qm create 300 -ide0 local-lvm:4 -net0 e1000 -cdrom local:iso/proxmox- ↩
   mailgateway_2.1.iso
```

启动新 VM：

```
# qm start 300
```

发送关闭请求，然后等待 VM 停止。

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

与上面相同，但只等待 40 秒。

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

如果 VM 未关闭，请强制停止它并覆盖任何正在运行的关闭任务。停止 VM 可能导致数据丢失，请谨慎使

```
# qm stop 300 -overrule-shutdown 1
```

销毁 VM 始终会将其从访问控制列表中移除，并始终移除该 VM 的防火墙配置。 如果还想将 VM 从复制作业、备份作业和 HA 资源配置中移除，必须启用 *--purge*。

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

将磁盘镜像移动到其他存储。

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

将磁盘镜像重新分配给其他 VM。这会从源 VM 移除磁盘 `scsi1`，并将其作为 `scsi3` 附加到目标 VM。 后台会重命名磁盘镜像，使其名称与新的所有者匹配。

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

10.14 配置

VM 配置文件存储在 Proxmox 集群文件系统中，可通过 `/etc/pve/qemu-server/<VMID>.conf` 访问。 与存储在 `/etc/pve/` 中的其他文件一样，它们会自动复制到所有其他集群节点。

> **Note**
> VMID < 100 保留用于内部用途，VMID 需要在整个集群范围内唯一。

VM 配置示例

```
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
```

这些配置文件是简单文本文件，可以使用普通文本编辑器（`vi`、`nano` 等）编辑。 这有时对进行小幅修 VM 才能应用此类更改。

因此，通常最好使用 `qm` 命令生成和修改这些文件，或直接使用 GUI 完成全部操作。 我们的工具集足够 VM。 此功能称为 ”hot plug”，在这种情况下不需要重启 VM。

10.14.1 文件格式

VM 配置文件使用简单的冒号分隔键/值格式。每行采用以下格式：

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

这些文件中的空行会被忽略，以 # 字符开头的行会被视为注释并同样被忽略。

10.14.2 快照

创建快照时，qm 会将快照时的配置存储到同一配置文件中的单独快照区段。 例如，创建名为 “test-snapshot” 的快照后，配置文件会类似如下：

带快照的 **VM** 配置

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

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

有几个与快照相关的属性，例如 parent 和 snaptime。 parent 属性用于存储快照之间的父/子关系。是快照创建时间戳（Unix epoch）。

也可以使用选项 vmstate 保存正在运行的 VM 的内存。 有关如何为 VM 状态选择目标存储的详情，请参休眠章节中的 状态存储选择。

10.14.3 选项

gicversion: <2 | 3 | 4 | max | host > (*default* = host)
设置 ARM 虚拟机的 gicversion。

virtualization: <boolean> (*default* = 0)
为 arm64 虚拟机启用嵌套虚拟化。

pxvditemplate <boolean> (*default* = 0)
启用/禁用 PXVDI 模板。

acpi: <boolean> (*default* = 1)
启用/禁用 ACPI。

affinity: <string>
用于执行客户机进程的主机核心列表，例如：0,5,8-11

agent: [enabled=]<1|0> [,freeze-fs-on-backup=<1|0>] [,fstrim_cloned_disks=<1|0>] [,type=<virtio|isa>]
启用/禁用与 QEMU Guest Agent 的通信及其属性。

enabled=<boolean> (*default* = 0)
启用/禁用与虚拟机中运行的 QEMU Guest Agent（QGA）的通信。

freeze-fs-on-backup=<boolean> (*default* = 1)
备份时冻结/解冻客户机文件系统，以保证一致性。

fstrim_cloned_disks=<boolean> (*default* = 0)
移动磁盘或迁移虚拟机后运行 fstrim。

type=<isa | virtio> (*default* = virtio)
选择 agent 类型。

amd-sev: [type=]<sev-type> [,allow-smt=<1|0>] [,kernel-hashes=<1|0>] [,no-debug=<1|0>] [,no-key-sharing=<1|0>]
AMD CPU 提供的 Secure Encrypted Virtualization（SEV）功能。

allow-smt=<boolean> (*default* = 1)
设置策略位以允许 Simultaneous Multi Threading（SMT）（除 SEV-SNP 外会被忽略）。

kernel-hashes=<boolean> (*default* = 0)
向客户机固件添加 kernel hashes，用于 measured Linux kernel launch。

no-debug=<boolean> (*default* = 0)
设置策略位以禁止调试客户机。

no-key-sharing=<boolean> (*default* = 0)
设置策略位以禁止与其他客户机共享密钥（对 SEV-SNP 会被忽略）。

type=<sev-type>
使用 type=*std* 启用标准 SEV；使用 *es* 选项启用实验性 SEV-ES；使用 *snp* 选项启用实验性 SEV-SNP。

arch: <aarch64 | x86_64>
虚拟处理器架构。默认为主机架构。

args: <string>
传递给 kvm 的任意参数，例如：

args: -no-reboot -smbios *type=0,vendor=FOO*

> **Note**
> 此选项仅供专家使用。

audio0: device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>]
配置音频设备，与 QXL/Spice 结合使用时很有用。

device=<AC97 | ich9-intel-hda | intel-hda>
配置音频设备。

driver=<none | spice> (*default* = spice)
音频设备的驱动后端。

autostart: <boolean> (*default* = 0)
崩溃后自动重启（当前会被忽略）。

balloon: <integer> (0 - N)
虚拟机目标 RAM 大小，单位为 MiB。设置为零会禁用 balloon 驱动。

bios: <ovmf | seabios> (*default* = seabios)
选择 BIOS 实现。

boot: [[legacy=]<[acdn]{1,4}>] [,order=<device[;device...]>]
指定客户机启动顺序。应使用 *order=* 子属性；无键用法或 *legacy=* 已弃用。

legacy=<[acdn]{1,4}> (*default* = cdn)
从软盘 (a)、硬盘 (c)、CD-ROM (d) 或网络 (n) 启动。已弃用，请改用 *order=*。

order=<device[;device...]>
客户机会按照此处设备出现的顺序尝试启动。
磁盘、光驱和直通的存储 USB 设备会被直接用于启动； NIC 会加载 PXE，PCIe 设备则会像
例如 NVMe，或加载 option ROM，例如 RAID 控制器、硬件 NIC。
请注意，只有此列表中的设备会被标记为可启动，并因此由 客户机固件（BIOS/UEFI）加载。
例如 software-raid，需要在此处指定全部磁盘。
指定后会覆盖已弃用的 *legacy=[acdn]** 值。

bootdisk: (ide|sata|scsi|virtio)\d+
允许从指定磁盘启动。已弃用：请改用 *boot: order=foo;bar*。

cdrom: <volume>
这是选项 -ide2 的别名。

cicustom: [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>]
cloud-init：指定自定义文件，用于在启动时替换自动生成的文件。

meta=<volume>
指定一个自定义文件，其中包含通过 cloud-init 传递给虚拟机的所有元数据。 此内容与 provider 相关，意味着 configdrive2 和 nocloud 的格式不同。

network=<volume>
通过 cloud-init 向虚拟机传递包含所有网络数据的自定义文件。

user=<volume>
通过 cloud-init 向虚拟机传递包含所有用户数据的自定义文件。

vendor=<volume>
通过 cloud-init 向虚拟机传递包含所有 vendor 数据的自定义文件。

cipassword: <string>
cloud-init：要分配给用户的密码。通常不建议使用此项，应改用 SSH 密钥。另请注意，较旧版本
cloud-init 不支持哈希密码。

citype: <configdrive2 | nocloud | opennebula>
指定 cloud-init 配置格式。默认值取决于配置的操作系统类型（`ostype`）。Linux 使用 `nocloud`
格式，Windows 使用 `configdrive2`。

ciupgrade: <boolean> (*default* = 1)
cloud-init：首次启动后自动执行软件包升级。

ciuser: <string>
cloud-init：要修改 SSH 密钥和密码的用户名，用于替代镜像中配置的默认用户。

cores: <integer> (1 - N) (*default* = 1)
每个 socket 的核心数。

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

cputype=<string> (*default* = kvm64)
模拟的 CPU 类型。可以是默认名称或自定义名称（自定义模型名称必须以 *custom-* 为前缀）

flags=<+FLAG[;-FLAG...]>
额外 CPU flags 列表，以 *;* 分隔。使用 *+FLAG* 启用 flag，使用 *-FLAG* 禁用 flag。自定义 CPU 模型可以指定 QEMU/KVM 支持的任意 flag；出于安全原因，虚拟机专用 flag 必须来自 spec-ctrl, ibpb, ssbd, virt-ssbd, amd-ssbd, amd-no-ssb, pdpe1gb, md-clear, hv-tlbflush, hv-evmcs, aes

hidden=<boolean> (*default* = 0)
不要将其标识为 KVM 虚拟机。

hv-vendor-id=<vendor-id>
Hyper-V vendor ID。Windows 客户机内的某些驱动或程序需要特定 ID。

phys-bits=<8-64|host>
报告给客户机操作系统的物理内存地址位数。应小于或等于主机值。设置为 *host* 可使用主机 CPU 的值，但请注意，这会破坏到具有其他值的 CPU 的在线迁移。

reported-model=<486 | Broadwell | Broadwell-IBRS | Broadwell-noTSX | Broadwell-noTSX-IBRS | Cascadelake-Server | Cascadelake-Server-noTSX | Cascadelake-Server-v2 | Cascadelake-Server-v4 | Cascadelake-Server-v5 | Conroe | Cooperlake | Cooperlake-v2 | EPYC | EPYC-Genoa | EPYC-IBPB | EPYC-Milan | EPYC-Milan-v2 | EPYC-Rome | EPYC-Rome-v2 | EPYC-Rome-v3 | EPYC-Rome-v4 | EPYC-v3 | EPYC-v4 | GraniteRapids | Haswell | Haswell-IBRS | Haswell-noTSX | Haswell-noTSX-IBRS | Icelake-Client | Icelake-Client-noTSX | Icelake-Server | Icelake-Server-noTSX | Icelake-Server-v3 | Icelake-Server-v4 | Icelake-Server-v5 | Icelake-Server-v6 | IvyBridge | IvyBridge-IBRS | KnightsMill | Nehalem | Nehalem-IBRS | Opteron_G1 | Opteron_G2 | Opteron_G3 | Opteron_G4 | Opteron_G5 | Penryn | SandyBridge | SandyBridge-IBRS | SapphireRapids | SapphireRapids-v2 | Skylake-Client | Skylake-Client-IBRS | Skylake-Client-noTSX-IBRS | Skylake-Client-v4 | Skylake-Server | Skylake-Server-IBRS | Skylake-Server-noTSX-IBRS | Skylake-Server-v4 | Skylake-Server-v5 | Westmere | Westmere-IBRS | athlon | core2duo | coreduo | host | kvm32 | kvm64 | max | pentium | pentium2 | pentium3 | phenom | qemu32 | qemu64> (*default* = kvm64)
报告给客户机的 CPU 模型和厂商。必须是 QEMU/KVM 支持的模型。仅对自定义 CPU 模型定义有效，默认模型始终会向客户机操作系统报告其自身。

cpulimit: <number> (0 - 128) (*default* = 0)
CPU 使用限制。

Note
如果计算机有 2 个 CPU，则总共有 *2* 份 CPU 时间。值 *0* 表示不限制 CPU。

cpuunits: <integer> (1 - 262144) (*default* = cgroup v1: 1024, cgroup v2: 100)
虚拟机的 CPU 权重。该参数用于内核公平调度器。数值越大，该虚拟机获得的 CPU 时间越多。该

description: <string>
虚拟机说明。显示在 Web 界面的虚拟机摘要中，并作为注释保存在配置文件内。

efidisk0: [file=]<volume> [,efitype=<2m|4m>] [,format=<enum>] [,pre-enrolled-keys=<1|0>] [,size=<DiskSize>]
配置用于存储 EFI vars 的磁盘。

efitype=<2m | 4m> (*default* = 2m)
OVMF EFI vars 的大小和类型。*4m* 较新且推荐使用，并且 Secure Boot 需要它。为保持向 *2m*。对于 arch=aarch64（ARM）的虚拟机会被忽略。

file=<volume>
驱动器的后端卷。

format=<cloop | qcow | qcow2 | qed | raw | vmdk>
驱动器后端文件的数据格式。

pre-enrolled-keys=<boolean> (*default* = 0)
在使用 *efitype=4m* 时，使用已注册发行版特定密钥和 Microsoft Standard keys 的 EFI vars 模板。请注意，这会默认启用 Secure Boot，但仍可在虚拟机内部关闭。

size=<DiskSize>
磁盘大小。此项仅供信息展示，没有实际影响。

freeze: <boolean>
启动时冻结 CPU（使用 *c* monitor 命令开始执行）。

hookscript: <string>
在虚拟机生命周期的各个阶段执行的脚本。

hostpci[n]: [[host=]<HOSTPCIID[;HOSTPCIID2...]>] [,device-id=<hex id>] [,legacy-igd=<1|0>] [,mapping=<mapping-id>] [,mdev=<string>] [,pcie=<1|0>] [,rombar=<1|0>] [,romfile=<string>] [,sub-device-id=<hex id>] [,sub-vendor-id=<hex id>] [,vendor-id=<hex id>] [,x-vga=<1|0>] [,ramfb=<1|0>]
将主机 PCI 设备映射到客户机。

Note
此选项允许直接访问主机硬件。因此此类机器将 无法再迁移，请特别谨慎使用。

Caution
实验性功能！已有用户报告此选项存在问题。

device-id=<hex id>
覆盖客户机可见的 PCI device ID。

host=<HOSTPCIID[;HOSTPCIID2...]>
主机 PCI 设备直通。主机 PCI 设备的 PCI ID，或主机 PCI virtual functions 列表。HOSTPC
语法为：

bus:dev.func (hexadecimal numbers)

可以使用 *lspci* 命令列出现有 PCI 设备。

此项和 *mapping* 键必须至少设置一个。

legacy-igd=<boolean> (*default* = 0)
以 legacy IGD 模式传递此设备，使其成为虚拟机中的主图形设备且为独占图形设备。需要 *pc-i440fx* 机器类型，并将 VGA 设置为 *none*。

mapping=<mapping-id>
集群范围 mapping 的 ID。此项和默认键 *host* 必须至少设置一个。

mdev=<string>
要使用的 mediated device 类型。 虚拟机启动时会创建此类型的一个实例， 虚拟机停止时会

pcie=<boolean> (*default* = 0)
选择 PCI-express 总线（需要 *q35* 机器模型）。

rombar=<boolean> (*default* = 1)
指定设备 ROM 是否在客户机内存映射中可见。

romfile=<string>
自定义 PCI 设备 ROM 文件名（必须位于 /usr/share/kvm/）。

sub-device-id=<hex id>
覆盖客户机可见的 PCI subsystem device ID。

sub-vendor-id=<hex id>
覆盖客户机可见的 PCI subsystem vendor ID。

vendor-id=<hex id>
覆盖客户机可见的 PCI vendor ID。

x-vga=<boolean> (*default* = 0)
启用 vfio-vga 设备支持。

ramfb=<boolean> (*default* = 1)
启用 vfio-pci 设备 ramfb 显示支持。

hotplug: <string> (*default* = network,disk,usb)
选择性启用热插拔功能。这是以逗号分隔的热插拔功能列表：*network*、*disk*、*cpu*、*memory*、和 *cloudinit*。使用 *0* 可完全禁用热插拔。值 *1* 是默认值 network,disk,usb 的别名。对于 machine version >= 7.1 且 ostype 为 l26 或 windows > 7 的客户机，可以使用 USB 热插拔。

hugepages: <1024 | 2 | any>
启用/禁用 hugepages 内存。

**ide[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>] [,model=<model>]
[,replicate=<1|0>] [,rerror=<ignore|report|stop>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>]**

将卷用作 IDE 硬盘或 CD-ROM (n 为 0 到 3) 。

aio=<io_uring | native | threads>

要使用的 AIO 类型。

backup=<boolean>

进行备份时是否应包含该驱动器。

bps=<bps>

最大读/写速度，单位为字节每秒。

bps_max_length=<seconds>

I/O burst 的最大长度，单位为秒。

bps_rd=<bps>

最大读取速度，单位为字节每秒。

bps_rd_max_length=<seconds>

读取 I/O burst 的最大长度，单位为秒。

bps_wr=<bps>

最大写入速度，单位为字节每秒。

bps_wr_max_length=<seconds>

写入 I/O burst 的最大长度，单位为秒。

cache=<directsync | none | unsafe | writeback | writethrough>

驱动器的缓存模式。

cyls=<integer>

强制驱动器物理几何参数使用指定柱面数。

detect_zeroes=<boolean>

控制是否检测并尝试优化零写入。

discard=<ignore | on>

控制是否将 discard/trim 请求传递到底层存储。

file=<volume>

驱动器的后端卷。

format=<cloop | qcow | qcow2 | qed | raw | vmdk>
驱动器后端文件的数据格式。

heads=<integer>
强制驱动器物理几何参数使用指定磁头数。

iops=<iops>
最大读/写 I/O，单位为操作每秒。

iops_max=<iops>
最大未限速读/写 I/O 池，单位为操作每秒。

iops_max_length=<seconds>
I/O burst 的最大长度，单位为秒。

iops_rd=<iops>
最大读取 I/O，单位为操作每秒。

iops_rd_max=<iops>
最大未限速读取 I/O 池，单位为操作每秒。

iops_rd_max_length=<seconds>
读取 I/O burst 的最大长度，单位为秒。

iops_wr=<iops>
最大写入 I/O，单位为操作每秒。

iops_wr_max=<iops>
最大未限速写入 I/O 池，单位为操作每秒。

iops_wr_max_length=<seconds>
写入 I/O burst 的最大长度，单位为秒。

mbps=<mbps>
最大读/写速度，单位为兆字节每秒。

mbps_max=<mbps>
最大未限速读/写池，单位为兆字节每秒。

mbps_rd=<mbps>
最大读取速度，单位为兆字节每秒。

mbps_rd_max=<mbps>
最大未限速读取池，单位为兆字节每秒。

mbps_wr=<mbps>
最大写入速度，单位为兆字节每秒。

mbps_wr_max=<mbps>
最大未限速写入池，单位为兆字节每秒。

media=<cdrom | disk> (*default* = disk)
驱动器的介质类型。

model=<model>
驱动器报告的型号名称，经过 URL 编码，最长 40 字节。

replicate=<boolean> (*default* = 1)
该驱动器是否应纳入复制任务。

rerror=<ignore | report | stop>
读取错误时的操作。

secs=<integer>
强制驱动器物理几何参数使用指定扇区数。

serial=<serial>
驱动器报告的序列号，经过 URL 编码，最长 20 字节。

shared=<boolean> (*default* = 0)
将此本地管理的卷标记为在所有节点上可用。

Warning
此选项不会自动共享该卷，而是假定它已经被共享。

size=<DiskSize>
磁盘大小。此项仅供信息展示，没有实际影响。

snapshot=<boolean>
控制 qemu 的 snapshot mode 功能。如果启用，对磁盘所做的更改是临时的，并会在虚拟

ssd=<boolean>
是否将此驱动器暴露为 SSD，而不是旋转式硬盘。

trans=<auto | lba | none>
强制磁盘几何参数 BIOS translation 模式。

werror=<enospc | ignore | report | stop>
写入错误时的操作。

wwn=<wwn>
驱动器的 worldwide name，编码为 16 字节十六进制字符串，并以 *0x* 为前缀。

ipconfig[n]: [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>]
cloud-init：为对应接口指定 IP 地址和网关。

IP 地址使用 CIDR 表示法；网关为可选项，但必须已指定同类型的 IP 地址。

IP 地址可以使用特殊字符串 *dhcp* 来启用 DHCP；在这种情况下 不应提供显式网关。 对于 IPv6，可以使用特殊字符串 *auto* 来启用无状态自动配置。这需要 cloud-init 19.4 或更新版本。

如果启用了 cloud-init，且既未指定 IPv4 地址也未指定 IPv6 地址，则默认使用 IPv4 dhcp。

gw=<GatewayIPv4>
IPv4 流量的默认网关。

Note
需要选项：ip

gw6=<GatewayIPv6>
IPv6 流量的默认网关。

Note
需要选项：ip6

ip=<IPv4Format/CIDR> (*default* = dhcp)
CIDR 格式的 IPv4 地址。

ip6=<IPv6Format/CIDR> (*default* = dhcp)
CIDR 格式的 IPv6 地址。

ivshmem: size=<integer> [,name=<string>]
虚拟机间共享内存。适用于虚拟机之间或虚拟机与主机之间的直接通信。

name=<string>
文件名称。会添加 *pve-shm-* 前缀。默认为 VMID。虚拟机停止时会删除该文件。

size=<integer> (1 - N)
文件大小，单位为 MB。

keephugepages: <boolean> (*default* = 0)
与 hugepages 配合使用。启用后，虚拟机关机后 hugepages 不会被删除，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>
VNC 服务器的键盘布局。通常不需要此选项，更适合在客户机操作系统内部处理。

kvm: <boolean> (*default* = 1)
启用/禁用 KVM 硬件虚拟化。

localtime: <boolean>
将 real time clock（RTC）设置为本地时间。如果 ostype 表示 Microsoft Windows OS，则默认

lock: <backup | clone | create | migrate | rollback | snapshot | snapshot-delete | suspended | suspending>
锁定/解锁虚拟机。

machine: [[type=]<machine type>] [,enable-s3=<1|0>] [,enable-s4=<1|0>] [,viommu=<intel|virtio>]
指定 QEMU machine。

enable-s3=<boolean>
启用 S3 电源状态。从 machine types 9.2+pve1 开始默认为 false，之前默认为 true。

enable-s4=<boolean>
启用 S4 电源状态。从 machine types 9.2+pve1 开始默认为 false，之前默认为 true。

type=<machine type>
指定 QEMU machine type。

viommu=<intel | virtio>
启用并设置客户机 vIOMMU 变体（Intel vIOMMU 需要将 machine type 设置为 q35）。

memory: [current=]<integer>
内存属性。

current=<integer> (16 - N) (*default* = 512)
虚拟机当前在线 RAM 容量，单位为 MiB。使用 balloon 设备时，这是最大可用内存。

migrate_downtime: <number> (0 - N) (*default* = 0.1)
设置迁移可容忍的最大停机时间（秒）。如果迁移最后阶段由于需要传输过多新写脏内存而无法收

migrate_speed: <integer> (0 - N) (*default* = 0)
设置迁移最大速度（MB/s）。值 0 表示不限制。

name: <string>
设置虚拟机名称。仅用于配置 Web 界面。

nameserver: <string>
cloud-init：设置容器的 DNS 服务器 IP 地址。创建时如果既未设置 searchdomain 也未设置 nameserver，则会自动使用主机上的设置。

net[n]: [model=]<enum> [,bridge=<bridge>] [,firewall=<1|0>] [,link_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX:XX>] [,mtu=<integer>] [,queues=<integer>] [,rate=<number>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,<model>=<macaddr>]
指定网络设备。

bridge=<bridge>
网络设备要连接到的 bridge。Proxmox VE 标准 bridge 名为 *vmbr0*。
如果未指定 bridge，系统会创建一个 kvm user（NATed）网络设备， 并提供 DHCP 和 DNS 服务。使用以下地址：

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

DHCP 服务器会从 10.0.2.15 开始为客户机分配地址。

firewall=<boolean>
此接口是否应受防火墙保护。

link_down=<boolean>
此接口是否应断开连接（类似拔掉网线）。

macaddr=<XX:XX:XX:XX:XX:XX>
未设置 I/G（Individual/Group）位的普通 MAC 地址。

model=<e1000 | e1000-82540em | e1000-82544gc | e1000-82545em | e1000e | i82551 | i82557b | i82559er | ne2k_isa | ne2k_pci | pcnet | rtl8139 | virtio | vmxnet3>
网卡模型。*virtio* 模型以很低的 CPU 开销提供最佳性能。如果客户机不支持此驱动，通常最
e1000。

mtu=<integer> (1 - 65520)
强制 MTU，仅适用于 VirtIO。设置为 *1* 表示使用 bridge MTU。

queues=<integer> (0 - 64)
设备上使用的数据包队列数量。

rate=<number> (0 - N)
速率限制，单位为 mbps（兆字节每秒），使用浮点数表示。

tag=<integer> (1 - 4094)
应用到此接口数据包的 VLAN tag。

trunks=<vlanid[;vlanid...]>
允许通过此接口的 VLAN trunks。

numa: <boolean> (*default* = 0)
启用/禁用 NUMA。

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

cpus=<id[-id];...>
访问此 NUMA 节点的 CPU。

hostnodes=<id[-id];...>
要使用的主机 NUMA 节点。

memory=<number>
此 NUMA 节点提供的内存量。

policy=<bind | interleave | preferred>
NUMA 分配策略。

onboot: <boolean> (*default* = 0)
指定虚拟机是否在系统启动期间启动。

ostype: <l24 | l26 | other | solaris | w2k | w2k3 | w2k8 | win10 | win11 | win7 | win8 | wvista | wxp>
指定客户机操作系统。用于为特定操作系统启用特殊 优化/功能：

other	未指定 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 11/2022/2025

l24	Linux 2.4 Kernel
l26	Linux 2.6 - 6.X Kernel
solaris	Solaris/OpenSolaris/OpenIndiania kernel

parallel[n]: /dev/parport\d+|/dev/usb/lp\d+
映射主机并口设备（n 为 0 到 2）。

> **Note**
> 此选项允许直接访问主机硬件。因此此类机器将无法再迁移， 请特别谨慎使用。

> **Caution**
> 实验性功能！已有用户报告此选项存在问题。

protection: <boolean> (*default* = 0)
设置虚拟机的保护标志。这会禁用删除虚拟机和删除磁盘操作。

reboot: <boolean> (*default* = 1)
允许重启。如果设置为 *0*，虚拟机会在重启时退出。

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)
每个 *period* 毫秒允许注入客户机的最大熵字节数。使用 0 禁用限制（可能有风险）。

period=<integer> (*default* = 1000)
每隔 *period* 毫秒会重置熵注入配额，允许客户机再获取 *max_bytes* 的熵。

source=</dev/hwrng | /dev/random | /dev/urandom>
主机上用于收集熵的文件。使用 urandom 在实际意义上*不会*降低安全性，因为它仍由真实
可用于从主机直通硬件 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>]

将卷用作 SATA 硬盘或 CD-ROM（n 为 0 到 5）。

--nvme[n] [file=]<volume> [,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>] [,format=<enum>]
[,heads=<integer>] [,import-from=<source volume>] [,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>] [,size=<DiskSize>] [,snapshot=<1|0>]

将卷用作 NVME 硬盘或 CD-ROM（n 为 0 到 5）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。使用 STORAGE_ID:0 和 *import-from* 参数从现有卷导入。

aio=<io_uring | native | threads>

要使用的 AIO 类型。

backup=<boolean>

进行备份时是否应包含该驱动器。

bps=<bps>

最大读/写速度，单位为字节每秒。

bps_max_length=<seconds>

I/O burst 的最大长度，单位为秒。

bps_rd=<bps>

最大读取速度，单位为字节每秒。

bps_rd_max_length=<seconds>

读取 I/O burst 的最大长度，单位为秒。

bps_wr=<bps>

最大写入速度，单位为字节每秒。

bps_wr_max_length=<seconds>
写入 I/O burst 的最大长度，单位为秒。

cache=<directsync | none | unsafe | writeback | writethrough>
驱动器的缓存模式。

cyls=<integer>
强制驱动器物理几何参数使用指定柱面数。

detect_zeroes=<boolean>
控制是否检测并尝试优化零写入。

discard=<ignore | on>
控制是否将 discard/trim 请求传递到底层存储。

file=<volume>
驱动器的后端卷。

format=<cloop | qcow | qcow2 | qed | raw | vmdk>
驱动器后端文件的数据格式。

heads=<integer>
强制驱动器物理几何参数使用指定磁头数。

iops=<iops>
最大读/写 I/O，单位为操作每秒。

iops_max=<iops>
最大未限速读/写 I/O 池，单位为操作每秒。

iops_max_length=<seconds>
I/O burst 的最大长度，单位为秒。

iops_rd=<iops>
最大读取 I/O，单位为操作每秒。

iops_rd_max=<iops>
最大未限速读取 I/O 池，单位为操作每秒。

iops_rd_max_length=<seconds>
读取 I/O burst 的最大长度，单位为秒。

iops_wr=<iops>
最大写入 I/O，单位为操作每秒。

iops_wr_max=<iops>
最大未限速写入 I/O 池，单位为操作每秒。

iops_wr_max_length=<seconds>
写入 I/O burst 的最大长度，单位为秒。

mbps=<mbps>
最大读/写速度，单位为兆字节每秒。

mbps_max=<mbps>
最大未限速读/写池，单位为兆字节每秒。

mbps_rd=<mbps>
最大读取速度，单位为兆字节每秒。

mbps_rd_max=<mbps>
最大未限速读取池，单位为兆字节每秒。

mbps_wr=<mbps>
最大写入速度，单位为兆字节每秒。

mbps_wr_max=<mbps>
最大未限速写入池，单位为兆字节每秒。

media=<cdrom | disk> (*default* = disk)
驱动器的介质类型。

replicate=<boolean> (*default* = 1)
该驱动器是否应纳入复制任务。

rerror=<ignore | report | stop>
读取错误时的操作。

secs=<integer>
强制驱动器物理几何参数使用指定扇区数。

serial=<serial>
驱动器报告的序列号，经过 URL 编码，最长 20 字节。

shared=<boolean> (*default* = 0)
将此本地管理的卷标记为在所有节点上可用。

Warning
此选项不会自动共享该卷，而是假定它已经被共享。

size=<DiskSize>
磁盘大小。此项仅供信息展示，没有实际影响。

snapshot=<boolean>
控制 qemu 的 snapshot mode 功能。如果启用，对磁盘所做的更改是临时的，并会在虚拟

ssd=<boolean>
是否将此驱动器暴露为 SSD，而不是旋转式硬盘。

trans=<auto | lba | none>
强制磁盘几何参数 BIOS translation 模式。

werror=<enospc | ignore | report | stop>
写入错误时的操作。

wwn=<wwn>
驱动器的 worldwide name，编码为 16 字节十六进制字符串，并以 *0x* 为前缀。

**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>]
[,product=<product>] [,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>] [,vendor=<vendor>] [,werror=<enum>]
[,wwn=<wwn>]**

将卷用作 SCSI 硬盘或 CD-ROM（n 为 0 到 30）。

aio=<io_uring | native | threads>
要使用的 AIO 类型。

backup=<boolean>
进行备份时是否应包含该驱动器。

bps=<bps>
最大读/写速度，单位为字节每秒。

bps_max_length=<seconds>
I/O burst 的最大长度，单位为秒。

bps_rd=<bps>
最大读取速度，单位为字节每秒。

bps_rd_max_length=<seconds>
读取 I/O burst 的最大长度，单位为秒。

bps_wr=<bps>
最大写入速度，单位为字节每秒。

bps_wr_max_length=<seconds>
写入 I/O burst 的最大长度，单位为秒。

cache=<directsync | none | unsafe | writeback | writethrough>
驱动器的缓存模式。

cyls=<integer>
强制驱动器物理几何参数使用指定柱面数。

detect_zeroes=<boolean>
控制是否检测并尝试优化零写入。

discard=<ignore | on>
控制是否将 discard/trim 请求传递到底层存储。

file=<volume>
驱动器的后端卷。

format=<cloop | qcow | qcow2 | qed | raw | vmdk>
驱动器后端文件的数据格式。

heads=<integer>
强制驱动器物理几何参数使用指定磁头数。

iops=<iops>
最大读/写 I/O，单位为操作每秒。

iops_max=<iops>
最大未限速读/写 I/O 池，单位为操作每秒。

iops_max_length=<seconds>
I/O burst 的最大长度，单位为秒。

iops_rd=<iops>
最大读取 I/O，单位为操作每秒。

iops_rd_max=<iops>
最大未限速读取 I/O 池，单位为操作每秒。

iops_rd_max_length=<seconds>
读取 I/O burst 的最大长度，单位为秒。

iops_wr=<iops>
最大写入 I/O，单位为操作每秒。

iops_wr_max=<iops>
最大未限速写入 I/O 池，单位为操作每秒。

iops_wr_max_length=<seconds>
写入 I/O burst 的最大长度，单位为秒。

iothread=<boolean>
是否为此驱动器使用 iothreads。

mbps=<mbps>
最大读/写速度，单位为兆字节每秒。

mbps_max=<mbps>
最大未限速读/写池，单位为兆字节每秒。

mbps_rd=<mbps>
最大读取速度，单位为兆字节每秒。

mbps_rd_max=<mbps>
最大未限速读取池，单位为兆字节每秒。

mbps_wr=<mbps>
最大写入速度，单位为兆字节每秒。

mbps_wr_max=<mbps>
最大未限速写入池，单位为兆字节每秒。

media=<cdrom | disk> (*default* = disk)
驱动器的介质类型。

product=<product>
驱动器的 product 名称，最长 16 字节。

queues=<integer> (2 - N)
队列数量。

replicate=<boolean> (*default* = 1)
该驱动器是否应纳入复制任务。

rerror=<ignore | report | stop>
读取错误时的操作。

ro=<boolean>
驱动器是否为只读。

scsiblock=<boolean> (*default* = 0)
是否使用 scsi-block 对主机块设备进行完整直通。

Warning
与主机低内存或高内存碎片同时出现时，可能导致 I/O 错误。

secs=<integer>
强制驱动器物理几何参数使用指定扇区数。

serial=<serial>
驱动器报告的序列号，经过 URL 编码，最长 20 字节。

shared=<boolean> (*default* = 0)
将此本地管理的卷标记为在所有节点上可用。

Warning
此选项不会自动共享该卷，而是假定它已经被共享。

size=<DiskSize>
磁盘大小。此项仅供信息展示，没有实际影响。

snapshot=<boolean>
控制 qemu 的 snapshot mode 功能。如果启用，对磁盘所做的更改是临时的，并会在虚拟

ssd=<boolean>
是否将此驱动器暴露为 SSD，而不是旋转式硬盘。

trans=<auto | lba | none>
强制磁盘几何参数 BIOS translation 模式。

vendor=<vendor>
驱动器的 vendor 名称，最长 8 字节。

werror=<enospc | ignore | report | stop>
写入错误时的操作。

wwn=<wwn>
驱动器的 worldwide name，编码为 16 字节十六进制字符串，并以 *0x* 为前缀。

scsihw: <lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci | virtio-scsi-single> (*default* = lsi)

SCSI 控制器模型。

searchdomain: <string>

cloud-init：设置容器的 DNS 搜索域。创建时如果既未设置 searchdomain 也未设置 nameserver，则会自动使用主机上的设置。

serial[n]: (/dev/.+|socket)

在虚拟机内部创建串口设备（n 为 0 到 3），并直通 主机串口设备（例如 /dev/ttyS0），或在主机端创建 unix socket（使用 *qm terminal* 打开终端连接）。

Note
如果直通主机串口设备，此类机器将无法再迁移， 请特别谨慎使用。

Caution
实验性功能！已有用户报告此选项存在问题。

shares: <integer> (0 - 50000) (*default* = 1000)

auto-ballooning 的内存 shares 数量。数值越大，该虚拟机获得的内存越多。该数值相对于所有 auto-ballooning。auto-ballooning 由 pvestatd 执行。

smbios1: [base64=<1|0>] [,family=<Base64 encoded string>] [,manufacturer=<Base64 encoded string>] [,product=<Base64 encoded string>] [,serial=<Base64 encoded string>] [,sku=<Base64 encoded string>] [,uuid=<UUID>] [,version=<Base64 encoded string>]

指定 SMBIOS type 1 字段。

base64=<boolean>
用于指示 SMBIOS 值已进行 base64 编码的标志。

family=<Base64 encoded string>
设置 SMBIOS1 family 字符串。

manufacturer=<Base64 encoded string>
设置 SMBIOS1 manufacturer。

product=<Base64 encoded string>
设置 SMBIOS1 product ID。

serial=<Base64 encoded string>
设置 SMBIOS1 serial number。

sku=<Base64 encoded string>
设置 SMBIOS1 SKU 字符串。

uuid=<UUID>
设置 SMBIOS1 UUID。

version=<Base64 encoded string>
设置 SMBIOS1 version。

smp: <integer> (1 - N) (*default* = 1)
CPU 数量。请改用 -sockets 选项。

sockets: <integer> (1 - N) (*default* = 1)
CPU sockets 数量。

spice_enhancements: [foldersharing=<1|0>]
[,videostreaming=<off|all|filter>]
配置 SPICE 的其他增强功能。

foldersharing=<boolean> (*default* = 0)
通过 SPICE 启用文件夹共享。需要在虚拟机中安装 Spice-WebDAV 守护进程。

videostreaming=<all | filter | off> (*default* = off)
启用视频流。对检测到的视频流使用压缩。

sshkeys: <string>
cloud-init：设置 SSH 公钥（每行一个密钥，OpenSSH 格式）。

startdate: (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (*default* = now)
设置 real time clock 的初始日期。有效日期格式为：*now*、*2006-06-17T16:01:21* 或 *2006-06-17*。

startup: `[[order=]\d+] [,up=\d+] [,down=\d+]`
启动和关机行为。Order 是定义总体启动顺序的非负数。关机会按相反顺序执行。此外，可以设置 *up* 或 *down* 延迟（秒），用于指定启动或停止下一台虚拟机前等待的延迟。

tablet: <boolean> (*default* = 1)
启用/禁用 USB tablet 设备。通常需要此设备来允许 VNC 使用绝对鼠标定位。否则鼠标会与普通 VNC 客户端不同步。如果在一台主机上运行大量仅控制台客户机，可以考虑禁用此项以节省部分 spice（`qm set <vmid> --vga qxl`）时默认关闭。

tags: <string>
虚拟机标签。此项仅为元信息。

tdf: <boolean> (*default* = 0)
启用/禁用时间漂移修正。

template: <boolean> (*default* = 0)
启用/禁用模板。

tpmstate0: [file=]<volume> [,size=<DiskSize>]
[,version=<v1.2|v2.0>]
配置用于存储 TPM state 的磁盘。格式固定为 *raw*。

file=<volume>
驱动器的后端卷。

size=<DiskSize>
磁盘大小。此项仅供信息展示，没有实际影响。

version=<v1.2 | v2.0> (*default* = v1.2)
TPM 接口版本。v2.0 较新，应优先使用。请注意，此项之后无法更改。

unused[n]: [file=]<volume>
未使用卷的引用。此项供内部使用，不应手动修改。

file=<volume>
驱动器的后端卷。

usb[n]: [[host=]<HOSTUSBDEVICE|spice>] [,mapping=<mapping-id>] [,usb3=<1|0>]
配置 USB 设备（n 为 0 到 4；对于 machine version >= 7.1 且 ostype 为 l26 或 windows > 7 的客户机，n 最大可为 14）。

host=<HOSTUSBDEVICE|spice>
主机 USB 设备、端口，或值 *spice*。HOSTUSBDEVICE 语法为：

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

可以使用 *lsusb -t* 命令列出现有 USB 设备。

> **Note**
> 此选项允许直接访问主机硬件。因此此类机器将无法再迁移， 请特别谨慎使用。

值 *spice* 可用于为 spice 添加 USB 重定向设备。
此项和 *mapping* 键必须至少设置一个。

mapping=<mapping-id>
集群范围 mapping 的 ID。此项和默认键 *host* 必须至少设置一个。

usb3=<boolean> (*default* = 0)
指定给定的 host 选项是否为 USB3 设备或端口。对于现代客户机（machine version >= 7.1 且 ostype 为 l26 和 windows > 7），此标志无关紧要（所有设备都会插入 xhci 控制器）。

vcpus: <integer> (1 - N) (*default* = 0)
热插拔 vCPU 数量。

vga: [[type=]<enum>] [,clipboard=<vnc>] [,memory=<integer>]
配置 VGA 硬件。如果要使用高分辨率模式（>= 1280x1024x16），可能需要增加 vga memory 选项。自 QEMU 2.9 起，除某些使用 *cirrus* 的 Windows 版本（XP 及更早版本）外，所 OS 类型的默认 VGA 显示类型均为 *std*。*qxl* 选项会启用 SPICE 显示服务器。对于 win* OS，可以选择所需的独立显示器数量；Linux 客户机可以自行添加显示器。 也可以不使用任何显

clipboard=<vnc>
启用特定剪贴板。如果未设置，会根据显示类型添加 SPICE 剪贴板。尚不支持带 VNC 剪贴板的迁移。

memory=<integer> (4 - 512)
设置 VGA 内存（MiB）。对串口显示无效。

type=<cirrus | none | qxl | qxl2 | qxl3 | qxl4 | serial0 | serial1 | serial2 | serial3 | std | virtio | virtio-gl | vmware> (*default* = std)
选择 VGA 类型。不建议使用类型 *cirrus*。

**virtio[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>]
[,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,ro=<1|0>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,trans=<none|lba|auto>]
[,werror=<enum>]**

将卷用作 VIRTIO 硬盘（n 为 0 到 15）。

aio=<io_uring | native | threads>
要使用的 AIO 类型。

backup=<boolean>
进行备份时是否应包含该驱动器。

bps=<bps>
最大读/写速度，单位为字节每秒。

bps_max_length=<seconds>
I/O burst 的最大长度，单位为秒。

bps_rd=<bps>
最大读取速度，单位为字节每秒。

bps_rd_max_length=<seconds>
读取 I/O burst 的最大长度，单位为秒。

bps_wr=<bps>
最大写入速度，单位为字节每秒。

bps_wr_max_length=<seconds>
写入 I/O burst 的最大长度，单位为秒。

cache=<directsync | none | unsafe | writeback | writethrough>
驱动器的缓存模式。

cyls=<integer>
强制驱动器物理几何参数使用指定柱面数。

detect_zeroes=<boolean>
控制是否检测并尝试优化零写入。

discard=<ignore | on>
控制是否将 discard/trim 请求传递到底层存储。

file=<volume>
驱动器的后端卷。

format=<cloop | qcow | qcow2 | qed | raw | vmdk>
驱动器后端文件的数据格式。

heads=<integer>
强制驱动器物理几何参数使用指定磁头数。

iops=<iops>
最大读/写 I/O，单位为操作每秒。

iops_max=<iops>
最大未限速读/写 I/O 池，单位为操作每秒。

iops_max_length=<seconds>
I/O burst 的最大长度，单位为秒。

iops_rd=<iops>
最大读取 I/O，单位为操作每秒。

iops_rd_max=<iops>
最大未限速读取 I/O 池，单位为操作每秒。

iops_rd_max_length=<seconds>
读取 I/O burst 的最大长度，单位为秒。

iops_wr=<iops>
最大写入 I/O，单位为操作每秒。

iops_wr_max=<iops>
最大未限速写入 I/O 池，单位为操作每秒。

iops_wr_max_length=<seconds>
写入 I/O burst 的最大长度，单位为秒。

iothread=<boolean>
是否为此驱动器使用 iothreads。

mbps=<mbps>
最大读/写速度，单位为兆字节每秒。

mbps_max=<mbps>
最大未限速读/写池，单位为兆字节每秒。

mbps_rd=<mbps>
最大读取速度，单位为兆字节每秒。

mbps_rd_max=<mbps>
最大未限速读取池，单位为兆字节每秒。

mbps_wr=<mbps>
最大写入速度，单位为兆字节每秒。

mbps_wr_max=<mbps>
最大未限速写入池，单位为兆字节每秒。

media=<cdrom | disk> (*default* = disk)
驱动器的介质类型。

replicate=<boolean> (*default* = 1)
该驱动器是否应纳入复制任务。

rerror=<ignore | report | stop>
读取错误时的操作。

ro=<boolean>
驱动器是否为只读。

secs=<integer>
强制驱动器物理几何参数使用指定扇区数。

serial=<serial>
驱动器报告的序列号，经过 URL 编码，最长 20 字节。

shared=<boolean> (*default* = 0)
将此本地管理的卷标记为在所有节点上可用。

Warning
此选项不会自动共享该卷，而是假定它已经被共享。

size=<DiskSize>
磁盘大小。此项仅供信息展示，没有实际影响。

snapshot=<boolean>
控制 qemu 的 snapshot mode 功能。如果启用，对磁盘所做的更改是临时的，并会在虚拟

trans=<auto | lba | none>
强制磁盘几何参数 BIOS translation 模式。

werror=<enospc | ignore | report | stop>
写入错误时的操作。

virtiofs[n]: [dirid=]<mapping-id> [,cache=<enum>] [,direct-io=<1|0>] [,expose-acl=<1|0>] [,expose-xattr=<1|0>]
使用 Virtio-fs 在主机与客户机之间共享目录的配置。

cache=<always | auto | metadata | never> (*default* = auto)
文件系统应使用的缓存策略（auto、always、metadata、never）。

direct-io=<boolean> (*default* = 0)
遵循客户机应用程序传递下来的 O_DIRECT 标志。

dirid=<mapping-id>
要与客户机共享的目录 mapping 的标识符。也会用作虚拟机内部的 mount tag。

expose-acl=<boolean> (*default* = 0)
为此挂载启用 POSIX ACL 支持（启用 ACL 意味着启用 xattr）。

expose-xattr=<boolean> (*default* = 0)
为此挂载启用扩展属性支持。

vmgenid: <UUID> (*default* = 1 (autogenerated))
VM generation ID（vmgenid）设备会向客户机操作系统暴露一个 128 位整数值标识符。当虚拟
dirty、重新初始化随机数生成器等。 请注意，自动创建仅在通过 API/CLI 创建或更新方法执行时

vmstatestorage: <storage ID>
VM state 卷/文件的默认存储。

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

创建虚拟硬件 watchdog 设备。一旦由客户机操作启用，客户机内的 agent 必须定期轮询 watchdog，否则 watchdog 会重置客户机（或执行指定的相应操作）。

action=<debug | none | pause | poweroff | reset | shutdown>

激活后如果客户机未能及时轮询 watchdog，要执行的操作。

model=<i6300esb | ib700> (*default* = i6300esb)

要模拟的 watchdog 类型。

10.15 锁定

在线迁移、快照和备份（vzdump）会设置锁，以防止对受影响 VM 执行不兼容的并发操作。 有时需要手

```
# qm unlock <vmid>
```

Caution

只有在确定设置该锁的操作已不再运行时，才执行此操作。

Chapter 11

Proxmox 容器工具包

容器是完全虚拟化机器（VM）的轻量级替代方案。它们使用所在主机系统的内核，而不是模拟完整操作

容器的运行时开销很低，通常可以忽略不计。但也需要考虑一些缺点：

- Proxmox 容器中只能运行 Linux 发行版。无法在容器内运行其他操作系统，例如 FreeBSD 或 Microsoft Windows。
- 出于安全原因，需要限制对主机资源的访问。因此，容器运行在各自独立的 namespace 中。此外，syscall（用户空间对 Linux 内核的请求）。

Proxmox VE 使用 Linux Containers (LXC) 作为底层容器技术。“Proxmox Container Toolkit”（pct LXC 的使用和管理。

容器与 Proxmox VE 紧密集成。这意味着它们感知集群设置，并且可以与虚拟机使用相同的网络和存储 Proxmox VE 防火墙，或通过 HA 框架管理容器。

主要目标是提供一种既具备虚拟机优势、又没有额外开销的环境。这意味着 Proxmox 容器可归类为‘系

Note
如果要运行应用容器，例如 *Docker* 镜像，建议在 Proxmox QEMU 虚拟机内运行它们。这样既可以获得应用容器化的所有优势，也能获得虚拟机提供的优势，例如与主

11.1 技术概览

- LXC (https://linuxcontainers.org/)
- 集成到 Proxmox VE 图形化 Web 用户界面（GUI）
- 易用的命令行工具 `pct`
- 可通过 Proxmox VE REST API 访问
- 使用 *lxcfs* 提供容器化的 /proc 文件系统
- 使用 control groups（*cgroups*）进行资源隔离和限制

- 使用 *AppArmor* 和 *seccomp* 提升安全性
- 现代 Linux 内核
- 基于镜像的部署（模板）
- 使用 Proxmox VE 存储库
- 从主机配置容器（网络、DNS、存储等）

11.2 支持的发行版

官方支持的发行版列表如下。
以下发行版的模板可通过我们的软件仓库获取。可以使用 pveam 工具或图形化用户界面下载它们。

11.2.1 Alpine Linux

> Alpine Linux 是基于 musl libc 和 busybox 的安全导向轻量级 Linux 发行版。
>
> — https://alpinelinux.org

当前支持版本请参见：
https://alpinelinux.org/releases/

11.2.2 Arch Linux

> Arch Linux 是一种轻量且灵活的 Linux® 发行版，追求 Keep It Simple。
>
> — https://archlinux.org/

Arch Linux 使用滚动发布模型，详情请参见其 wiki：
https://wiki.archlinux.org/title/Arch_Linux

11.2.3 CentOS, Almalinux, Rocky Linux

CentOS / CentOS Stream

> CentOS Linux 发行版是一个稳定、可预测、可管理且可复现的平台，源自 Red Hat Enterprise Linux（RHEL）的源码。
>
> — https://centos.org

当前支持版本请参见：
https://en.wikipedia.org/wiki/CentOS#End-of-support_schedule

Almalinux

> 一个开源、由社区拥有和治理、永久免费使用的企业级 Linux 发行版，专注于长期稳定性，提供稳
> OS 与 RHEL® 和 pre-Stream CentOS 保持 1:1 二进制兼容。
>
> — https://almalinux.org

当前支持版本请参见：

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

Rocky Linux

> Rocky Linux 是社区企业级操作系统，设计目标是在其下游伙伴转向之后，与美国顶级企业级
> Linux 发行版保持 100% bug-for-bug 兼容。
>
> — https://rockylinux.org

当前支持版本请参见：

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

11.2.4 Debian

> Debian 是由 Debian 项目开发和维护的自由操作系统。它是一个自由 Linux 发行版，包含数千个
>
> — https://www.debian.org/intro/index#software

当前支持版本请参见：

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

11.2.5 Devuan

> Devuan GNU+Linux 是不含 systemd 的 Debian 分支，通过避免不必要的耦合并确保
> Init Freedom，让用户重新掌控自己的系统。
>
> — https://www.devuan.org

当前支持版本请参见：

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

11.2.6 Fedora

> Fedora 为硬件、云和容器创建创新、自由且开源的平台，使软件开发者和社区成员能够为用户构
>
> — https://getfedora.org

当前支持版本请参见：

https://fedoraproject.org/wiki/Releases

11.2.7 Gentoo

> 一个高度灵活、基于源码的 Linux 发行版。
>
> — https://www.gentoo.org

Gentoo 使用滚动发布模型。

11.2.8 OpenSUSE

> 面向系统管理员、开发者和桌面用户的创作者之选。
>
> — https://www.opensuse.org

当前支持版本请参见：
https://get.opensuse.org/leap/

11.2.9 Ubuntu

> Ubuntu 是面向企业服务器、桌面、云和 IoT 的现代开源 Linux 操作系统。
>
> — https://ubuntu.com/

当前支持版本请参见：
https://wiki.ubuntu.com/Releases

11.3 容器镜像

容器镜像有时也称为“templates”或“appliances”，是包含运行容器所需全部内容的 `tar` 归档文件。

Proxmox VE 本身为 最常见的 Linux 发行版 提供多种基础模板。可以使用 GUI 或 `pveam`（Proxmox VE Appliance Manager 的缩写）命令行工具下载它们。此外，也可以下载 TurnKey Linux 容器模板。

可用模板列表会通过 *pve-daily-update* timer 每日更新。也可以执行以下命令手动触发更新：

```
# pveam update
```

要查看可用镜像列表，请运行：

```
# pveam available
```

可以通过指定感兴趣的 `section` 来限制这个较大的列表，例如基础 `system` 镜像：

列出可用 **system** 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
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
```

使用这类模板前，需要先将它们下载到某个存储中。如果不确定使用哪个存储，可以直接使用名为 `local` 的存储。对于集群安装，建议使用共享存储，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
```

现在可以使用该镜像创建容器，并可使用以下命令列出存储 `local` 上已下载的所有镜像：

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

> **Tip**
> 也可以使用 Proxmox VE Web 界面 GUI 下载、列出和删除容器模板。

`pct` 使用这些模板创建新容器，例如：

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

上面的命令展示了完整的 Proxmox VE 卷标识符。它们包含存储名称，且大多数其他 Proxmox VE 命令都可以使用它们。例如，稍后可以使用以下命令删除该镜像：

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

11.4 容器设置

11.4.1 常规设置

容器的常规设置包括：

- **Node**：容器将运行所在的物理服务器
- **CT ID**：此 Proxmox VE 安装中用于标识容器的唯一编号
- **Hostname**：容器主机名
- **Resource Pool**：容器和虚拟机的逻辑分组
- **Password**：容器 root 密码
- **SSH Public Key**：用于通过 SSH 连接 root 账户的公钥
- **Unprivileged container**：该选项允许在创建时选择创建特权容器还是非特权容器。

非特权容器

非特权容器使用名为 user namespaces 的新内核特性。容器内的 root UID 0 会映射到容器外的非特权
LXC 问题。LXC 团队认为非特权容器在设计上是安全的。

这是创建新容器时的默认选项。

> **Note**
> 如果容器使用 systemd 作为 init 系统，请注意容器内运行的 systemd 版本应等于或高于 220。

特权容器

容器安全性通过强制访问控制 *AppArmor* 限制、*seccomp* 过滤器和 Linux 内核 namespace 实现。LXC 团队认为这类容器不安全，并且不会将新的容器逃逸漏洞视为值得分配 CVE 并快速修复的安

11.4.2 CPU

可以使用 `cores` 选项限制容器内可见 CPU 数量。它通过 Linux *cpuset* cgroup（**c**ontrol **group**）实
内部的一个特殊任务会定期尝试将正在运行的容器分布到可用 CPU 上。 要查看已分配 CPU，请运行以下

```
# pct cpusets
 ---------------------
 102:               6 7
 105:      2 3 4 5
 108:  0 1
 ---------------------
```

容器直接使用主机内核。容器内所有任务都由主机 CPU 调度器处理。Proxmox VE 默认使用 Linux *CFS*（**C**ompletely **F**air **S**cheduler）调度器，该调度器提供额外的带宽控制选项。

cpulimit: 可以使用该选项进一步限制分配的 CPU 时间。请注意，这是浮点数，因此可以为容器分配两个核心，但将总体 CPU 消耗限制为半个核心。

```
cores: 2
cpulimit: 0.5
```

cpuunits: 这是传递给内核调度器的相对权重。数值越大，该容器获得的 CPU 时间越多。该数值相对于所有其他正在运行容器的权重。默认值为 100（如果主机使用 legacy cgroup v1，则为 1024）。可以使用该设置为某些容器赋予更高优先级。

11.4.3 内存

容器内存通过 cgroup memory controller 控制。

memory: 限制总体内存使用量。对应 memory.limit_in_bytes cgroup 设置。

swap: 允许容器从主机 swap 空间使用额外 swap 内存。对应 memory.memsw.limit_in_bytes cgroup 设置，该值设置为二者之和（memory + swap）。

11.4.4 挂载点

根挂载点通过 rootfs 属性配置。最多可以额外配置 256 个挂载点。对应选项名为 mp0 到 mp255，可

rootfs: [volume=]<volume> [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskSize>]

将卷用作容器根文件系统。各选项的详细说明见下文。

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>]

将卷用作容器挂载点。可使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。

acl=<boolean>

显式启用或禁用 ACL 支持。

backup=<boolean>

是否将该挂载点包含在备份中（仅用于卷挂载点）。

mountoptions=<opt[;opt...]>

rootfs/mps 的额外挂载选项。

mp=<Path>

从容器内部看到的挂载点路径。

Note
出于安全原因，不得包含任何符号链接。

quota=<boolean>
在容器内启用用户配额（zfs subvolume 不支持）。

replicate=<boolean> (*default* = 1)
会将该卷包含在存储复制任务中。

ro=<boolean>
只读挂载点。

shared=<boolean> (*default* = 0)
将此非卷挂载点标记为在所有节点上可用。

Warning
此选项不会自动共享该挂载点，而是假定它已经被共享！

size=<DiskSize>
卷大小（只读值）。

volume=<volume>
要挂载到容器中的卷、设备或目录。

当前有三类挂载点：由存储支持的挂载点、bind mount 和 device mount。

典型容器 **rootfs** 配置

```
rootfs: thin1:base-100-disk-1,size=8G
```

由存储支持的挂载点

由存储支持的挂载点由 Proxmox VE 存储子系统管理，并有三种不同形式：

- 基于镜像：这是包含单个 ext4 格式文件系统的 raw 镜像。
- ZFS 子卷：从技术上讲它们是 bind mount，但带有托管存储，因此允许调整大小和创建快照。
- 目录：传递 `size=0` 会触发特殊情况，即创建目录而不是 raw 镜像。

Note
对于由存储支持的挂载点卷，特殊选项语法 `STORAGE_ID:SIZE_IN_GB` 会在指定存储上自动分配指定大小的卷。例如，调用：

```
pct set 100 -mp0 thin1:10,mp=/path/in/container
```

会在存储 `thin1` 上分配一个 10GB 卷，并用已分配的卷 ID 替换卷 ID 占位符 `10`，然后在容器中的 `/path/in/container` 设置挂载点。

Bind 挂载点

Bind mount 允许在容器内访问 Proxmox VE 主机上的任意目录。一些可能的使用场景包括：

- 在客户机中访问主目录
- 在客户机中访问 USB 设备目录
- 在客户机中访问主机上的 NFS 挂载

Bind mount 被视为不受存储子系统管理，因此不能在容器内创建快照或处理 quota。对于非特权容器
ACL。

Note
使用 vzdump 时不会备份 bind mount 挂载点的内容。

Warning
出于安全原因，bind mount 只能使用专门为此保留的源目录建立，例如 /mnt/bindmounts 下的目录层级。绝不要将 /、/var 或 /etc 等系统目录 bind mount 到容器中，这会带来很大的安全风险。

Note
bind mount 源路径不得包含任何符号链接。

例如，要让 ID 为 100 的容器在路径 /shared 下访问目录 /mnt/bindmounts/shared，请添加如下配

```
mp0: /mnt/bindmounts/shared,mp=/shared
```

到 /etc/pve/lxc/100.conf 中。

或者使用 pct 工具：

```
pct set 100 -mp0 /mnt/bindmounts/shared,mp=/shared
```

实现相同结果。

设备挂载点

设备挂载点允许将主机块设备直接挂载到容器中。与 bind mount 类似，device mount 不受 Prox-
mox VE 存储子系统管理，但会遵循 quota 和 acl 选项。

Note
设备挂载点只应在特殊情况下使用。多数情况下，由存储支持的挂载点能提供相同性能以及更多功能。

Note
使用 vzdump 时不会备份设备挂载点的内容。

11.4.5 网络

单个容器最多可以配置 10 个网络接口。对应选项名为 net0 到 net9，可包含以下设置：

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)>] [,link_down=<1|0>]
[,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>]
[,trunks=<vlanid[;vlanid...]>] [,type=<veth>]

指定容器的网络接口。

bridge=<bridge>
要连接该网络设备的网桥。

firewall=<boolean>
控制是否使用此接口的防火墙规则。

gw=<GatewayIPv4>
IPv4 流量的默认网关。

gw6=<GatewayIPv6>
IPv6 流量的默认网关。

hwaddr=<XX:XX:XX:XX:XX:XX>
未设置 I/G（Individual/Group）位的普通 MAC 地址。

ip=<(IPv4/CIDR|dhcp|manual)>
CIDR 格式的 IPv4 地址。

ip6=<(IPv6/CIDR|auto|dhcp|manual)>
CIDR 格式的 IPv6 地址。

link_down=<boolean>
此接口是否应断开连接（类似拔掉网线）。

mtu=<integer> (64 - 65535)
接口的最大传输单元。 (lxc.network.mtu)

name=<string>
从容器内部看到的网络设备名称。 (lxc.network.name)

rate=<mbps>
对该接口应用速率限制。

tag=<integer> (1 - 4094)
此接口的 VLAN 标签。

trunks=<vlanid[;vlanid...]>
允许通过此接口的 VLAN ID。

type=<veth>
网络接口类型。

11.4.6 容器的自动启动和关闭

要在主机系统启动时自动启动容器，可以在 Web 界面中容器的 *Options* 面板选择 *Start at boot* 选项，或运行以下命令：

```
# pct set CTID -onboot 1
```

启动和关闭顺序

如果要微调容器启动顺序，可以使用以下参数：

- **Start/Shutdown order**：定义启动顺序优先级。例如，如果希望 CT 最先启动，可将其设置为 1。（关闭时使用反向启动顺序，因此启动顺序为 1 的容器会最后关闭）
- **Startup delay**：定义该容器启动与后续容器启动之间的间隔。例如，如果希望等待 240 秒后再启
  240。
- **Shutdown timeout**：定义 Proxmox VE 在发出关闭命令后等待容器离线的秒数。默认值为 60，这意味着 Proxmox VE 会发出关闭请求，等待 60s 让机器离线；如果 60s 后机器仍在线，则通

请注意，未设置启动/关闭顺序参数的容器始终会在已设置该参数的容器之后启动，并且该参数只对同一

如果需要在主机启动与第一个容器启动之间加入延迟，请参见 Proxmox VE 节点管理 章节。

11.4.7 Hook 脚本

可以通过配置属性 `hookscript` 为 CT 添加 hook script。

```
# pct set 100 -hookscript local:snippets/hookscript.pl
```

它会在客户机生命周期的不同阶段被调用。示例和文档请参见 `/usr/share/pve-docs/examples/g`
下的示例脚本。

11.5 安全注意事项

容器使用主机系统内核。这会向恶意用户暴露攻击面。通常，完整虚拟机提供更好的隔离。如果向未知

为了降低攻击面，LXC 使用 AppArmor、CGroups 和内核 namespace 等多种安全特性。

11.5.1 AppArmor

AppArmor profile 用于限制对潜在危险操作的访问。某些系统调用（例如 `mount`）会被禁止执行。

要跟踪 AppArmor 活动，请使用：

```
# dmesg | grep apparmor
```

虽然不建议这样做，但可以为容器禁用 AppArmor。这会带来安全风险。如果系统配置错误，或存在 LXC/Linux 内核漏洞，某些 syscall 在容器内执行时可能导致权限提升。

要为容器禁用 AppArmor，请在位于 `/etc/pve/lxc/CTID.conf` 的容器配置文件中添加以下行：

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

Warning
请注意，不建议在生产环境中这样做。

11.5.2 Control Groups（*cgroup*）

cgroup 是一种内核机制，用于按层级组织进程并分配系统资源。

通过 *cgroups* 控制的主要资源包括 CPU 时间、内存和 swap 限制，以及对设备节点的访问。*cgroups* 也用于在创建快照前 "freeze" 容器。

当前有 2 个可用的 *cgroups* 版本： legacy and *cgroupv2*.

自 Proxmox VE 7.0 起，默认环境为纯 *cgroupv2*。此前使用 "hybrid" 设置，资源控制主要在 *cgroupv1* 中完成，并额外使用一个 *cgroupv2* controller；该 controller 可通过 *cgroup_no_v1* 内核命令行参数接管部分子系统。（详情请参见 内核参数文档。）

CGroup 版本兼容性

就 Proxmox VE 而言，纯 *cgroupv2* 与旧 hybrid 环境的主要区别在于：使用 *cgroupv2* 时，内存和 swap 现在可以独立控制。容器的内存和 swap 设置可以直接映射到这些值；而此前只能限制内存上限以及 swap *总和*的上限。

另一个重要区别是 *devices* controller 的配置方式完全不同。因此，纯 *cgroupv2* 环境目前不支持文件 quota。

要在纯 *cgroupv2* 环境中运行，容器操作系统需要支持 *cgroupv2*。运行 *systemd* 231 或更新版本的容器 *cgroupv2* [1]；不使用 *systemd* 作为 init 系统的容器也支持 [2]。

[1] this includes all newest major versions of container templates shipped by Proxmox VE
[2] for example Alpine Linux

Note
CentOS 7 和 Ubuntu 16.10 是两个典型的 Linux 发行版版本，其 *systemd* 版本过旧，无法在 *cgroupv2* 环境中运行。可以选择：

- 将整个发行版升级到较新版本。对于上述示例，可以是 Ubuntu 18.04 或 20.04，以及 CentOS 8（或 AlmaLinux、Rocky Linux 等 RHEL/CentOS 衍生版）。这样可以获得最新 bug 和安全修复，通常也能获得新功能，并将 EOL 日期推后。
- 升级容器的 systemd 版本。如果发行版提供 backports 仓库，这可以作为一种简单快速的临时措
- 将容器或其中的服务迁移到虚拟机。虚拟机与主机的交互少得多，因此可以在其中正常安装年代久 OS 版本。
- 切换回 legacy *cgroup* controller。请注意，虽然这可能是有效方案，但不是永久方案。从 Proxmox VE 9.0 起，将不再支持 legacy controller。

更改 **CGroup** 版本

Tip
如果不需要文件系统 quota，且所有容器都支持 *cgroupv2*，建议保持使用新的默认设置。

要切换回先前版本，可以使用以下内核命令行参数：

```
systemd.unified_cgroup_hierarchy=0
```

关于应在何处添加该参数，请参见 本节 中关于编辑内核启动命令行的说明。

11.6 客户机操作系统配置

Proxmox VE 会尝试检测容器中的 Linux 发行版，并修改一些文件。容器启动时会执行的事项简要列表

set /etc/hostname
: 设置容器名称

modify /etc/hosts
: 允许解析本地主机名

network setup
: 将完整网络设置传递给容器

configure DNS
: 传递 DNS 服务器信息

adapt the init system
: 例如，修正生成的 getty 进程数量

set the root password
: 创建新容器时设置 root 密码

rewrite ssh_host_keys
: 确保每个容器拥有唯一密钥

randomize crontab
: 避免所有容器中的 cron 同时启动

Proxmox VE 所做更改会被注释标记包围：

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

这些标记会插入到文件中的合理位置。如果此类段落已存在，则会就地更新且不会移动。

可以通过为某个文件添加 `.pve-ignore.` 文件来阻止修改。例如，如果文件 `/etc/.pve-ignore.ho` 存在，则不会触碰 `/etc/hosts` 文件。可以通过以下命令创建一个简单空文件：

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

大多数修改依赖 OS，因此会因发行版和版本不同而不同。可以通过手动将 `ostype` 设置为 `unmanaged` 完全禁用修改。

OS 类型检测通过检查容器内的特定文件完成。Proxmox VE 首先检查 `/etc/os-release` 文件[3]。如果该文件不存在，或不包含清晰可识别的发行版标识符，则检查以下发行版专用 release 文件。

Ubuntu
: 检查 /etc/lsb-release（`DISTRIB_ID=Ubuntu`）

Debian
: 测试 /etc/debian_version

Fedora
: 测试 /etc/fedora-release

RedHat 或 CentOS
: 测试 /etc/redhat-release

ArchLinux
: 测试 /etc/arch-release

Alpine
: 测试 /etc/alpine-release

Gentoo
: 测试 /etc/gentoo-release

Note
如果配置的 `ostype` 与自动检测到的类型不同，容器启动会失败。

---

[3] /etc/os-release replaces the multitude of per-distribution release files https://manpages.debian.org/stable/systemd/os-release.5.en.html

11.7 容器存储

Proxmox VE LXC 容器存储模型比传统容器存储模型更灵活。一个容器可以有多个挂载点。这使得每个

例如，容器的根文件系统可以位于较慢且廉价的存储上，而数据库可以通过第二个挂载点放在快速的分
挂载点 章节。

可以使用 Proxmox VE 存储库支持的任何存储类型。这意味着容器可以存储在本地存储（例如 lvm、z
或目录）、共享外部存储（如 iSCSI、NFS），甚至 Ceph 等分布式存储系统上。如果底层存储支持快
备份工具可以使用快照提供一致的容器备份。

此外，本地设备或本地目录可以使用 *bind mounts* 直接挂载。这使容器能够以几乎零开销访问本地资源
mount 也可作为容器之间共享数据的简单方式。

11.7.1 FUSE 挂载

Warning
由于 Linux 内核 freezer 子系统存在问题，强烈不建议在容器内使用 FUSE 挂载，因为在 suspend 或 snapshot 模式备份时需要冻结容器。

如果无法用其他挂载机制或存储技术替代 FUSE 挂载，可以在 Proxmox 主机上建立 FUSE 挂载，并使用
bind mount 挂载点使其在容器内可访问。

11.7.2 在容器内使用 Quota

Quota 允许在容器内限制每个用户可使用的磁盘空间量。

Note
这目前需要使用 legacy *cgroups*。

Note
这只适用于基于 ext4 镜像的存储类型，并且目前只适用于特权容器。

启用 quota 选项会使挂载点使用以下挂载选项：`usrjquota=aquota.user,grpjquota=aquota.`

这允许像在其他系统上一样使用 quota。可以运行以下命令初始化 `/aquota.user` 和 `/aquota.grou`
文件：

```
# quotacheck -cmug /
# quotaon /
```

然后使用 edquota 命令编辑 quota。详情请参考容器内所运行发行版的文档。

Note
需要为每个挂载点运行上述命令，并传入挂载点路径，而不是只传入 /。

11.7.3 在容器内使用 ACL

标准 Posix **A**ccess **C**ontrol **L**ists 在容器内同样可用。ACL 允许设置比传统 user/group/others 模型更细粒度的文件所有权。

11.7.4 容器挂载点备份

要将挂载点包含在备份中，请在容器配置中为其启用 backup 选项。对于现有挂载点 mp0：

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

添加 backup=1 以启用：

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

> **Note**
> 在 GUI 中创建新挂载点时，该选项默认启用。

要为挂载点禁用备份，请按上述方式添加 backup=0，或在 GUI 中取消勾选 **Backup** 复选框。

11.7.5 容器挂载点复制

默认情况下，复制 Root Disk 时也会复制额外挂载点。如果希望 Proxmox VE 存储复制机制跳过某个挂
Skip replication 选项。从 Proxmox VE 5.0 起，复制要求使用 zfspool 类型的存储。当容器已配置
Skip replication。

11.8 备份和还原

11.8.1 容器备份

可以使用 vzdump 工具备份容器。详情请参见 vzdump manual page。

11.8.2 还原容器备份

可以使用 pct restore 命令还原由 vzdump 创建的容器备份。默认情况下，pct restore 会尽量还
pct manual page）。

> **Note**
> 可以使用 pvesm extractconfig 查看 vzdump 归档中包含的备份配置。

有两种基本还原模式，区别仅在于对挂载点的处理：

“简单” 还原模式

如果既未显式设置 rootfs 参数，也未显式设置任何可选 mpX 参数，则会按以下步骤从备份的配置文件

1. 从备份中提取挂载点及其选项
2. 在 storage 参数提供的存储（默认：local）上，为由存储支持的挂载点创建卷。
3. 从备份归档中提取文件
4. 将 bind 和 device 挂载点添加到还原后的配置（仅限 root 用户）

Note
由于 bind 和 device 挂载点从不备份，因此最后一步不会还原任何文件，只会还原配置选项。其假设
bind mount 到多个容器中的 NFS 空间），要么根本不打算备份。

Web 界面中的容器还原操作也使用该简单模式。

“高级” 还原模式

通过设置 rootfs 参数（以及可选的任意 mpX 参数组合），pct restore 命令会自动切换到高级模式
rootfs 和 mpX 配置选项，而只使用作为参数显式提供的选项。
该模式允许在还原时灵活配置挂载点设置，例如：

- 为每个挂载点单独设置目标存储、卷大小和其他选项
- 根据新的挂载点方案重新分布备份文件
- 还原到 device 和/或 bind 挂载点（仅限 root 用户）

11.9 使用 pct 管理容器

“Proxmox Container Toolkit”（pct）是用于管理 Proxmox VE 容器的命令行工具。它可用于创建或

11.9.1 CLI 使用示例

基于 Debian 模板创建容器（假设已通过 Web 界面下载该模板）：

```
# pct create 100 /var/lib/vz/template/cache/debian-10.0-standard_10.0-1 ↩
   _amd64.tar.gz
```

启动容器 100：

```
# pct start 100
```

通过 getty 启动登录会话：

```
# pct console 100
```

进入 LXC namespace，并以 root 用户运行 shell：

```
# pct enter 100
```

显示配置：

```
# pct config 100
```

在容器运行时，添加名为 eth0 的网络接口，将其桥接到主机网桥 vmbr0，并设置地址和网关：

```
# pct set 100 -net0 name=eth0,bridge=vmbr0,ip=192.168.15.147/24,gw ↩
   =192.168.15.1
```

将容器内存减少到 512MB：

```
# pct set 100 -memory 512
```

销毁容器始终会将其从 Access Control Lists 中移除，并始终移除该容器的防火墙配置。如果还要额外从 HA 资源配置中移除，必须启用 *--purge*。

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

将挂载点卷移动到其他存储。

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

将卷重新分配给其他 CT。这会从源 CT 移除卷 mp0，并将其作为 mp1 附加到目标 CT。在后台，该卷会

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

11.9.2 常用命令示例

列出当前节点上的容器：

```
# pct list
```

查看容器状态并显示配置：

```
# pct status 100
# pct config 100
```

11.9.3 获取调试日志

如果 pct start 无法启动某个特定容器，可以通过传递 --debug 标志收集调试输出（将 CTID 替换为容器的 CTID）：

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

或者，可以使用以下 lxc-start 命令，该命令会将调试日志保存到 -o 输出选项指定的文件：

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

该命令会尝试以前台模式启动容器。要停止容器，请在第二个终端中运行 `pct shutdown CTID` 或 `pct stop CTID`。

收集到的调试日志会写入 `/tmp/lxc-CTID.log`。

> **Note**
> 如果自上次尝试使用 `pct start` 启动以来修改过容器配置，则需要至少运行一次 `pct start`，以同时更新 `lxc-start` 使用的配置。

11.10 迁移

如果有集群，可以使用以下命令迁移容器：

```
# pct migrate <ctid> <target>
```

只要容器处于离线状态，该命令即可工作。如果容器定义了本地卷或挂载点，并且目标主机上定义了相

由于技术限制，正在运行的容器无法在线迁移。可以执行 restart migration：先关闭容器，将其移动，

可以通过 Web 界面执行 restart migration，也可以在 `pct migrate` 命令中使用 `--restart` 标志。

restart migration 会关闭容器，并在指定超时时间后终止它（默认 180 秒）。然后会像离线迁移一样

11.11 配置

`/etc/pve/lxc/<CTID>.conf` 文件保存容器配置，其中 <CTID> 是给定容器的数字 ID。与存储在 `/etc/pve/` 中的所有其他文件一样，它们会自动复制到所有其他集群节点。

> **Note**
> CTID < 100 保留用于内部用途，且 CTID 需要在整个集群范围内唯一。

容器配置示例

```
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
```

配置文件是简单文本文件。可以使用普通文本编辑器编辑，例如 vi 或 nano。这有时适合进行小幅修正

因此，通常最好使用 pct 命令生成和修改这些文件，或通过 GUI 完成整个操作。该工具足够智能，可以“hot plug”，在这种情况下无需重启容器。

如果更改无法热插拔应用，则会注册为待处理更改（在 GUI 中以红色显示）。这些更改只有在重启容器

11.11.1 文件格式

容器配置文件使用简单的冒号分隔 key/value 格式。每行格式如下：

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

这些文件中的空行会被忽略，以 # 字符开头的行会被视为注释并同样忽略。

可以直接添加低层 LXC 风格配置，例如：

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

or

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

这些设置会直接传递给 LXC 低层工具。

11.11.2 快照

创建快照时，pct 会将快照时刻的配置存储到同一配置文件中的独立快照段。例如，创建名为 “test-snapshot” 的快照后，配置文件会类似如下：

带快照的容器配置

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

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

有一些与快照相关的属性，例如 parent 和 snaptime。parent 属性用于存储快照之间的父/子关系。是快照创建时间戳（Unix epoch）。

11.11.3 选项

arch: <amd64 | arm64 | armhf | i386 | riscv32 | riscv64> (*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，console 命令会尝试连接到一个可用的 tty 设备。将 cmode 设置为 console 时，它会改为尝试附加到 /dev/console。如果将 cmode 设置为 shell，则会直接在 shell（不登录）。

console: <boolean> (*default* = 1)
为容器附加一个控制台设备（/dev/console）。

cores: <integer> (1 - 8192)
分配给容器的核心数量。默认情况下，容器可以使用所有可用核心。

cpulimit: <number> (0 - 8192) (*default* = 0)
CPU 使用限制。

> **Note**
> 如果计算机有 2 个 CPU，则总共有 2 份 CPU 时间。值 0 表示不限制 CPU。

cpuunits: <integer> (0 - 500000) (*default* = cgroup v1: 1024, cgroup v2: 100)
容器的 CPU 权重。该参数用于内核公平调度器。数值越大，该容器获得的 CPU 时间越多。该数值相对于所有其他正在运行的来宾的权重。

debug: <boolean> (*default* = 0)
尝试输出更详细的信息。目前这只会在启动时启用 debug 日志级别。

description: <string>
容器的描述。显示在 Web 界面中 CT 的概要页。该内容会作为注释保存在配置文件中。

dev[n]: [[path=]<Path>] [,deny-write=<1|0>] [,gid=<integer>] [,mode=<Octal access mode>] [,uid=<integer>]
要直通给容器的设备。

deny-write=<boolean> (*default* = 0)
禁止容器写入该设备。

gid=<integer> (0 - N)
要分配给设备节点的组 ID。

mode=<Octal access mode>
要在设备节点上设置的访问模式。

path=<Path>
要直通给容器的设备路径。

uid=<integer> (0 - N)
要分配给设备节点的用户 ID。

features: [force_rw_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>] [,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>]
允许容器访问高级功能。

force_rw_sys=<boolean> (*default* = 0)
在非特权容器中将 /sys 挂载为 rw，而不是 mixed。这可能会破坏较新版（>= v245）sys
network 下的网络功能。

fuse=<boolean> (*default* = 0)
允许在容器中使用 fuse 文件系统。请注意，fuse 与 freezer cgroup 之间的交互 可能导致
I/O 死锁。

keyctl=<boolean> (*default* = 0)
仅用于非特权容器：允许使用 keyctl() 系统调用。在容器内使用 docker 需要此功能。默认情况下，非特权容器会看到该系统调用不存在。这主要是针对 systemd-networkd 的 变通措施，因为当部分 keyctl() 操作因权限不足而被内核拒绝时，它会将其视为致命错误本质上，需要在运行 systemd-networkd 和 docker 之间做出选择。

mknod=<boolean> (*default* = 0)
允许非特权容器使用 mknod() 添加某些设备节点。这需要内核支持 seccomp trap 到用户空（5.3 或更新版本）。此功能为实验性。

mount=<fstype;fstype;...>
允许挂载特定类型的文件系统。该值应为 mount 命令所使用的文件系统类型列表。请注意，这可能对容器安全性产生负面影响。如果可以访问 loop 设备，挂载文件可能绕过 devices cgroup 的 mknod 权限；挂载 NFS 文件系统可能完全阻塞主机 I/O 并阻止其重启，等

nesting=<boolean> (*default* = 0)
允许嵌套。最好与带有额外 ID 映射的非特权容器一起使用。请注意，这会将主机的 procfs 和 sysfs 内容暴露给来宾。

hookscript: <string>
在容器生命周期的多个阶段执行的脚本。

hostname: <string>
设置容器的主机名。

lock: <backup | create | destroyed | disk | fstrim | migrate | mounted | rollback | snapshot | snapshot-delete>
锁定或解锁容器。

memory: <integer> (16 - N) (*default* = 512)
容器的 RAM 容量，单位为 MB。

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>]
将卷用作容器挂载点。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 可分配新卷。

acl=<boolean>
显式启用或禁用 ACL 支持。

backup=<boolean>
是否将该挂载点包含在备份中（仅用于卷挂载点）。

mountoptions=<opt[;opt...]>
rootfs/mps 的额外挂载选项。

mp=<Path>
从容器内部看到的挂载点路径。

> **Note**
> 出于安全原因，不得包含任何符号链接。

quota=<boolean>
在容器内启用用户配额（zfs 子卷不支持）。

replicate=<boolean> (*default* = 1)
会将该卷包含到存储复制作业中。

ro=<boolean>
只读挂载点。

shared=<boolean> (*default* = 0)
将此非卷挂载点标记为在所有节点上可用。

Warning
此选项不会自动共享挂载点，而是假定它已经被共享。

size=<DiskSize>
卷大小（只读值）。

volume=<volume>
要挂载到容器中的卷、设备或目录。

nameserver: <string>
设置容器的 DNS 服务器 IP 地址。如果既未设置 searchdomain 也未设置 nameserver，创建时会自动使用主机上的设置。

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)>] [,link_down=<1|0>]
[,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>]
[,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
指定容器的网络接口。

bridge=<bridge>
要附加网络设备的网桥。

firewall=<boolean>
控制是否使用此接口的防火墙规则。

gw=<GatewayIPv4>
IPv4 流量的默认网关。

gw6=<GatewayIPv6>
IPv6 流量的默认网关。

hwaddr=<XX:XX:XX:XX:XX:XX>
常规 MAC 地址，未设置 I/G（Individual/Group）位。

ip=<(IPv4/CIDR|dhcp|manual)>
CIDR 格式的 IPv4 地址。

ip6=<(IPv6/CIDR|auto|dhcp|manual)>
CIDR 格式的 IPv6 地址。

link_down=<boolean>
是否断开该接口（类似拔掉网线）。

mtu=<integer> (64 - 65535)
接口的最大传输单元。（lxc.network.mtu）

name=<string>
从容器内部看到的网络设备名称。（lxc.network.name）

rate=<mbps>
对该接口应用速率限制。

tag=<integer> (1 - 4094)
此接口的 VLAN 标签。

trunks=<vlanid[;vlanid...]>
允许通过该接口的 VLAN ID。

type=<veth>
网络接口类型。

onboot: <boolean> (*default* = 0)
指定容器是否在系统启动期间自动启动。

ostype: <alpine | archlinux | centos | debian | devuan | fedora | gentoo | nixos | opensuse | ubuntu | unmanaged>
操作系统类型。该值用于在容器内设置配置，并对应 /usr/share/lxc/config/<ostype>.commo
中的 lxc 设置脚本。值 unmanaged 可用于跳过特定于操作系统的设置。

protection: <boolean> (*default* = 0)
设置容器的保护标志。这会阻止对 CT 或 CT 磁盘执行移除/更新操作。

rootfs: [volume=]<volume> [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskSize>]
将卷用作容器根文件系统。

acl=<boolean>
显式启用或禁用 ACL 支持。

mountoptions=<opt[;opt...]>
rootfs/mps 的额外挂载选项。

quota=<boolean>
在容器内启用用户配额（zfs 子卷不支持）。

replicate=<boolean> (*default* = 1)
会将该卷包含到存储复制作业中。

ro=<boolean>
只读挂载点。

shared=<boolean> (*default* = 0)
将此非卷挂载点标记为在所有节点上可用。

Warning
此选项不会自动共享挂载点，而是假定它已经被共享。

size=<DiskSize>
卷大小（只读值）。

volume=<volume>
要挂载到容器中的卷、设备或目录。

searchdomain: <string>
设置容器的 DNS 搜索域。如果既未设置 searchdomain 也未设置 nameserver，创建时会自动

startup: `[[order=]\d+] [,up=\d+] [,down=\d+]`
启动和关闭行为。Order 是一个非负数，用于定义总体启动顺序。关闭时按相反顺序执行。
此外，可以设置以秒为单位的 up 或 down 延迟，用于指定启动或停止下一个虚拟机前等待的延迟。

swap: <integer> (0 - N) (*default* = 512)
容器的 SWAP 容量，单位为 MB。

tags: <string>
容器的标签。这只是元信息。

template: <boolean> (*default* = 0)
启用或禁用模板。

timezone: <string>
容器中使用的时区。如果未设置该选项，则不会执行任何操作。可设置为 host 以匹配主机时区，
也可以设置为 /usr/share/zoneinfo/zone.tab 中的任意时区选项。

tty: <integer> (0 - 6) (*default* = 2)
指定容器可用的 tty 数量。

unprivileged: <boolean> (*default* = 0)
使容器以非特权用户身份运行。（不应手动修改。）

unused[n]: [volume=]<volume>
对未使用卷的引用。该项供内部使用，不应手动修改。

volume=<volume>
当前未使用的卷。

11.12 锁

容器迁移、快照和备份（vzdump）会设置锁，以防止对受影响容器执行不兼容的并发操作。有时需要手

```
# pct unlock <CTID>
```

Caution
只有在确定设置该锁的操作已不再运行时，才执行此操作。

Chapter 12

软件定义网络

Proxmox VE 中的软件定义网络（**S**oftware-**D**efined **N**etwork，SDN） 功能可用于创建虚拟区域和虚
环境的搭建。

12.1 简介

Proxmox VE SDN 通过灵活的软件控制配置，实现客户机虚拟网络的隔离和细粒度控制。

隔离通过*区域*（zone）、虚拟网络（**VNet**）和*子网*进行管理。区域是一个 独立的虚拟隔离网络范
是隶属于某个区域的虚拟网络。子网是 VNet 内部的 IP 地址范围。

根据区域类型不同，网络行为也会不同，并提供各自特定的功能、优势和限制。

SDN 的使用场景范围很广，从每个独立节点上的隔离私有网络，到跨多个不同位置 PVE 集群的复杂覆盖

在集群范围的数据中心 SDN 管理界面中配置 VNet 后，每个节点本地都会出现一个 普通 Linux
网桥，可分配给虚拟机和容器使用。

12.2 支持状态

12.2.1 历史

Proxmox VE SDN 栈自 2019 年起作为实验性功能提供，并持续由众多开发者和用户改进、 测试。随着
Proxmox VE 6.2 中集成到 Web 界面，SDN 朝更广泛集成迈出了重要一步。 在 Proxmox VE 7
发布周期中，又加入了大量改进和功能。根据用户反馈可以看出，其基础 设计选择和实现已经相当可靠
SDN 栈标记为 ‘experimental’ 已经不能准确反映其状态。到 Proxmox VE 8，项目决定为 SDN
功能的完整集成奠定基础， 将网络和接口管理提升为 Proxmox VE 访问控制栈中的核心组件。在
Proxmox VE 8.1 中达成了 两个重要里程碑：首先，为 IP 地址管理（IPAM）功能加入了 DHCP
集成；其次， SDN 集成现在默认安装。

12.2.2 当前状态

当前 SDN 安装中各层的支持状态如下：

- 核心 SDN，包括 VNet 管理及其与 Proxmox VE 栈的集成，已获得完整支持。
- IPAM，包括虚拟客户机的 DHCP 管理，处于技术预览状态。
- 通过 FRRouting 实现的复杂路由以及控制器集成，处于技术预览状态。

12.3 安装

12.3.1 SDN 核心

自 Proxmox VE 8.1 起，核心软件定义网络（SDN）软件包会默认安装。

如果从旧版本升级，需要在每个节点上安装 `libpve-network-perl` 软件包：

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

> **Note**
> Proxmox VE 7.0 及以上版本默认安装 `ifupdown2` 软件包。如果系统最初是用更早版本安装的，则需要显式安装 `ifupdown2` 软件包。

安装完成后，需要确保所有节点的 `/etc/network/interfaces` 配置文件末尾存在以下一行，以便包含 SDN 配置。

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

12.3.2 DHCP IPAM

内置 *PVE* IP 地址管理栈中的 DHCP 集成目前使用 `dnsmasq` 分配 DHCP 租约。 该功能目前需要显式启用。

要使用该功能，需要在每个节点上安装 `dnsmasq` 软件包：

```
dnf makecache
apt install dnsmasq
# disable default instance
systemctl disable --now dnsmasq
```

12.3.3 FRRouting

Proxmox VE SDN 栈在高级场景中使用 FRRouting 项目。 该功能目前需要显式启用。

要使用 SDN 路由集成，需要在所有节点上安装 `frr-pythontools` 软件包：

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

然后在所有节点上启用 frr 服务：

```
systemctl enable frr.service
```

12.4 配置概览

配置在数据中心级别的 Web UI 中完成，并分为以下几个部分：

- SDN:: 在这里可以查看当前活动 SDN 状态的概览，并将所有待应用变更应用到整个 集群。
- Zones: 创建和管理虚拟隔离的网络区域
- VNets VNets: 创建虚拟网络网桥并管理子网

Options 类别可用于添加和管理 SDN 设置中使用的附加服务。

- Controllers: 用于在复杂设置中控制第 3 层路由
- DHCP: 为区域定义 DHCP 服务器，在 IPAM 中自动为客户机分配 IP，并通过 DHCP 将其租约提供给
- IPAM: 为客户机启用外部 IP 地址管理
- DNS: 定义 DNS 服务器集成，用于注册虚拟客户机的主机名 和 IP 地址

常用命令示例

```
# pvesh get /cluster/sdn
# pvesh get /cluster/sdn/zones
# pvesh get /cluster/sdn/vnets
```

12.5 技术与配置

Proxmox VE 软件定义网络实现尽可能使用标准 Linux 网络能力。原因是现代 Linux 网络功能已经能够满足功能完整的 SDN 实现所需的几乎全部需求，同时避免增加外部依赖，并减少可能发生故

Proxmox VE SDN 配置位于 `/etc/pve/sdn`，并通过 Proxmox VE 配置文件系统 与所有其他集群节点底层网络栈管理工具各自的配置格式（例如 `ifupdown2` 或 `frr`）。

新的变更不会立即应用，而是先记录为待处理状态。随后可以在 Web 界面的主 *SDN* 概览面板中一次性推出。

SDN 通过位于 */etc/pve/sdn* 的 *.running-config* 和 *.version* 文件跟踪已推出 状态。

12.6 区域

区域定义一个虚拟隔离网络。区域会限制到特定节点并分配权限，从而将用户限制在 某个区域及其包含 VNet 内。

可使用不同技术实现隔离：

- Simple: 隔离网桥。简单的第 3 层路由网桥（NAT）

- VLAN: 虚拟 LAN，是划分 LAN 的经典方法
- QinQ: 堆叠 VLAN（正式名称为 IEEE 802.1ad）
- VXLAN: 通过 UDP 隧道实现第 2 层 VXLAN 网络
- EVPN (BGP EVPN): 结合 BGP 的 VXLAN，用于建立第 3 层路由

12.6.1 通用选项

以下选项适用于所有区域类型：

Nodes
应部署该区域及其关联 VNet 的节点。

IPAM
使用 IP 地址管理（IPAM）工具管理区域中的 IP。可选，默认为 pve。

DNS
DNS API 服务器。可选。

ReverseDNS
反向 DNS API 服务器。可选。

DNSZone
DNS 域名。用于注册主机名，例如 <hostname>.<domain>。DNS 区域必须 已经存在于 DNS 服务器上。可选。

12.6.2 Simple 区域

这是最简单的插件。它会创建一个隔离的 VNet 网桥。该网桥不连接物理接口，虚拟机 流量只在各自节 NAT 或路由式设置。

12.6.3 VLAN 区域

VLAN 插件使用现有的本地 Linux 或 OVS 网桥连接到节点的物理接口。它使用 VNet 中 定义的 VLAN 标记来隔离网段。这样可实现不同节点上虚拟机之间的连通。

VLAN 区域配置选项：

Bridge
本地网桥或 OVS 交换机，需已在允许节点间连接的*每个*节点上配置。

12.6.4 QinQ 区域

QinQ 也称为 VLAN 堆叠，它使用多层 VLAN 标记进行隔离。QinQ 区域定义外层 VLAN 标记（*Service VLAN*），内层 VLAN 标记则由 VNet 定义。

Note
要使用该配置，物理网络交换机必须支持堆叠 VLAN。

QinQ 区域配置选项：

Bridge
已在每个本地节点上配置的本地 VLAN 感知网桥

Service VLAN
该区域的主 VLAN 标记

Service VLAN Protocol
允许选择 802.1q（默认）或 802.1ad 服务 VLAN 类型。

MTU
由于标记会双层堆叠，QinQ VLAN 需要额外 4 字节。例如，如果物理接口 MTU 为 1500，则必须将 MTU 降至 1496。

12.6.5 VXLAN 区域

VXLAN 插件在现有网络（underlay）之上建立隧道（overlay）。它使用默认目标端口 4789，将第 2 层以太网帧封装在第 4 层 UDP 数据报中。

需要自行配置 underlay 网络，以启用所有对等节点之间的 UDP 连通性。

例如，可以在公网之上创建 VXLAN 覆盖网络，让虚拟机看起来像共享同一个本地 第 2 层网络。

Warning
VXLAN 本身不提供任何加密。通过 VXLAN 连接多个站点时，请确保在站点之间建立安全连接，例如使用站点到站点 VPN。

VXLAN 区域配置选项：

Peers Address List
VXLAN 区域中各节点的 IP 地址列表。这里也可以是可通过这些 IP 地址访问的外部节点。集群中的

MTU
由于 VXLAN 封装会使用 50 字节，MTU 需要比出站物理接口低 50 字节。

12.6.6 EVPN 区域

EVPN 区域会创建可路由的第 3 层网络，并能够跨越多个集群。这通过建立 VPN 并将 BGP 用作路由协
EVPN 的 VNet 可以拥有任播 IP 地址和/或 MAC 地址。每个节点上的网桥 IP 相同， 这意味着虚拟客户机
路由可以通过 VRF（Virtual Routing and Forwarding）接口在不同区域的 VNet 之间 工作。
EVPN 区域配置选项：

VRF VXLAN ID
用于 VNet 间专用路由互连的 VXLAN-ID。它必须不同于各 VNet 的 VXLAN-ID。

Controller
该区域要使用的 EVPN-controller。（参见控制器插件章节）。

VNet MAC Address
分配给该区域中所有 VNet 的任播 MAC 地址。如果未定义，将自动 生成。

Exit Nodes
应配置为 EVPN 网络出口网关的节点，通过真实网络出站。配置的节点会 在 EVPN 网络中通告默

Primary Exit Node
如果使用多个出口节点，可强制流量经过该主出口节点，而不是在 所有节点上负载均衡。可选；
SNAT，或上游路由器不支持 ECMP，则是 必需的。

Exit Nodes Local Routing
如果需要从出口节点访问 VM/CT 服务，可以使用该特殊 选项。（默认情况下，出口节点只允许在
EVPN 网络之间转发流量）。可选。

Advertise Subnets
在 EVPN 网络中通告完整子网。如果存在静默 VM/CT（例如，有 多个 IP，而任播网关没有看到来
IP 的流量，则这些 IP 地址在 EVPN 网络内 将无法访问）。可选。

Disable ARP ND Suppression
不抑制 ARP 或 ND（Neighbor Discovery）数据包。 如果虚拟机中使用浮动 IP（IP 和
MAC 地址会在系统之间迁移），则需要该选项。 可选。

Route-target Import
允许导入外部 EVPN route target 列表。用于跨数据中心或不同 EVPN 网络之间的互连。可选。

MTU
由于 VXLAN 封装会使用 50 字节，MTU 需要比出站物理接口的最大 MTU 小 50 字节。可选，默
1450。

12.7 VNet

通过 SDN GUI 创建虚拟网络（VNet）后，每个节点上都会出现一个同名的本地网络 接口。要将客户机
VNet，请将该接口分配给客户机，并相应设置 IP 地址。

根据区域不同，这些选项具有不同含义，并在本文档对应区域章节中说明。

Warning
在当前状态下，某些选项在特定区域中可能没有效果或无法工作。

VNet 配置选项：

ID
用于标识 VNet 的 ID，最长 8 个字符

Comment
更具描述性的标识符。会作为接口别名分配。可选

Zone
该 VNet 关联的区域

Tag
唯一的 VLAN 或 VXLAN ID

VLAN Aware
在接口上启用 vlan-aware 选项，从而允许在客户机内进行配置。

Isolate Ports
为该接口的所有客户机端口设置 isolated 标志，但不为接口本身设置。 这意味着客户机只能向非
重启受影响的客户机。

Note
端口隔离是每台主机本地的行为。请使用 VNET Firewall 进一步隔离跨节点 VNET 流量。
例如，默认 DROP，只允许 IP 子网到网关以及反向的流量。

12.8 子网

子网定义一个由 CIDR 网络地址描述的特定 IP 范围。每个 VNet 可以有一个或多个 子网。
子网可用于：

- 限制可在特定 VNet 上定义的 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 插件为虚拟客户机（VM 或 CT）自动分配 IP
- 通过 DNS 插件进行 DNS 注册

如果子网所在区域关联了 IPAM 服务器，子网前缀会自动注册到 IPAM 中。
子网配置选项：

ID
CIDR 网络地址，例如 10.0.0.0/8

Gateway
网络默认网关的 IP 地址。在第 3 层区域（Simple/EVPN 插件）中，它会 部署到 VNet 上。

SNAT
启用 Source NAT，允许 VNet 内部的虚拟机通过将数据包转发到节点出站接口 来连接外部网络。EVPN 区域中，转发由 EVPN gateway-nodes 完成。可选。

DNS Zone Prefix
为域注册添加前缀，例如 <hostname>.prefix.<domain>。可选。

12.9 控制器

某些区域实现了分离的控制平面和数据平面，因此需要外部控制器来管理 VNet 的控制 平面。

目前，只有 EVPN 区域需要外部控制器。

12.9.1 EVPN Controller

EVPN 区域需要外部控制器来管理控制平面。EVPN controller 插件会配置 Free Range Routing（frr）路由器。

要启用 EVPN controller，需要在每个节点上启用 FRR，参见 安装 FRRouting。

EVPN controller 配置选项：

ASN #
唯一的 BGP ASN 编号。强烈建议使用私有 ASN 编号（64512 - 65534， 4200000000 - 4294967294），否则可能会意外破坏全局路由。

Peers
属于 EVPN 区域的所有节点 IP 列表。（也可以是外部节点或 route reflector 服务器）

12.9.2 BGP Controller

BGP controller 不会被区域直接使用。可以使用它配置 FRR 以管理 BGP 对等体。

对于 BGP-EVPN，它可用于按节点定义不同 ASN，从而实现 EBGP。它也可用于向外部 BGP 对等体导 EVPN 路由。

Note
默认情况下，对于简单的全网状 EVPN，不需要定义 BGP controller。

BGP controller 配置选项：

Node
该 BGP controller 所在节点

ASN #
唯一的 BGP ASN 编号。强烈建议使用 (64512 - 65534) 或 (4200000000 - 4294967294) 范围内的私有 ASN 编号，否则可能会意外破坏全局路由。

Peer
要通过底层 BGP 网络通信的对等 IP 地址列表。

EBGP
如果对等体的 remote-AS 不同，此选项会启用 EBGP。

Loopback Interface
使用 loopback 或 dummy 接口作为 EVPN 网络源地址（用于 多路径）。

ebgp-mutltihop
增加到达对等体的跳数，用于对等体不直接相连或使用 loopback 的 情况。

bgp-multipath-as-path-relax
当对等体使用不同 ASN 时允许 ECMP。

12.9.3 ISIS Controller

ISIS controller 不会被区域直接使用。可以使用它配置 FRR，以将 EVPN 路由导出到 ISIS 域。

ISIS controller 配置选项：

Node
该 ISIS controller 所在节点。

Domain
唯一的 ISIS 域。

Network Entity Title
用于标识该节点的唯一 ISIS 网络地址。

Interfaces
ISIS 使用的物理接口列表。

Loopback
使用 loopback 或 dummy 接口作为 EVPN 网络源地址（用于多路径）。

12.10 IPAM

IP 地址管理（IPAM）工具用于管理网络中客户端的 IP 地址。Proxmox VE 中的 SDN 会使用 IPAM，例 IP 地址。

单个 IPAM 实例可以关联到一个或多个区域。

12.10.1 PVE IPAM Plugin

Proxmox VE 集群默认内置的 IPAM。

可以通过数据中心配置中 SDN 部分的 IPAM 面板查看 PVE IPAM Plugin 的当前状态。 该 UI 可用于创建、更新和删除 IP 映射。配合 DHCP 功能 使用时尤其方便。

如果正在使用 DHCP，可以通过 IPAM 面板为特定虚拟机创建或编辑租约，从而更改通过 DHCP 分配的 IP。编辑使用 DHCP 的虚拟机 IP 时，必须确保强制客户机获取新的 DHCP 租约。通常可以通过

12.10.2 NetBox IPAM Plugin

NetBox 是一个开源 IP 地址管理 （IPAM）和数据中心基础设施管理（DCIM）工具。

要将 NetBox 与 Proxmox VE SDN 集成，请按照此处说明在 NetBox 中创建 API token： https://doc
en/stable/integrations/rest-api/#tokens

NetBox 配置属性如下：

URL
: NetBox REST API 端点：`http://yournetbox.domain.com/api`

Token
: API 访问 token

12.10.3 phpIPAM Plugin

在 phpIPAM 中，需要创建一个 ”application”，并向该 application 添加具有管理员权限的 API token。

phpIPAM 配置属性如下：

URL
: REST-API 端点：`http://phpipam.domain.com/api/<appname>/`

Token
: API 访问 token

Section
: 整数 ID。Section 是 phpIPAM 中的一组子网。默认安装会为客户使用 `sectionid=1`。

12.11 DNS

Proxmox VE SDN 中的 DNS 插件用于定义 DNS API 服务器，以注册主机名和 IP 地址。DNS 配置会关联到一个或多个区域，为某个区域中配置的所有子网 IP 提供 DNS 注册。

12.11.1 PowerDNS Plugin

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

需要在 PowerDNS 配置中启用 Web 服务器和 API：

```
api=yes
api-key=arandomgeneratedstring
webserver=yes
webserver-port=8081
```

PowerDNS 配置选项如下：

url
: REST API 端点：http://yourpowerdnserver.domain.com:8081/api/v1/servers/localhost

key
: API 访问密钥

ttl
: 记录的默认 TTL

12.12 DHCP

Proxmox VE SDN 中的 DHCP 插件可用于为区域自动部署 DHCP 服务器。它会为区域内已配置 DHCP 范围的所有子网提供 DHCP。目前唯一可用的 DHCP 后端插件是 dnsmasq 插件。

DHCP 插件的工作方式是：为 VM/CT 添加新网络接口时，在该区域配置的 IPAM 插件中 分配一个 IP。关于如何配置 IPAM，可以在本文档 相应章节中找到更多信息。

虚拟机启动时，该区域的 DHCP 插件会创建 MAC 地址和 IP 的映射。当网络接口被移除，或 VM/CT 被销毁时，IPAM 和 DHCP 服务器中的条目也会被删除。

Note
某些功能（添加/编辑/移除 IP 映射）目前仅在使用 PVE IPAM plugin 时可用。

12.12.1 配置

可以在 Web UI 的 Zones 面板中，通过区域高级选项启用 DHCP，为区域启用自动 DHCP。

Note
目前只有 Simple Zones 支持自动 DHCP

为区域启用自动 DHCP 后，需要为该区域中的子网配置 DHCP 范围。为此，请进入 Vnets 面板并选择要 DHCP 范围的子网。在编辑对话框中，可以在相应标签页配置 DHCP 范围。也可以通过以下 CLI 命令为子网设置 DHCP 范围：

```
pvesh set /cluster/sdn/vnets/<vnet>/subnets/<subnet>
 -dhcp-range start-address=10.0.1.100,end-address=10.0.1.200
 -dhcp-range start-address=10.0.2.100,end-address=10.0.2.200
```

还需要为子网配置网关，否则自动 DHCP 将无法工作。

随后，DHCP 插件只会在配置的范围内从 IPAM 分配 IP。

也不要忘记按照 dnsmasq DHCP plugin 的安装步骤 操作。

12.12.2 Plugins

Dnsmasq Plugin

目前这是唯一的 DHCP 插件，因此启用区域 DHCP 时会使用该插件。

安装

安装说明请参见 DHCP IPAM 章节。

配置

该插件会为每个部署 dnsmasq 的区域创建一个新的 systemd 服务。服务名称为 `dnsmasq@<zone>`。DHCP 插件管理。

该插件会在 `/etc/dnsmasq.d/<zone>` 文件夹中自动生成以下配置文件：

`00-default.conf`
: 其中包含 dnsmasq 实例的默认全局配置。

`10-<zone>-<subnet_cidr>.conf`
: 该文件配置子网的特定选项，例如应通过 DHCP 配置的 DNS 服务器。

`10-<zone>-<subnet_cidr>.ranges.conf`
: 该文件为 dnsmasq 实例配置 DHCP 范围。

`ethers`
: 该文件包含来自 IPAM 插件的 MAC 地址和 IP 映射。要覆盖这些映射，请使用相应的 IPAM 插件，而不是编辑此文件，因为它会被 dnsmasq 插件覆盖。

不要编辑上述任何文件，因为它们由 DHCP 插件管理。要自定义 dnsmasq 配置，可以在 配置文件夹中 `90-custom.conf`），dnsmasq DHCP 插件不会修改 这些文件。

配置文件会按顺序读取，因此可以通过合理命名自定义配置文件来控制配置指令顺序。

DHCP 租约存储在 `/var/lib/misc/dnsmasq.<zone>.leases` 文件中。

使用 PVE IPAM plugin 时，可以更新、创建和删除 DHCP 租约。更多信息请参阅 PVE IPAM plugin 文档。其他 IPAM 插件目前不支持 更改 DHCP 租约。

12.13 防火墙集成

SDN 会与 Proxmox VE 防火墙集成，自动生成 IPSets；随后可在防火墙规则的 source / destination 字段中引用这些 IPSets。对于 VNet 和 IPAM 条目，这一过程会自动发生。

12.13.1 VNet 和子网

防火墙会在 SDN 范围内为每个 VNet 自动生成以下 IPSets：

vnet-all
: 包含 VNet 中所有子网的 CIDR

vnet-gateway
: 包含 VNet 中所有子网网关的 IP

vnet-no-gateway
: 包含 VNet 中所有子网的 CIDR，但排除网关

vnet-dhcp
: 包含 VNet 中各子网配置的所有 DHCP 范围

更改配置时，IPSets 会自动更新，因此更改子网配置时不必同步更新防火墙规则。

Simple 区域示例

假设某个 VNet 及其包含子网的配置如下：

```
# /etc/pve/sdn/vnets.cfg

vnet: vnet0
        zone simple

# /etc/pve/sdn/subnets.cfg

subnet: simple-192.0.2.0-24
        vnet vnet0
        dhcp-range start-address=192.0.2.100,end-address=192.0.2.199
        gateway 192.0.2.1

subnet: simple-2001:db8::-64
        vnet vnet0
        dhcp-range start-address=2001:db8::1000,end-address=2001:db8::1999
        gateway 2001:db8::1
```

在此示例中，我们在 VNet `vnet0` 中配置了一个 IPv4 子网，其 IP 范围为 *192.0.2.0/24*，网关为 *192.0.2.1*，DHCP 范围为 *192.0.2.100 - 192.0.2.199*。

此外，我们还配置了一个 IPv6 子网，其 IP 范围为 *2001:db8::/64*，网关为 *2001:db8::1*，DHCP 范围为 *2001:db8::1000 - 2001:db8::1999*。

随后，vnet0 对应的自动生成 IPsets 将包含以下元素：

vnet0-all

- *192.0.2.0/24*
- *2001:db8::/64*

vnet0-gateway

- *192.0.2.1*
- *2001:db8::1*

vnet0-no-gateway

- *192.0.2.0/24*
- *2001:db8::/64*
- *!192.0.2.1*
- *!2001:db8::1*

vnet0-dhcp

- *192.0.2.100 - 192.0.2.199*
- *2001:db8::1000 - 2001:db8::1999*

12.13.2 IPAM

如果使用内置 PVE IPAM，防火墙会为 IPAM 中存在条目的每个客户机自动生成一个 IPset。ID 为 100 的客户机对应 IPset 为 `guest-ipam-100`。它包含所有 IPAM 条目 中的全部 IP 地址。因此，如果 100 属于多个 VNet，该 IPset 将包含来自 所有 VNet 的 IP。

条目被添加、更新或删除时，相应 IPSets 会同步更新。

Warning
如果移除了某个客户机的所有条目，但仍有防火墙规则引用自动生成的 IPSet，则防火墙会因为引用不存在的 IPSet 而无法更新规则集。

12.14 示例

本节给出多个面向常见 SDN 使用场景的配置示例。其目标是提供具体实现，并补充 细节，帮助理解可用

12.14.1 Simple 区域示例

Simple 区域网络会为单台主机上的客户机创建一个隔离网络，使它们能够互相通信。

Tip
如果所有客户机都位于同一主机上，则客户机之间可以通信，但其他节点无法访问 它们。

- 创建名为 simple 的 simple 区域。
- 添加名为 vnet1 的 VNet。
- 创建一个带网关且启用 SNAT 选项的子网。
- 这会在节点上创建网络网桥 vnet1。将该网桥分配给要加入网络的客户机，并配置 IP 地址。

两个虚拟机中的网络接口配置可以如下所示，这允许它们通过 10.0.1.0/24 网络通信。

```
allow-hotplug ens19
iface ens19 inet static
        address 10.0.1.14/24
```

```
allow-hotplug ens19
iface ens19 inet static
        address 10.0.1.15/24
```

12.14.2 源 NAT 示例

如果希望允许 simple 网络区域中的客户机向外建立连接，simple 区域提供了 源 NAT（SNAT）选项。

基于xref:pvesdn_setup_example_simple[上述]配置，向 VNet vnet1 添加一个子网， 设置网关 IP，并启用 SNAT 选项。

```
Subnet: 172.16.0.0/24
Gateway: 172.16.0.1
SNAT: checked
```

在客户机中配置位于该子网 IP 范围内的静态 IP 地址。

节点本身会使用网关 IP *172.16.0.1* 加入该网络，并作为子网范围内客户机的 NAT 网关。

12.14.3 VLAN 设置示例

当不同节点上的虚拟机需要通过隔离网络通信时，VLAN 区域可使用 VLAN 标记实现 网络级隔离。

创建名为 myvlanzone 的 VLAN 区域：

```
ID: myvlanzone
Bridge: vmbr0
```

创建名为 myvnet1 的 VNet，使用 VLAN 标记 10，并关联前面创建的 myvlanzone。

```
ID: myvnet1
Zone: myvlanzone
Tag: 10
```

通过主 SDN 面板应用配置，以便在每个节点本地创建 VNet。

在 node1 上创建一个基于 Debian 的虚拟机（*vm1*），并为其配置位于 myvnet1 上的 vNIC。

该虚拟机使用以下网络配置：

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

在 node2 上创建第二台虚拟机（*vm2*），并为其配置与 vm1 相同 VNet myvnet1 上的 vNIC。

该虚拟机使用以下网络配置：

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

完成后，应能够通过该网络在两台虚拟机之间互相 ping 通。

12.14.4 QinQ 设置示例

本示例配置两个 QinQ 区域，并向每个区域添加两台虚拟机，用于演示额外 VLAN 标记层 如何支持配置 VLAN。

该配置的典型使用场景是托管服务提供商为客户提供隔离网络供虚拟机通信，同时将 这些虚拟机与其他

创建名为 qinqzone1、service VLAN 为 20 的 QinQ 区域

```
ID: qinqzone1
Bridge: vmbr0
Service VLAN: 20
```

创建另一个名为 qinqzone2、service VLAN 为 30 的 QinQ 区域

```
ID: qinqzone2
Bridge: vmbr0
Service VLAN: 30
```

在前面创建的 qinqzone1 区域上创建名为 myvnet1、VLAN-ID 为 100 的 VNet。

```
ID: qinqvnet1
Zone: qinqzone1
Tag: 100
```

在 qinqzone2 区域上创建 VLAN-ID 为 100 的 myvnet2。

```
ID: qinqvnet2
Zone: qinqzone2
Tag: 100
```

在主 SDN Web 界面面板上应用配置，以在每个节点本地创建 VNet。

创建四台基于 Debian 的虚拟机（vm1、vm2、vm3、vm4），为 vm1 和 vm2 添加使用 网桥 qinqvnet1 的网络接口，为 vm3 和 vm4 添加使用网桥 qinqvnet2 的网络接口。

在虚拟机内部配置接口 IP 地址，例如通过 /etc/network/interfaces：

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

为四台虚拟机配置 *10.0.3.101* 到 *10.0.3.104* 范围内的 IP 地址。

现在应能够在虚拟机 *vm1* 与 *vm2* 之间，以及 *vm3* 与 *vm4* 之间互相 ping 通。 但是，*vm1* 或 *vm2* 都无法 ping 通 *vm3* 或 *vm4*，因为它们位于不同区域，并 使用不同的 service-VLAN。

12.14.5 VXLAN 设置示例

本示例假设有一个包含三个节点的集群，节点 IP 地址分别为 192.168.0.1、 192.168.0.2 和 192.168

创建名为 myvxlanzone 的 VXLAN 区域，并将所有节点 IP 添加到 peer address list。 使用默认 MTU 1450，或根据实际情况配置。

```
ID: myvxlanzone
Peers Address List: 192.168.0.1,192.168.0.2,192.168.0.3
```

使用前面创建的 VXLAN 区域 myvxlanzone 创建名为 vxvnet1 的 VNet。

```
ID: vxvnet1
Zone: myvxlanzone
Tag: 100000
```

在主 SDN Web 界面面板上应用配置，以在每个节点本地创建 VNet。

在 node1 上创建一台基于 Debian 的虚拟机（*vm1*），并为其配置位于 vxvnet1 上的 vNIC。

该虚拟机使用以下网络配置（注意较低的 MTU）。

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

在 node3 上创建第二台虚拟机（*vm2*），并为其配置与 vm1 相同 VNet vxvnet1 上的 vNIC。

该虚拟机使用以下网络配置：

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

随后，应能够在 *vm1* 与 *vm2* 之间互相 ping 通。

12.14.6 EVPN 设置示例

本示例假设有一个包含三个节点（node1、node2、node3）的集群，节点 IP 地址分别为 192.168.0
和 192.168.0.3。

创建一个 EVPN controller，使用私有 ASN 编号，并将上述节点地址作为 peers。

```
ID: myevpnctl
ASN#: 65000
Peers: 192.168.0.1,192.168.0.2,192.168.0.3
```

创建名为 myevpnzone 的 EVPN 区域，分配前面创建的 EVPN-controller，并将 *node1* 和 *node2* 定义为出口节点。

```
ID: myevpnzone
VRF VXLAN Tag: 10000
Controller: myevpnctl
MTU: 1450
VNet MAC Address: 32:F4:05:FE:6C:0A
Exit Nodes: node1,node2
```

使用 EVPN 区域 myevpnzone 创建第一个名为 myvnet1 的 VNet。

```
ID: myvnet1
Zone: myevpnzone
Tag: 11000
```

在 myvnet1 上创建子网：

```
Subnet: 10.0.1.0/24
Gateway: 10.0.1.1
```

使用同一个 EVPN 区域 myevpnzone 创建第二个名为 myvnet2 的 VNet。

```
ID: myvnet2
Zone: myevpnzone
Tag: 12000
```

在 myvnet2 上创建另一个子网：

```
Subnet: 10.0.2.0/24
Gateway: 10.0.2.1
```

从主 SDN Web 界面面板应用配置，以在每个节点本地创建 VNet 并生成 FRR 配置。

在 node1 上创建一台基于 Debian 的虚拟机（*vm1*），并为其配置位于 myvnet1 上的 vNIC。

为 *vm1* 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
        mtu 1450
```

在 node2 上创建第二台虚拟机（*vm2*），并为其配置位于另一个 VNet myvnet2 上的 vNIC。

为 *vm2* 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
        mtu 1450
```

现在应能够从 vm1 ping 通 vm2，也能从 vm2 ping 通 vm1。

如果从非网关 node3 上的 *vm2* ping 外部 IP，数据包会先到达配置的 *myvnet2* 网关，然后路由到出口
或 *node2*），再从这些节点经由 node1 或 node2 上配置的默认网关离开。

Note
需要在外部网关上为 *10.0.1.0/24* 和 *10.0.2.0/24* 网络添加指向 node1 和 node2 的反向路由，以便公网能够回包。

如果已配置外部 BGP 路由器，BGP-EVPN 路由（本例中为 10.0.1.0/24 和 10.0.2.0/24）将动态通告。

12.15 说明

12.15.1 多个 EVPN 出口节点

如果有多个网关节点，应禁用 rp_filter（Strict Reverse Path Filter）选项，因为 数据包可能从一个

将以下内容添加到 /etc/sysctl.conf：

```
net.ipv4.conf.default.rp_filter=0
net.ipv4.conf.all.rp_filter=0
```

12.15.2 VXLAN IPSEC 加密

要在 VXLAN 之上添加 IPSEC 加密，本示例展示了如何使用 strongswan。

需要为加密额外降低 *MTU*：IPv4 降低 60 字节，IPv6 降低 80 字节。

因此，在实际默认 MTU 为 1500 的情况下，需要使用 1370 的 MTU（1370 + 80 (IPSEC) + 50 (VXLAN) == 1500）。

在主机上安装 strongswan。

```
apt install strongswan
```

向 /etc/ipsec.conf 添加配置。这里只需要加密来自 VXLAN UDP 端口 *4789* 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
    authby=psk
    auto=route

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

使用以下命令生成预共享密钥：

```
openssl rand -base64 128
```

并将密钥添加到 /etc/ipsec.secrets，使文件内容如下所示：

```
: PSK <generatedbase64key>
```

将 PSK 和配置复制到参与 VXLAN 网络的所有节点。

Chapter 13

Proxmox VE 防火墙

Proxmox VE 防火墙提供了一种简单方式来保护 IT 基础设施。你可以为集群中的所有主机设置 防火墙规
集和别名等功能可使这项 工作更加轻松。

所有配置都存储在集群文件系统中，而基于 `iptables` 的防火墙服务会在每个集群节点上运行，
因此可在虚拟机之间提供完整隔离。该系统的分布式特性也比集中式防火墙方案提供更高带宽。

防火墙完整支持 IPv4 和 IPv6。IPv6 支持完全透明，并且默认会同时过滤两种协议的流量。 因此无需为
IPv6 维护另一套规则。

13.1 方向与区域

Proxmox VE 防火墙将网络分为多个逻辑区域。可以分别为每个区域定义规则。根据区域不同， 可以为

13.1.1 方向

为区域定义规则时，可以从 3 个方向中选择：

In
进入某个区域的流量。

Out
离开某个区域的流量。

Forward
穿过某个区域的流量。在主机区域中，这可以是路由流量（当主机充当网关或执行 NAT 时）。
在 VNet 级别，这会影响所有经过 VNet 的流量，包括来自/发往桥接网络接口的流量。

Important
当前只有在使用新的 基于 nftables 的 proxmox-firewall 时，才能为转发流量创建规则。任何 forward 规则都会被原有 `pve-firewall` 忽略，且不会 产生效果！

13.1.2 区域

可以为 3 个不同区域定义防火墙规则：

Host
发往/来自主机的流量，或由主机转发的流量。可以在数据中心级别或主机级别为该区域定义规则。主机级别规则优先于数据中心级别规则。

VM
发往/来自 VM 或 CT 的流量。不能为转发流量定义规则，只能为入站/出站流量定义规则。

VNet
穿过 SDN VNet 的流量，可以是客户机到客户机，也可以是主机到客户机或反向流量。由于该流量始终属于转发流量，因此只能创建方向为 forward 的规则。

Important
当前只有在使用新的 基于 nftables 的 proxmox-firewall 时，才能在 VNet 级别创建规则。任何 VNet 级别规则都会被原有 `pve-firewall` 忽略，且不会产生效果！

13.2 配置文件

所有防火墙相关配置都存储在 proxmox 集群文件系统中。因此，这些文件会自动分发到所有 集群节点，`pve-firewall` 服务会在配置变更时自动更新底层 `iptables` 规则。

可以使用 GUI 配置所有内容（即 **Datacenter → Firewall**，或在 **Node → Firewall** 下配置），也可

防火墙配置文件包含由键值对组成的节。以 # 开头的行和空行视为注释。节以标题行开始， 节名用 `[` 和 `]` 包围。

13.2.1 集群范围设置

集群范围的防火墙配置存储在：

`/etc/pve/firewall/cluster.fw`

该配置可以包含以下节：

`[OPTIONS]`
用于设置集群范围的防火墙选项。

ebtables: <boolean> (*default* = 1)
在整个集群范围内启用 ebtables 规则。

enable: <integer> (0 - N)
在整个集群范围内启用或禁用防火墙。

log_ratelimit: [enable=]<1|0> [,burst=<integer>] [,rate=<rate>]
日志限速设置

burst=<integer> (0 - N) (*default* = 5)
应用速率限制前始终会记录的初始突发包数量

enable=<boolean> (*default* = 1)
启用或禁用日志速率限制

rate=<rate> (*default* = 1/second)
突发 bucket 重新填充的频率

policy_forward: <ACCEPT | DROP>
转发策略。

policy_in: <ACCEPT | DROP | REJECT>
入站策略。

policy_out: <ACCEPT | DROP | REJECT>
出站策略。

[RULES]
该节包含适用于所有节点的集群范围防火墙规则。

[IPSET <name>]
集群范围 IP 集定义。

[GROUP <name>]
集群范围安全组定义。

[ALIASES]
集群范围别名定义。

启用防火墙

防火墙默认完全禁用，因此需要在此设置 enable 选项：

```
[OPTIONS]
# enable firewall (cluster-wide setting, default is disabled)
enable: 1
```

Important
如果启用防火墙，默认会阻止到所有主机的流量。唯一例外是来自本地网络的 WebGUI(8006) 和 ssh(22)。

如果希望远程管理 Proxmox VE 主机，需要创建规则，允许来自这些远程 IP 的流量访问 Web GUI（端口 8006）。你可能还希望允许 ssh（端口 22），以及可能需要的 SPICE（端口 3128）。

Tip
启用防火墙前，请先打开到某个 Proxmox VE 主机的 SSH 连接。这样即使出现问题，仍然可以访问该主机。

为简化此任务，也可以创建名为 “management” 的 IPSet，并将所有远程 IP 添加到其中。 这样会创建 GUI 所需的所有防火墙规则。

13.2.2 主机专用配置

主机相关配置读取自：

`/etc/pve/nodes/<nodename>/host.fw`

如果想覆盖 `cluster.fw` 配置中的规则，这会很有用。也可以提高日志详细程度，并设置 netfilter 相关选项。该配置可以包含以下节：

[OPTIONS]
用于设置主机相关防火墙选项。

enable: <boolean>
启用主机防火墙规则。

log_level_forward: <alert | crit | debug | emerg | err | info | nolog | notice | warning>
转发流量的日志级别。

log_level_in: <alert | crit | debug | emerg | err | info | nolog | notice | warning>
入站流量的日志级别。

log_level_out: <alert | crit | debug | emerg | err | info | nolog | notice | warning>
出站流量的日志级别。

log_nf_conntrack: <boolean> (*default* = 0)
启用 conntrack 信息日志记录。

ndp: <boolean> (*default* = 0)
启用 NDP（Neighbor Discovery Protocol，邻居发现协议）。

nf_conntrack_allow_invalid: <boolean> (*default* = 0)
允许连接跟踪中的无效数据包。

nf_conntrack_helpers: <string> (*default* = ``)
为特定协议启用 conntrack helper。支持的协议：amanda, ftp, irc, netbios-ns, pptp, sane, sip, snmp, tftp

nf_conntrack_max: <integer> (32768 - N) (*default* = 262144)
可跟踪连接的最大数量。

nf_conntrack_tcp_timeout_established: <integer> (7875 - N) (*default* = 432000)
conntrack 已建立连接的超时时间。

nf_conntrack_tcp_timeout_syn_recv: <integer> (30 - 60) (*default* = 60)
conntrack syn recv 状态的超时时间。

nftables: <boolean> (*default* = 0)
启用基于 nftables 的防火墙（技术预览）

nosmurfs: <boolean>
启用 SMURFS 过滤器。

protection_synflood: <boolean> (*default* = 0)
启用 synflood 防护

protection_synflood_burst: <integer> (*default* = 1000)
按源 IP 统计的 synflood 防护突发速率。

protection_synflood_rate: <integer> (*default* = 200)
按源 IP 统计的 synflood 防护速率，单位为 syn/sec。

smurf_log_level: <alert | crit | debug | emerg | err | info | nolog | notice | warning>
SMURFS 过滤器的日志级别。

tcp_flags_log_level: <alert | crit | debug | emerg | err | info | nolog | notice | warning>
非法 TCP 标志过滤器的日志级别。

tcpflags: <boolean> (*default* = 0)
过滤非法的 TCP 标志组合。

[RULES]
该节包含主机专用防火墙规则。

13.2.3 VM/容器配置

VM 防火墙配置读取自：

`/etc/pve/firewall/<VMID>.fw`

并包含以下数据：

[OPTIONS]
用于设置 VM/容器相关防火墙选项。

dhcp: <boolean> (*default* = 0)
启用 DHCP。

enable: <boolean> (*default* = 0)
启用或禁用防火墙规则。

ipfilter: <boolean>
启用默认 IP 过滤器。这等同于为每个接口添加一个空的 ipfilter-net<id> ipset。 此类 ipset 隐式包含合理的默认限制，例如将 IPv6 链路本地地址限制为由接口 MAC 地址派生出的地址。对于 IP 地址会被隐式加入。

log_level_in: <alert | crit | debug | emerg | err | info | nolog | notice | warning>
入站流量的日志级别。

log_level_out: <alert | crit | debug | emerg | err | info | nolog | notice | warning>
出站流量的日志级别。

macfilter: <boolean> (*default* = 1)
启用或禁用 MAC 地址过滤器。

ndp: <boolean> (*default* = 0)
启用 NDP (Neighbor Discovery Protocol) 。

policy_in: <ACCEPT | DROP | REJECT>
输入策略。

policy_out: <ACCEPT | DROP | REJECT>
输出策略。

radv: <boolean>
允许发送 Router Advertisement。

[RULES]
该节包含 VM/容器防火墙规则。

[IPSET <name>]
IP 集定义。

[ALIASES]
IP 别名定义。

为 **VM** 和容器启用防火墙

每个虚拟网络设备都有自己的防火墙启用标志。因此可以为每个接口选择性启用防火墙。 除了通用防火墙 enable 选项之外，还必须设置该标志。

13.2.4 VNet 配置

VNet 相关配置读取自：

`/etc/pve/sdn/firewall/<vnet_name>.fw`

这可用于在 VNet 级别全局设置防火墙配置，而不必为 VNet 内部的每个 VM 分别设置防火墙规则。
它只能包含 FORWARD 方向的规则，因为这里没有入站或出站流量的概念。这会影响从一个桥接 端口流

Warning
当前该功能仅适用于新的 基于 nftables 的 proxmox-firewall

由于经过 FORWARD 链的流量是双向的，如果希望流量双向通过，就需要为两个方向都创建规则。
例如，如果要允许某个特定主机的 HTTP 流量，需要创建以下规则：

```
FORWARD ACCEPT -dest 10.0.0.1 -dport 80
FORWARD ACCEPT -source 10.0.0.1 -sport 80
```

[OPTIONS]
用于设置 VNet 相关防火墙选项。

enable: <boolean> (*default* = 0)
启用或禁用防火墙规则。

log_level_forward: <alert | crit | debug | emerg | err | info | nolog | notice | warning>
转发流量的日志级别。

policy_forward: <ACCEPT | DROP>
转发策略。

[RULES]
该节包含 VNet 专用防火墙规则。

13.3 防火墙规则

防火墙规则由方向（IN、OUT 或 FORWARD）和动作（ACCEPT、DENY、REJECT）组成。也可以指定宏
| 来禁用规则。

防火墙规则语法

```
[RULES]

DIRECTION ACTION [OPTIONS]
|DIRECTION ACTION [OPTIONS] # disabled rule

DIRECTION MACRO(ACTION) [OPTIONS] # use predefined macro
```

可使用以下选项细化规则匹配。

--dest <string>

限制数据包目标地址。可以引用单个 IP 地址、IP set（*+ipsetname*）或 IP 别名定义。也可以指定 *20.34.101.207-201.3.9.99* 的地址范围，或 IP 地址和网络列表（条目用逗号分隔）。请不要在 IPv4 和 IPv6 地址。

--dport <string>

限制 TCP/UDP 目标端口。可以使用 */etc/services* 中定义的服务名或简单数字（0-65535）。端 *\d+:\d+* 指定，例如 *80:85*；也可以使用逗号分隔列表匹配多个端口或范围。

--icmp-type <string>

指定 icmp-type。仅当 proto 等于 *icmp* 或 *icmpv6*/*ipv6-icmp* 时有效。

--iface <string>

网络接口名称。对于虚拟机和容器，必须使用网络配置键名（*net\d+*）。主机相关规则可以使用

--log <alert | crit | debug | emerg | err | info | nolog | notice | warning>

防火墙规则的日志级别。

--proto <string>

IP 协议。可以使用 */etc/protocols* 中定义的协议名（*tcp*/*udp*）或简单数字。

--source <string>

限制数据包源地址。可以引用单个 IP 地址、IP set（*+ipsetname*）或 IP 别名定义。也可以指定 *20.34.101.207-201.3.9.99* 的地址范围，或 IP 地址和网络列表（条目用逗号分隔）。请不要在 IPv4 和 IPv6 地址。

--sport <string>

限制 TCP/UDP 源端口。可以使用 */etc/services* 中定义的服务名或简单数字（0-65535）。端口 *\d+:\d+* 指定，例如 *80:85*；也可以使用逗号分隔列表匹配多个端口或范围。

下面是一些示例：

```
[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

安全组是在集群级别定义的一组规则，可用于所有 VM 的规则。例如，可以定义一个名为 “web-server” 的组，其中包含打开 *http* 和 *https* 端口的规则。

```
# /etc/pve/firewall/cluster.fw

[group webserver]
IN  ACCEPT -p tcp -dport 80
IN  ACCEPT -p tcp -dport 443
```

然后，可以将该组添加到 VM 的防火墙：

```
# /etc/pve/firewall/<VMID>.fw

[RULES]
GROUP webserver
```

13.5 IP 别名

IP 别名允许将网络 IP 地址与名称关联。之后可以引用这些名称：

- 在 IP 集定义中
- 在防火墙规则的 source 和 dest 属性中

13.5.1 标准 IP 别名 local_network

该别名会自动定义。请使用以下命令查看分配的值：

```
# pve-firewall localnet
local hostname: example
local IP address: 192.168.2.100
network auto detect: 192.168.0.0/20
using detected local_network: 192.168.0.0/20
```

防火墙会使用此别名自动设置规则，以允许集群通信（corosync、API、SSH）所需的全部流量。

用户可以在 cluster.fw 的别名节中覆盖这些值。如果在公共网络上使用单台主机，最好显式 分配本地 IP 地址：

```
#  /etc/pve/firewall/cluster.fw
[ALIASES]
local_network 1.2.3.4 # use the single IP address
```

13.6 IP 集

IP 集可用于定义网络和主机组。可以在防火墙规则的 source 和 dest 属性中使用 +name 引用它们。

以下示例允许来自 management IP 集的 HTTP 流量。

```
IN HTTP(ACCEPT) -source +management
```

13.6.1 标准 IP 集 management

该 IP 集仅适用于主机防火墙（不适用于 VM 防火墙）。这些 IP 被允许执行正常管理任务（Proxmox VE GUI、VNC、SPICE、SSH）。

本地集群网络会自动添加到此 IP 集（别名 cluster_network），以启用主机间集群通信（multicast、ssh 等）。

```
# /etc/pve/firewall/cluster.fw

[IPSET management]
192.168.2.10
192.168.2.10/24
```

13.6.2 标准 IP 集 blacklist

来自这些 IP 的流量会被每台主机和每个 VM 的防火墙丢弃。

```
# /etc/pve/firewall/cluster.fw

[IPSET blacklist]
77.240.159.182
213.87.123.0/24
```

13.6.3 标准 IP 集 ipfilter-net*

这些过滤器属于 VM 的网络接口，主要用于防止 IP 欺骗。如果某个接口存在此类集合，则任何源 IP 与该接口对应 ipfilter 集不匹配的出站流量都会被丢弃。

对于已配置 IP 地址的容器，如果这些集合存在（或通过 VM 防火墙 **options** 选项卡中的通用 IP Filter 选项启用），它们会隐式包含关联的 IP 地址。

对于虚拟机和容器，它们还会隐式包含标准的、由 MAC 派生的 IPv6 link-local 地址，以允许 邻居发现

```
/etc/pve/firewall/<VMID>.fw

[IPSET ipfilter-net0] # only allow specified IPs on net0
192.168.2.10
```

13.7 服务和命令

防火墙会在每个节点上运行两个服务守护进程：

- pvefw-logger：NFLOG 守护进程（ulogd 替代品）。
- pve-firewall：更新 iptables 规则

还有一个名为 pve-firewall 的 CLI 命令，可用于启动和停止防火墙服务：

```
# pve-firewall start
# pve-firewall stop
```

要获取状态，请使用：

```
# pve-firewall status
```

上述命令会读取并编译所有防火墙规则，因此如果防火墙配置包含错误，就会看到警告。

如果想查看生成的 iptables 规则，可以使用：

```
# iptables-save
```

13.8 默认防火墙规则

默认防火墙配置会过滤以下流量：

13.8.1 数据中心入站/出站 DROP/REJECT

如果防火墙的 input 或 output policy 设置为 DROP 或 REJECT，集群中所有 Proxmox VE 主机仍会允许 以下流量：

- loopback 接口上的流量
- 已建立连接
- 使用 IGMP 协议的流量
- 从管理主机到端口 8006 的 TCP 流量，用于允许访问 Web 界面
- 从管理主机到端口范围 5900 到 5999 的 TCP 流量，用于允许 VNC Web 控制台流量
- 从管理主机到端口 3128 的 TCP 流量，用于连接 SPICE 代理
- 从管理主机到端口 22 的 TCP 流量，用于允许 ssh 访问
- 集群网络中到端口 5405-5412 的 UDP 流量，用于 corosync
- 集群网络中的 UDP multicast 流量

- ICMP 流量类型 3（Destination Unreachable）、4（congestion control）或 11 （Time Exceeded）

以下流量会被丢弃，即使启用了日志记录也不会记录：

- 连接状态无效的 TCP 连接
- 与 corosync 无关的 Broadcast、multicast 和 anycast 流量，即不是通过端口 5405-5412 进入的流量
- 到端口 43 的 TCP 流量
- 到端口 135 和 445 的 UDP 流量
- 到端口范围 137 到 139 的 UDP 流量
- 源端口为 137、目标端口范围为 1024 到 65535 的 UDP 流量
- 到端口 1900 的 UDP 流量
- 到端口 135、139 和 445 的 TCP 流量
- 源端口为 53 的 UDP 流量

其余流量会分别被丢弃或拒绝，并且会被记录。具体行为可能因 **Firewall → Options** 中启用的额外选
NDP、SMURFS 和 TCP flag filtering。

请检查以下系统命令的输出：

```
# iptables-save
```

以查看系统上当前活动的防火墙链和规则。该输出也包含在 System Report 中，可以通过 Web GUI 中节点的订阅选项卡访问，也可以通过 pvereport 命令行工具获取。

13.8.2 VM/CT 入站/出站 DROP/REJECT

这会丢弃或拒绝发往 VM 的所有流量，但会根据配置对 DHCP、NDP、Router Advertisement、MAC 和 IP filtering 做出一些例外处理。丢弃/拒绝数据包的规则会从数据中心继承，而主机 已接受入站

同样，可以使用 iptables-save（见上文）检查所有已应用的 规则和链。

13.9 防火墙规则日志

默认情况下，防火墙规则过滤流量的所有日志记录均被禁用。要启用日志记录，需要在 **Firewall → Options** 中为入站和/或出站流量设置 loglevel。主机防火墙和 VM/CT 防火墙都可以单独设置。这
Proxmox VE 标准防火墙规则的日志记录，并可在 **Firewall → Log** 中查看输出。此外，对于标准规则
（参见 默认防火墙规则）。

loglevel 不会影响被过滤流量中有多少会被记录。它会改变附加在日志输出前缀中的 LOGID，以便过滤和后处理。

loglevel 是以下标志之一：

loglevel	LOGID
nolog	—
emerg	0
alert	1
crit	2
err	3
warning	4
notice	5
info	6
debug	7

典型的防火墙日志输出如下：

```
VMID LOGID CHAIN TIMESTAMP POLICY: PACKET_DETAILS
```

对于主机防火墙，VMID 等于 0。

13.9.1 用户定义防火墙规则的日志记录

为了记录由用户定义防火墙规则过滤的数据包，可以为每条规则单独设置 log-level 参数。这样可以进行细粒度日志记录，并且不受 **Firewall → Options** 中为标准规则定义的 log-level 影响。

每条规则的 `loglevel` 可以在 Web UI 中创建或修改规则时轻松定义或更改，也可以通过对应的 pvesh API 调用设置。

此外，也可以通过防火墙配置文件设置 log-level，即在所选规则后追加 `-log <loglevel>`（参见 可用 log-level）。

例如，以下两条规则等价：

```
IN REJECT -p icmp -log nolog
IN REJECT -p icmp
```

而：

```
IN REJECT -p icmp -log debug
```

会产生带有 debug 级别标记的日志输出。

13.10 提示与技巧

13.10.1 如何允许 FTP

FTP 是一种旧式协议，会使用端口 21 以及其他多个动态端口。因此需要一条规则来接受端口 21。此外，还需要加载 `ip_conntrack_ftp` 模块。请运行：

```
modprobe ip_conntrack_ftp
```

并将 `ip_conntrack_ftp` 添加到 `/etc/modules`（以便重启后仍然生效）。

13.10.2 Suricata IPS 集成

如果希望使用 Suricata IPS（Intrusion Prevention System），这是可行的。

只有在防火墙 ACCEPT 数据包后，数据包才会转发给 IPS。

被拒绝/丢弃的防火墙数据包不会进入 IPS。

在 proxmox 主机上安装 suricata：

```
# apt-get install suricata
# modprobe nfnetlink_queue
```

不要忘记将 nfnetlink_queue 添加到 /etc/modules，以便下次重启后生效。

然后，为特定 VM 启用 IPS：

```
# /etc/pve/firewall/<VMID>.fw

[OPTIONS]
ips: 1
ips_queues: 0
```

ips_queues 会为该 VM 绑定特定 CPU 队列。

可用队列定义在：

```
# /etc/default/suricata
NFQUEUE=0
```

13.11 IPv6 说明

防火墙包含一些 IPv6 专用选项。需要注意的是，IPv6 不再使用 ARP 协议，而是使用工作在 IP 层的 NDP（Neighbor Discovery Protocol），因此需要 IP 地址才能成功工作。为此，会使用从接口 MAC 地址派生出的 link-local 地址。默认情况下，主机和 VM 级别都会启用 NDP 选项，以允许发送和接收邻居发现（NDP）数据包。

除邻居发现外，NDP 还用于其他一些功能，例如自动配置和发布路由器。

默认情况下，允许 VM 发送 router solicitation 消息（用于查询路由器），并接收 router advertisement 数据包。这使它们可以使用无状态自动配置。另一方面，除非设置 “Allow Router Advertisement”（radv: 1）选项，否则 VM 不能将自己通告为路由器。

对于 NDP 所需的 link-local 地址，也可以启用 “IP Filter”（ipfilter: 1）选项，其效果 等同于为 VM 的每个网络接口添加一个包含对应 link-local 地址的 ipfilter-net* ipset。（详情请参见 标准 IP 集 ipfilter-net* 一节。）

13.12 Proxmox VE 使用的端口

- Web 界面：8006（TCP，HTTP/1.1 over TLS）
- VNC Web 控制台：5900-5999（TCP，WebSocket）

- SPICE 代理：3128（TCP）
- sshd（用于集群操作）：22（TCP）
- rpcbind：111（UDP）
- sendmail：25（TCP，出站）
- corosync 集群流量：5405-5412 UDP
- 在线迁移（VM 内存和本地磁盘数据）：60000-60050（TCP）

13.13 nftables

作为 pve-firewall 的替代方案，我们提供 proxmox-firewall。它是 Proxmox VE 防火墙的一个实
nftables， 而不是 iptables。

Warning
proxmox-firewall 当前处于技术预览阶段。它可能存在错误，或与原防火墙不兼容。目前不适合生产使用。

该实现使用相同的配置文件和配置格式，因此切换时可以使用旧配置。除少数例外外，它提供完全相同的功能：

- 当前无法对客户机流量使用 REJECT（流量会改为被丢弃）。
- 使用 NDP、Router Advertisement 或 DHCP 选项时，*始终*会创建防火墙规则，无论默认策略如
- 即使连接已有 conntrack table entries，也会评估客户机防火墙规则。

13.13.1 安装和使用

安装 proxmox-firewall 软件包：

```
apt install proxmox-firewall
```

通过主机上的 Web UI（Host > Firewall > Options > nftables）启用 nftables 后端，或在主机配置文件 (/etc/pve/nodes/<node_name>/host.fw) 中启用：

```
[OPTIONS]

nftables: 1
```

Note
启用/禁用 proxmox-firewall 后，需要重启所有正在运行的 VM 和容器，旧/新防火墙才能正常工作。

设置 nftables 配置键后，新的 proxmox-firewall 服务会接管。可以通过检查 proxmox-firewall 的 systemctl 状态来确认新服务是否正常工作：

```
systemctl status proxmox-firewall
```

也可以检查生成的 ruleset。更多信息可在 常用命令一节中找到。还应检查 pve-firewall 是否不再生成 iptables 规则，相关命令可在 服务和命令一节中找到。

要切回旧防火墙，只需将配置值重新设置为 0 / No。

13.13.2 使用

proxmox-firewall 会创建两个由 proxmox-firewall 服务管理的表：proxmox-firewall 和 proxmox-firewall-guests。如果希望创建位于 Proxmox VE 防火墙配置之外的自定义规则，可以创建自己的表来管理自定义防火墙规则。proxmox-firewall 只会修改它生成的表，因此可以通过添加自己的表轻松扩展和修改 proxmox-firewall 的行为。

基于 nftables 的防火墙不使用 pve-firewall 命令，而是使用 proxmox-firewall。它是 systemd 服务，因此可以通过 systemctl 启动和停止：

```
systemctl start proxmox-firewall
systemctl stop proxmox-firewall
```

停止防火墙服务会移除所有生成的规则。

要查询防火墙状态，可以查询 systemctl 服务状态：

```
systemctl status proxmox-firewall
```

13.13.3 常用命令

可以通过以下命令检查生成的 ruleset：

```
nft list ruleset
```

如果要调试 proxmox-firewall，可以将 RUST_LOG 环境变量设置为 trace，并直接在前台 运行该守

```
RUST_LOG=trace /usr/libexec/proxmox/proxmox-firewall
```

如果希望防火墙守护进程提供详细输出，也可以编辑 systemctl 服务：

```
systemctl edit proxmox-firewall
```

然后需要为 RUST_LOG 环境变量添加 override：

```
[Service]
Environment="RUST_LOG=trace"
```

这会很快生成大量日志，因此仅应在调试时使用。其他较低详细程度的日志级别包括 info 和 debug。

前台运行会将日志输出写入 STDERR，因此可以使用以下命令重定向日志（例如用于向社区论坛提交日志）：

```
RUST_LOG=trace /usr/libexec/proxmox/proxmox-firewall 2> firewall_log_$( ↩
   hostname).txt
```

为调试防火墙规则，跟踪数据包在不同链之间的流动会很有帮助。可以通过为希望跟踪的数据包设置 `nftrace` 为 1 来实现。建议不要为*所有*数据包设置该标志；下面示例只检查 ICMP 数据包。

```
#!/usr/sbin/nft -f
table bridge tracebridge
delete table bridge tracebridge

table bridge tracebridge {
    chain trace {
        meta l4proto icmp meta nftrace set 1
    }

    chain prerouting {
        type filter hook prerouting priority -350; policy accept;
        jump trace
    }

    chain postrouting {
        type filter hook postrouting priority -350; policy accept;
        jump trace
    }
}
```

保存该文件、赋予可执行权限并运行一次后，会创建相应的跟踪链。然后可以通过 Proxmox VE Web UI（Firewall > Log）或 `nft monitor trace` 检查跟踪输出。

上述示例会跟踪所有桥上的流量，这通常是客户机流量经过的位置。如果要检查主机流量，请在 `inet` 表而不是 `bridge` 表中创建这些链。

Note
请注意，这可能生成*大量*日志，并显著降低网络栈性能。

可以运行以下命令移除跟踪规则：

```
nft delete table bridge tracebridge
```

Chapter 14

用户管理

Proxmox VE 支持多个认证源，例如 Linux PAM、集成的 Proxmox VE 认证服务器、LDAP、Microsoft Active Directory 和 OpenID Connect。

通过对所有对象（VM、存储、节点等）使用基于角色的用户和权限管理，可以定义细粒度访问控制。

14.1 用户

Proxmox VE 将用户属性存储在 `/etc/pve/user.cfg` 中。密码不存储在这里；用户会关联到下文描述的认证域。因此，在内部通常以 `<userid>@<realm>` 的形式， 通过用户名和域来标识用户。

该文件中的每个用户条目都包含以下信息：

- 名
- 姓
- 电子邮件地址
- 组成员关系
- 可选的过期日期
- 关于该用户的注释或备注
- 该用户是否启用或禁用
- 可选的双因素认证密钥

Caution
当禁用或删除用户，或者设置的过期日期已在过去时，该用户将无法登录新会话或启动新任务。该用户已经启动的所有任务（例如终端会话）都不会因这些事件而自动终止。

14.1.1 系统管理员

系统的 root 用户始终可以通过 Linux PAM 域登录，并且是不受限制的管理员。该用户不能删除，但仍可更改其属性。系统邮件会发送到分配给该用户的电子邮件地址。

14.2 组

每个用户可以属于多个组。组是组织访问权限的首选方式。应始终向组授予权限，而不是向单个用户授予权限。这样可以获得更易维护的访问控制列表。

14.3 API token

API token 允许其他系统、软件或 API 客户端以无状态方式访问 REST API 的大部分内容。 可以为单个用 token，并为其授予独立权限和过期日期，以限制访问范围和持续时间。如果 API token 泄露，可以撤销

API token 有两种基本类型：

- Separated privileges：需要通过 ACL 显式授予 token 访问权限。其有效权限通过用户权限与 token 权限的交集计算得出。
- Full privileges：token 权限与关联用户的权限完全相同。

Caution
token 值只会在生成 token 时显示/返回一次。之后无法再通过 API 取回！

使用 API token 时，请在发起 API 请求时将 HTTP header *Authorization* 设置为显示的值，格式为 `PVEAPIToken=USER@REALM!TOKENID=UUID`；也可以参考所用 API 客户端的文档。

14.4 资源池

资源池是一组虚拟机、容器和存储设备。在某些用户应受控访问特定资源集合的场景中，它非常适合处理权限，因为可以将单个权限应用到一组元素，而不必逐个资源管理。资源池通常与组配合使用，使组成员对一组机器和存储拥有权限。

14.5 认证域

由于 Proxmox VE 用户只是某些外部域中已有用户的对应项，因此必须在 `/etc/pve/domains.cfg` 中配置这些域。可用的域（认证方法）如下：

Linux PAM 标准认证
: Linux PAM 是用于系统范围用户认证的框架。这些用户通过 adduser 等命令在主机系统上创建。如果 Proxmox VE 主机系统中存在 PAM 用户，可以在 Proxmox VE 中添加对应条目，使这些用户系统用户名和密码登录。

Proxmox VE Authentication Server
: 这是一个类 Unix 的密码存储，会在 /etc/pve/priv/shadow.cfg 中存储哈希后的密码。密码 SHA-256 哈希算法处理。对于用户不需要访问 Proxmox VE 外部任何内容的小规模（甚至中等规这是最方便的域。在这种情况下，用户完全由 Proxmox VE 管理，并且可以通过 GUI 修改自己的

LDAP
: LDAP（Lightweight Directory Access Protocol）是一种开放、跨平台的协议，用于通过目录服务进行认证。OpenLDAP 是流行的开源 LDAP 协议实现。

Microsoft Active Directory（**AD**）
: Microsoft Active Directory（AD）是用于 Windows 域网络的目录服务，Proxmox VE 支持将其作为 认证域。它支持 LDAP 作为认证协议。

OpenID Connect
: OpenID Connect 是构建在 OAuth 2.0 协议之上的身份层。它允许客户端基于外部授权服务器执行的认证来验证用户身份。

14.5.1 Linux PAM 标准认证

由于 Linux PAM 对应主机系统用户，因此用户允许登录的每个节点上都必须存在系统用户。 用户使用其

通过 GUI 或等价的 /access/password API 端点修改密码时，只会应用到本地节点，而不是 集群范围 Proxmox VE 采用多主设计，为不同节点使用不同密码仍可带来安全收益。

在可配置性方面，管理员可以选择要求来自该域的登录使用双因素认证，也可以将该域设置为默认认证域。

14.5.2 Proxmox VE Authentication Server

Proxmox VE 认证服务器域是一个简单的类 Unix 密码存储。该域默认创建；与 Linux PAM 一样，可用的配置项只有：要求该域用户使用双因素认证，以及将其设置为默认登录域。

与其他 Proxmox VE 域类型不同，这类用户完全通过 Proxmox VE 创建和认证，而不是向另一个系统认因此，创建此类用户时必须为其设置密码。

14.5.3 LDAP

也可以使用外部 LDAP 服务器进行用户认证（例如 OpenLDAP）。在此域类型中，会在 *Base Domain Name* (base_dn) 下搜索用户，并使用 *User Attribute Name* (user_attr) 字段中指定的用户

可以配置服务器和可选备用服务器，并可通过 SSL 加密连接。此外，可以为目录和组配置过滤器。过滤器允许进一步限制该域的范围。

例如，如果用户由以下 LDIF 数据集表示：

```
# 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.
```

Base Domain Name 将是 ou=People,dc=ldap-test,dc=com，用户属性将是 uid。

如果 Proxmox VE 在查询和认证用户前需要先向 LDAP 服务器认证（bind），可以通过 /etc/pve/dc 中的 bind_dn 属性配置 bind domain name。其密码必须存储在 /etc/pve/priv/realm/<realm 中（例如 /etc/pve/priv/realm/my-ldap.pw）。 该文件应只包含一行原始密码。

要验证证书，需要设置 capath。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 证书的系统路径（/etc/ssl/certs）。此外，还需要设置 verify 选项，这也可以通过 Web 界面完成。

LDAP 服务器域的主要配置选项如下：

- Realm (realm): Proxmox VE 用户的域标识符
- Base Domain Name (base_dn): 搜索用户所在的目录
- User Attribute Name (user_attr): 包含用户登录用户名的 LDAP 属性
- Server (server1): 托管 LDAP 目录的服务器
- Fallback Server (server2): 可选备用服务器地址，用于主服务器不可达时
- Port (port): LDAP 服务器监听的端口

Note
要允许特定用户使用 LDAP 服务器认证，还必须从 Proxmox VE 服务器将其添加为该域的用户。 这可以通过 同步 自动完成。

14.5.4 Microsoft Active Directory（AD）

要将 Microsoft AD 设置为域，需要指定服务器地址和认证域。Active Directory 支持大多数与 LDAP 相同的属性，例如可选备用服务器、端口和 SSL 加密。此外，配置完成后，可以通过 同步 操作自动将用户添加到 Proxmox VE。

与 LDAP 一样，如果 Proxmox VE 在绑定到 AD 服务器前需要认证，必须配置 *Bind User* (bind_dn) 属性。对于 Microsoft AD，通常默认需要该属性。

Microsoft Active Directory 的主要配置设置如下：

- Realm (realm): Proxmox VE 用户的域标识符

- Domain (domain): 服务器的 AD 域
- Server (server1): 服务器的 FQDN 或 IP 地址
- Fallback Server (server2): 可选备用服务器地址，用于主服务器不可达时
- Port (port): Microsoft AD 服务器监听的端口

Note
Microsoft AD 通常以大小写不敏感方式检查用户名等值。要让 Proxmox VE 执行相同行为，可以在 Web UI 中编辑该域，或使用 CLI 禁用默认的 case-sensitive 选项（将 ID 替换为域 ID）：pveum realm modify ID --case-sensitive 0

14.5.5 同步基于 LDAP 的域

可以为基于 LDAP 的域（LDAP 和 Microsoft Active Directory）自动同步用户和组，而不必手动将它们添加到 Proxmox VE。可以从 Web 界面 Authentication 面板的 Add/Edit 窗口访问同步选项，也可以通过 pveum realm add/modify 命令访问。之后可以从 GUI 的 Authentication 面板执行同步操作，或使用以下命令：

```
pveum realm sync <realm>
```

用户和组会同步到集群范围配置文件 /etc/pve/user.cfg。

属性到属性映射

如果同步响应包含用户属性，它们会同步到 user.cfg 中匹配的用户属性。例如：firstname 或 lastname。

如果属性名称与 Proxmox VE 属性不匹配，可以使用 sync_attributes 选项在配置中设置自定义字段到字段映射。

如果某些内容消失，这些属性的处理方式可以通过同步选项控制，见下文。

同步配置

同步基于 LDAP 的域所需的配置选项，可以在 Add/Edit 窗口的 Sync Options 选项卡中找到。

配置选项如下：

- Bind User (bind_dn): 指用于查询用户和组的 LDAP 账号。该账号需要能够访问所有目标条目。如果设置了该项，搜索会通过绑定执行；否则搜索会匿名执行。该用户必须是完整的 LDAP 格式 distinguished name（DN），例如 cn=admin,dc=example,dc=com。
- Groupname attr. (group_name_attr): 表示用户所属的组。只会同步符合 user.cfg 常规字符限制的条目。为避免命名冲突，组同步时会在名称后附加 -$realm。请确保同步不会覆盖手动创建的组。
- User classes (user_classes): 与用户关联的对象类。

- Group classes (group_classes): 与组关联的对象类。
- E-Mail attribute: 如果基于 LDAP 的服务器指定了用户电子邮件地址，可以在此处设置关联属性，将其包含在同步中。从命令行可通过 --sync_attributes 参数实现。
- User Filter (filter): 用于定位特定用户的进一步过滤选项。
- Group Filter (group_filter): 用于定位特定组的进一步过滤选项。

Note
过滤器允许创建一组额外匹配条件，以缩小同步范围。可用 LDAP 过滤器类型及其用法的信息可在 ldap.com 查看。

同步选项

除上一节指定的选项外，还可以配置更多用于描述同步操作行为的选项。
这些选项可以在同步前作为参数设置，也可以通过域选项 sync-defaults-options 设置为默认值。
主要同步选项如下：

- Scope (scope): 要同步的范围。可以是 users、groups 或 both。
- Enable new (enable-new): 如果设置，新同步的用户会被启用并可登录。默认值为 true。
- Remove Vanished (remove-vanished): 这是一个选项列表。启用后，当同步响应未返回相应内容时，会决定是否移除它们。选项包括：
  - ACL (acl): 移除同步响应中未返回的用户和组的 ACL。它通常与 Entry 一起使用 最有意义。
  - Entry (entry): 当同步响应未返回条目（即用户和组）时移除这些条目。
  - Properties (properties): 当同步响应中的用户未包含某些属性时，移除条目的这些属性。这包括所有属性，即使它们从未由同步设置。例外是 token 和 enable 标志，即使启用此选项，它们也会保留。
- Preview (dry-run): 不向配置写入任何数据。如果想查看哪些用户和组会同步到 user.cfg，这会

保留字符

某些字符是保留字符（参见 RFC2253），如果未正确转义， 就不能轻易用于 DN 中的属性值。
以下字符需要转义：

- 开头或结尾的空格（ ）
- 开头的井号 (#)
- 逗号 (,)
- 加号 (+)
- 双引号 (”)

- 正斜杠 (/)
- 尖括号 (<>)
- 分号 (;)
- 等号 (=)

要在 DN 中使用这类字符，请将属性值放入双引号中。例如，要以 CN（Common Name）为 Example User 的用户进行绑定，请使用 CN="Example, User",OU=people,DC=example,DC=com 作为 bind_dn 的值。

这适用于 base_dn、bind_dn 和 group_dn 属性。

Note
带有冒号和正斜杠的用户无法同步，因为这些是用户名中的保留字符。

14.5.6 OpenID Connect

OpenID Connect 的主要配置选项如下：

- Issuer URL (issuer-url): 授权服务器的 URL。Proxmox VE 使用 OpenID Connect Discovery 协议自动配置更多细节。

  虽然可以使用未加密的 http:// URL，但强烈建议使用加密的 https:// 连接。
- Realm (realm): Proxmox VE 用户的域标识符
- Client ID (client-id): OpenID Client ID。
- Client Key (client-key): 可选 OpenID Client Key。
- Autocreate Users (autocreate): 如果用户不存在，则自动创建用户。虽然认证在 OpenID 服务器完成，但所有用户仍需要在 Proxmox VE 用户配置中有条目。可以手动添加，也可以使用 autocreate 选项自动添加新用户。
- Username Claim (username-claim): 用于生成唯一用户名的 OpenID claim（subject、username 或 email）。
- Autocreate Groups (groups-autocreate): 创建 claim 中的所有组，而不是使用现有 PVE 组（默认行为）。
- Groups Claim (groups-claim): 用于从 ID token 或 userinfo endpoint 获取组的 OpenID claim。
- Overwrite Groups (groups-overwrite): 覆盖分配给用户的所有组，而不是追加到现有组（默认行为）。

用户名映射

OpenID Connect 规范定义了一个名为 `subject` 的唯一属性（OpenID 术语中称为 *claim*）。默认情况下，我们使用该属性的值生成 Proxmox VE 用户名，方式是简单追加 `@` 和域名：`${subject}`

遗憾的是，大多数 OpenID 服务器会为 `subject` 使用随机字符串，例如 `DGH76OKH34BNG3245SB`，因 `DGH76OKH34BNG3245SB@yourrealm`。 虽然它是唯一的，但这类随机字符串很难记忆，也几乎无法

`username-claim` 设置允许使用其他属性进行用户名映射。如果 OpenID Connect 服务器提供 `username` 属性并保证其唯一性，建议将其设置为 `username`。

另一个选项是使用 `email`，这也会生成便于人类阅读的用户名。同样，只有在服务器保证该属性 唯一时

组映射

在 OpenID 配置中指定 `groups-claim` 设置会启用组映射功能。`groups-claim` 中提供的数据应是字符串列表，对应用户在 Proxmox VE 中应属于的组。为避免冲突，来自 OpenID claim 的组名会附加 `-<realm name>` 后缀（例如，域 `oidc` 中的 OpenID 组名 `my-openid-group`，在 Proxmox VE 中的组名会是 `my-openid-group-oidc`）。

默认情况下，OpenID provider 报告但在 Proxmox VE 中不存在的任何组都会被忽略。如果希望 OpenID provider 报告的所有组都存在于 Proxmox VE 中，可以使用 `groups-autocreate` 选项在用户登录时 自动创建这些组。

默认情况下，组会追加到用户现有组中。某些场景下，可能希望使用 OpenID provider 提供的组覆盖用户在 Proxmox VE 中已有的组。启用 `groups-overwrite` 设置后，会先从 Proxmox VE 中移除该用户的 所有组，再添加 OpenID provider 报告的组。

在某些情况下，OpenID 服务器可能发送包含 Proxmox VE 组 ID 无效字符的 groups claim。任何包含 Proxmox VE 组名不允许字符的组都不会被包含，并会向日志发送警告。

高级设置

- `Query userinfo endpoint` (`query-userinfo`): 启用该选项后，OpenID Connect 认证器需要查询 ”userinfo” endpoint 以获取 claim 值。对于某些不支持 ”userinfo” endpoint 的身份提供方（例如 ADFS），禁用该选项会很有用。

示例

以下是使用 Google 创建 OpenID 域的示例。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`，因此 Proxmox VE 端的用户名类似 `example.user@go`

Keycloak (https://www.keycloak.org/) 是流行的开源 Identity and Access Management 工具，支持 OpenID Connect。在以下示例中，需要将 `--issuer-url` 和 `--client-id` 替换为你的信息：

```
pveum realm add myrealm2 --type openid --issuer-url  https://your.server ↩
   :8080/realms/your-realm --client-id XXX --username-claim username
```

使用 `--username-claim username` 会在 Proxmox VE 端启用简单用户名，例如 `example.user@m`

Warning
需要确保用户不允许自行编辑用户名设置（在 Keycloak 服务器上）。

14.6 双因素认证

双因素认证有两种使用方式：

可以由认证域强制要求，方式可以是 *TOTP*（Time-based One-Time Password）或 *YubiKey OTP*。在这种情况下，新创建用户需要立即添加其密钥，因为没有第二因素就无法登录。 对于 *TOTP*，*TOTP*。

或者，即使域没有强制要求，用户也可以稍后自行选择启用双因素认证。

14.6.1 可用的第二因素

可以设置多个第二因素，以避免因丢失智能手机或安全密钥而永久无法访问账号。

除域强制的 TOTP 和 YubiKey OTP 外，还可使用以下双因素认证方法：

- 用户配置的 TOTP (Time-based One-Time Password). 它是由共享密钥和当前时间派生出的短代码 30 秒变化一次。
- WebAuthn (Web Authentication). 它是一种通用认证标准。各种安全设备都实现了该标准，例如硬 trusted platform module（TPM）。
- 一次性 Recovery Keys。这是一组密钥，应打印出来并锁在安全位置，或以数字形式保存在电子保险库中。每个密钥只能使用一次。即使所有其他第二因素丢失或损坏，它们也非常适合用于确保你不会被锁定在账号之外。

在支持 WebAuthn 之前，用户可以设置 U2F。现有 U2F 因素仍可继续使用，但建议在服务器配置 WebAuthn 后切换到 WebAuthn。

14.6.2 认证域强制双因素认证

添加或编辑 Authentication Realm 时，可以通过 *TFA* 下拉框选择一种可用方法来完成此设置。当域启用 TFA 后，它会成为强制要求，只有已配置 TFA 的用户才能登录。

当前有两种可用方法：

Time-based OATH (TOTP)

它使用标准 HMAC-SHA1 算法，将当前时间与用户配置的密钥一起 进行哈希处理。时间步长和密

用户可以配置多个密钥（用空格分隔），密钥可以用 Base32（RFC3548）或十六进制表示法指

Proxmox VE 提供密钥生成工具 (`oathkeygen`)，它会打印一个 Base32 表示法的随机密钥，可直 各种 OTP 工具，例如 `oathtool` 命令行工具，或 Android 上的 Google Authenticator、FreeOTP、andOTP 或类似应用。

YubiKey OTP

要通过 YubiKey 认证，必须配置 Yubico API ID、API KEY 和验证服务器 URL，用户也必须拥有可用的 YubiKey。要从 YubiKey 获取 key ID，可以通过 USB 连接后触发一次 YubiKey，并将输入密码的前 12 个字符复制到用户的 *Key IDs* 字段中。

关于如何使用 YubiCloud or host your own verification server， 请参考 YubiKey OTP 文档。

14.6.3 双因素认证限制与锁定

第二因素用于在用户密码泄露或被猜中时保护用户。不过，某些因素仍可能被暴力破解。因此， 当第二

对于 TOTP，失败 8 次会禁用用户的 TOTP 因素。使用恢复密钥登录时会解锁它们。如果 TOTP 是唯一可用因素，则需要管理员干预，并强烈建议要求用户立即修改密码。

由于 FIDO2/Webauthn 和恢复密钥较不容易受到暴力攻击，因此限制更高（100 次尝试），但超过限制后所有第二因素都会被阻止一小时。

管理员可以随时通过 UI 中的用户列表或命令行解锁用户的双因素认证：

```
pveum user tfa unlock joe@pve
```

14.6.4 用户配置的 TOTP 认证

用户可以通过用户列表中的 *TFA* 按钮，选择启用 *TOTP* 或 *WebAuthn* 作为登录第二因素（除非该域强 *YubiKey OTP*）。

用户始终可以添加并使用一次性 *Recovery Keys*。

打开 *TFA* 窗口后，用户会看到用于设置 *TOTP* 认证的对话框。*Secret* 字段包含密钥，可通过 *Randomize* 按钮随机生成。可以添加可选的 *Issuer Name*，用于向 *TOTP* 应用提供该密钥 归属信息。大多 *TOTP* 应用会同时显示 issuer name 和对应的 *OTP* 值。用户名也会包含在 *TOTP* 应用的二维码中。

生成密钥后，会显示一个二维码，可与 FreeOTP 等大多数 OTP 应用一起使用。然后，用户需要验证当前用户密码（除非以 *root* 登录），并通过在 *Verification Code* 字段中输入当前 *OTP* 值并按 *Apply* 按钮，验证其能够正确使用 *TOTP* 密钥。

14.6.5 TOTP

不需要服务器端设置。只需在智能手机上安装 TOTP 应用（例如 FreeOTP），并使用 Proxmox VE Web 界面添加 TOTP 因素。

14.6.6 WebAuthn

要让 WebAuthn 工作，需要满足两个条件：

- 受信任的 HTTPS 证书（例如使用 Let’s Encrypt）。 虽然使用不受信任证书也可能工作，但如果证书 WebAuthn 操作。

- 设置 WebAuthn 配置（参见 Proxmox VE Web 界面中的 **Datacenter → Options → WebAuthn Settings**）。在大多数设置中，这可以自动填充。

满足这两个要求后，可以在 **Datacenter → Permissions → Two Factor** 下的 **Two Factor** 面板中添加 WebAuthn 配置。

14.6.7 Recovery Keys

恢复密钥代码不需要任何准备；可以直接在 **Datacenter → Permissions → Two Factor** 下的 **Two Factor** 面板中创建一组恢复密钥。

> **Note**
> 每个用户在任意时刻只能拥有一组一次性恢复密钥。

14.6.8 服务器端 Webauthn 配置

要允许用户使用 *WebAuthn* 认证，必须使用带有效 SSL 证书的有效域名；否则某些浏览器可能警告或完全拒绝认证。

> **Note**
> 更改 *WebAuthn* 配置可能导致所有现有 *WebAuthn* 注册无法使用！

这通过 `/etc/pve/datacenter.cfg` 完成。例如：

```
webauthn: rp=mypve.example.com,origin=https://mypve.example.com:8006,id= ↩
   mypve.example.com
```

14.6.9 服务器端 U2F 配置

> **Note**
> 建议改用 WebAuthn。

要允许用户使用 *U2F* 认证，可能需要使用带有效 SSL 证书的有效域名；否则某些浏览器可能显示警告，或完全拒绝使用 U2F。首先，需要配置 *AppId* [1] 。

> **Note**
> 更改 *AppId* 会导致所有现有 *U2F* 注册无法使用！

这通过 `/etc/pve/datacenter.cfg` 完成。例如：

[1] AppId https://developers.yubico.com/U2F/App_ID.html

```
u2f: appid=https://mypve.example.com:8006
```

对于单个节点，*AppId* 可以简单设置为 Web 界面的地址，完全按照浏览器中使用的形式，包括 *https:/*
和端口，如上所示。请注意，在匹配 *AppIds* 时，某些浏览器可能比其他浏览器更严格。

使用多个节点时，最好有一个单独的 `https` 服务器提供 `appid.json` [2] 文件，因为这似乎与大多数浏
TLD 作为 *AppId* 可能已经足够。但需要注意，某些浏览器可能不接受这种做法。

Note
错误的 *AppId* 通常会产生错误，但我们也遇到过不报错的情况，尤其是在 Chromium 中，通过子域访问某个节点却为其使用顶级域 *AppId* 时。因此建议使用多个浏览器测试配置，因为之后更改 *AppId* 会导致现有 *U2F* 注册无法使用。

14.6.10 用户启用 U2F

要启用 *U2F* 认证，请打开 *TFA* 窗口的 *U2F* 选项卡，输入当前密码（除非以 root 登录），然后按 *Register* 按钮。如果服务器设置正确，并且浏览器接受服务器提供的 *AppId*，会出现 提示用户按下 *U2F* 设备按钮的消息（如果是 *YubiKey*，按钮灯应稳定地亮灭切换，大约每秒两次）。

Firefox 用户可能需要先通过 *about:config* 启用 *security.webauth.u2f*，才能使用 *U2F* token。

14.7 权限管理

为了让用户执行某个操作（例如列出、修改或删除 VM 配置的部分内容），该用户需要具备相应权限。

Proxmox VE 使用基于角色和路径的权限管理系统。权限表中的条目允许用户、组或 token 在访问某个 *object* 或 *path* 时承担特定角色。这意味着此类访问规则可以表示为 *(path, user, role)*、*(path, group, role)* 或 *(path, token, role)* 三元组，其中角色 包含一组允许的动作，而路径表示这些动作的目

14.7.1 角色

角色本质上是一组权限列表。Proxmox VE 随附多个预定义角色，可满足大多数需求。

- `Administrator`: 拥有全部权限
- `NoAccess`: 没有任何权限（用于禁止访问）
- `PVEAdmin`: 可以执行大多数任务，但无权修改系统设置 (`Sys.PowerMgmt`, `Sys.Modify`, `Realm.A` 或权限 (`Permissions.Modify`)
- `PVEAuditor`: 拥有只读访问权限
- `PVEDatastoreAdmin`: 创建并分配备份空间和模板

[2] Multi-facet apps: https://developers.yubico.com/U2F/App_ID.html

- PVEDatastoreUser: 分配备份空间并查看存储
- PVEMappingAdmin: 管理资源映射
- PVEMappingUser: 查看和使用资源映射
- PVEPoolAdmin: 分配池
- PVEPoolUser: 查看池
- PVESDNAdmin: 管理 SDN 配置
- PVESDNUser: 访问 bridge/vnet
- PVESysAdmin: 审计、系统控制台和系统日志
- PVETemplateUser: 查看和克隆模板
- PVEUserAdmin: 管理用户
- PVEVMAdmin: 完整管理 VM
- PVEVMUser: 查看、备份、配置 CD-ROM、VM 控制台、VM 电源管理

可以在 GUI 中查看完整的预定义角色集合。

可以通过 GUI 或命令行添加新角色。

在 GUI 中，从 *Datacenter* 导航到 *Permissions → Roles* 选项卡，然后点击 *Create* 按钮。可以在那里 *Privileges* 下拉菜单中选择所需权限。

要通过命令行添加角色，可以使用 *pveum* CLI 工具，例如：

```
pveum role add VM_Power-only --privs "VM.PowerMgmt VM.Console"
pveum role add Sys_Power-only --privs "Sys.PowerMgmt Sys.Console"
```

Note
以 PVE 开头的角色始终是内置角色，自定义角色不允许使用此前缀。

14.7.2 权限

权限是执行特定操作的权利。为简化管理，权限列表会组合成角色，然后可在权限表中使用。请注意，权限不能不经角色而直接分配给用户和路径。

当前支持以下权限：

节点/系统相关权限

- Group.Allocate: 创建/修改/移除组
- Mapping.Audit: 查看资源映射
- Mapping.Modify: 管理资源映射
- Mapping.Use: 使用资源映射

- Permissions.Modify: 修改访问权限
- Pool.Allocate: 创建/修改/移除池
- Pool.Audit: 查看池
- Realm.AllocateUser: 将用户分配到域
- Realm.Allocate: 创建/修改/移除认证域
- SDN.Allocate: 管理 SDN 配置
- SDN.Audit: 查看 SDN 配置
- Sys.Audit: 查看节点状态/配置、Corosync 集群配置和 HA 配置
- Sys.Console: 节点控制台访问
- Sys.Incoming: 允许来自其他集群的入站数据流（实验性）
- Sys.Modify: 创建/修改/移除节点网络参数
- Sys.PowerMgmt: 节点电源管理（start、stop、reset、shutdown 等）
- Sys.Syslog: 查看 syslog
- User.Modify: 创建/修改/移除用户访问和详细信息。

虚拟机相关权限

- SDN.Use: 访问 SDN vnet 和本地网络 bridge
- VM.Allocate: 在服务器上创建/移除 VM
- VM.Audit: 查看 VM 配置
- VM.Backup: 备份/还原 VM
- VM.Clone: 克隆/复制 VM
- VM.Config.CDROM: 弹出/更换 CD-ROM
- VM.Config.CPU: 修改 CPU 设置
- VM.Config.Cloudinit: 修改 Cloud-init 参数
- VM.Config.Disk: 添加/修改/移除磁盘
- VM.Config.HWType: 修改模拟硬件类型
- VM.Config.Memory: 修改内存设置
- VM.Config.Network: 添加/修改/移除网络设备
- VM.Config.Options: 修改任何其他 VM 配置
- VM.Console: VM 控制台访问
- VM.Migrate: 将 VM 迁移到集群中的其他服务器
- VM.Monitor: 访问 VM monitor (kvm)
- VM.PowerMgmt: 电源管理（start、stop、reset、shutdown 等）
- VM.Snapshot.Rollback: 将 VM 回滚到某个快照
- VM.Snapshot: 创建/删除 VM 快照

存储相关权限

- Datastore.Allocate: 创建/修改/移除 datastore，并删除卷

- Datastore.AllocateSpace: 在 datastore 上分配空间
- Datastore.AllocateTemplate: 分配/上传模板和 ISO 镜像
- Datastore.Audit: 查看/浏览 datastore

Warning
Permissions.Modify 和 Sys.Modify 都应谨慎处理，因为它们允许修改系统及其配置中危险或敏感的部分。

Warning
请仔细阅读下文关于继承的章节，以理解已分配角色（及其权限）如何沿 ACL 树传播。

14.7.3 对象和路径

访问权限会分配给对象，例如虚拟机、存储或资源池。我们使用类似文件系统的路径来寻址这些对象。这些路径形成一棵自然树，高层级（较短路径）的权限可以选择性地在此层级结构中向下传播。

路径可以模板化。当 API 调用需要某个模板化路径上的权限时，该路径可能包含对 API 调用参数的引用。这些引用用花括号指定。有些参数会从 API 调用的 URI 中隐式取得。例如，在调用 */nodes/mynode/status* 时，权限路径 /nodes/{node} 要求拥有 /nodes/mynode 上的权限； 而对 /access/acl 的 PUT 请求中的路径 {path} 则引用该方法的 path 参数。

一些示例如下：

- /nodes/{node}: 访问 Proxmox VE 服务器机器
- /vms: 覆盖所有 VM
- /vms/{vmid}: 访问特定 VM
- /storage/{storeid}: 访问特定存储
- /pool/{poolname}: 访问特定 池 中包含的资源
- /access/groups: 组管理
- /access/realms/{realmid}: 对域的管理访问

继承

如前所述，对象路径形成一棵类似文件系统的树，权限可以被树中下层对象继承（propagate 标志默认设置）。我们使用以下继承规则：

- 单个用户的权限始终替代组权限。
- 当用户属于某个组时，组权限会生效。
- 更深层级的权限会替代从上层继承的权限。
- NoAccess 会取消给定路径上的所有其他角色。

此外，权限分离 token 在任何给定路径上都不能拥有其关联用户没有的权限。

14.7.4 池

池可用于对一组虚拟机和 datastore 分组。之后只需在池 (/pool/{poolid}) 上设置权限，所有池成

14.7.5 我需要哪些权限？

每个单独方法所需的 API 权限都有文档记录，可在 https://pve.proxmox.com/pve-docs/api-viewer/ 找到。

权限以列表形式指定，可解释为逻辑和访问检查函数构成的树：

["and", <subtests>...] and ["or", <subtests>...]
当前列表中的后续元素必须全部（and）或任意一个（or）为真。

["perm", <path>, [<privileges>...], <options>...]
path 是模板化参数（参见 对象和路径）。指定路径上必须允许列出的 所有权限（或者，如果使用 any 选项，则允许任意一个权限）。如果指定了 require-param 选项，那么即使 API 调用的 schema 将其列为可选参数，该选项指定的参数也仍是必需的。

["userid-group", [<privileges>...], <options>...]
调用者必须在 /access/groups 上拥有列出的任意权限。此外，根据是否设置 groups_param 选项，有两种可能检查：

- 设置了 groups_param：API 调用有一个不可选的 groups 参数，调用者必须在列出的所有组上拥有列出的任意权限。
- 未设置 groups_param：通过 userid 参数传入的用户必须存在，并且必须属于某个组，调用者在该组上拥有列出的任意权限（通过 /access/groups/<group> 路径）。

["userid-param", "self"]
为 API 调用的 userid 参数提供的值必须指向执行该操作的用户（通常与 or 结合使用，允许用户在没有提升权限的情况下对自己执行操作）。

["userid-param", "Realm.AllocateUser"]
用户需要对 /access/realm/<realm> 拥有 Realm.AllocateUser 访问权限，其中 <realm> 指通过 userid 参数传入用户所属的域。请注意，用户无需已经存在即可与域关联，因为用户 ID 以 <username>@<realm> 形式传入。

["perm-modify", <path>]
path 是模板化参数（参见 对象和路径）。用户需要 Permissions.Modify 权限，或者根据路径

- /storage/...: 需要 *Datastore.Allocate*
- /vms/...: 需要 *VM.Allocate*
- /pool/...: 需要 *Pool.Allocate*
  如果路径为空，则需要 /access 上的 Permissions.Modify。
  如果用户没有 Permissions.Modify 权限，则只能在给定路径上委派自己权限的子集（例如，拥有 PVEVMAdmin 的用户可以分配 PVEVMUser，但不能分配 PVEAdmin）。

14.8 命令行工具

大多数用户会直接使用 GUI 管理用户。但也有一个功能完整的命令行工具，名为 pveum（“**P**roxmox **VE** **U**ser **M**anager” 的缩写）。请注意，所有 Proxmox VE 命令行工具都是 API 的包装器，因此也可以通过 REST API 访问这些功能。

下面是一些简单用法示例。要显示帮助，请输入：

```
pveum
```

或者（显示特定命令的详细帮助）：

```
pveum help user add
```

创建新用户：

```
pveum user add testuser@pve -comment "Just a test"
```

设置或更改密码（并非所有域都支持）：

```
pveum passwd testuser@pve
```

禁用用户：

```
pveum user modify testuser@pve -enable 0
```

创建新组：

```
pveum group add testgroup
```

创建新角色：

```
pveum role add PVE_Power-only -privs "VM.PowerMgmt VM.Console"
```

14.9 真实场景示例

14.9.1 管理员组

管理员可能希望创建一组拥有完整管理员权限的用户（而不使用 root 账号）。

为此，首先定义该组：

```
pveum group add admin -comment "System Administrators"
```

然后分配角色：

```
pveum acl modify / -group admin -role Administrator
```

最后，可以将用户添加到新的 *admin* 组：

```
pveum user modify testuser@pve -group admin
```

14.9.2 审计员

可以通过向用户或组分配 PVEAuditor 角色，授予只读访问权限。

示例 1：允许用户 joe@pve 查看所有内容

```
pveum acl modify / -user joe@pve -role PVEAuditor
```

示例 2：允许用户 joe@pve 查看所有虚拟机

```
pveum acl modify /vms -user joe@pve -role PVEAuditor
```

14.9.3 委派用户管理

如果希望将用户管理委派给用户 joe@pve，可以使用：

```
pveum acl modify /access -user joe@pve -role PVEUserAdmin
```

用户 joe@pve 现在可以添加和移除用户，并更改其他用户属性，例如密码。这是一个非常强大的角色，通常应将其限制在选定域和组内。以下示例允许 joe@pve 修改 pve 域内且属于 customers 组的用户：

```
pveum acl modify /access/realm/pve -user joe@pve -role PVEUserAdmin
pveum acl modify /access/groups/customers -user joe@pve -role PVEUserAdmin
```

Note
该用户能够添加其他用户，但仅限这些用户属于 customers 组且位于 pve 域内。

14.9.4 用于监控的受限 API token

API token 上的权限始终是其对应用户权限的子集，这意味着 API token 不能用于执行其背后用户无权执行的任务。本节演示如何使用具有独立权限的 API token，进一步限制 token 所有者的权限。

授予用户 joe@pve 对所有 VM 的 PVEVMAdmin 角色：

```
pveum acl modify /vms -user joe@pve -role PVEVMAdmin
```

添加一个具有独立权限的新 API token，只允许查看 VM 信息（例如用于监控）：

```
pveum user token add joe@pve monitoring -privsep 1
pveum acl modify /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 资源池

企业通常由多个较小部门组成，经常需要为每个部门分配资源并委派管理任务。假设要为软件开发部门设置一个池。首先创建一个组：

```
pveum group add developers -comment "Our software developers"
```

现在创建一个属于该组的新用户：

```
pveum user add developer1@pve -group developers -password
```

> **Note**
> "-password" 参数会提示输入密码。

然后创建供开发部门使用的资源池：

```
pveum pool add dev-pool --comment "IT development pool"
```

最后，可以为该池分配权限：

```
pveum acl modify /pool/dev-pool/ -group developers -role PVEAdmin
```

现在，软件开发人员可以管理分配给该池的资源。

Chapter 15

高可用性

现代社会高度依赖由计算机通过网络提供的信息。移动设备进一步放大了这种依赖，因为人们可以随时
从数学上讲，可用性可以定义为 (A) 服务在给定时间区间内可被使用的总时间，与 (B) 该时间区间长度

Table 15.1: 可用性 - 每年停机时间

可用性 **%**	每年停机时间
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

提高可用性有多种方式。最优雅的方案是重写软件，使其能够同时在多台主机上运行。软件本身需要具
这相对简单。不过，这通常很复杂，有时甚至不可能，因为无法自行修改软件。以下方案无需修改软件

- 使用可靠的“服务器”组件

> **Note**
> 功能相同的计算机组件可能因为组件质量不同而具有不同的可靠性指标。
> 多数厂商会将可靠性更高的组件作为“服务器”组件销售，通常价格也更高。

- 消除单点故障（冗余组件）
  - 使用不间断电源（UPS）
  - 在服务器中使用冗余电源
  - 使用 ECC-RAM
  - 使用冗余网络硬件
  - 对本地存储使用 RAID

  - 为虚拟机数据使用分布式冗余存储

- 减少停机时间

  - 管理员可快速响应（24/7）
  - 具备备件可用性（Proxmox VE 集群中的其他节点）
  - 自动错误检测（由 ha-manager 提供）
  - 自动故障切换（由 ha-manager 提供）

像 Proxmox VE 这样的虚拟化环境让实现高可用性容易得多，因为它们消除了对“硬件”的依赖。它们还支持配置和使用冗余存储及网络设备，因此如果一台主机发生故障，可以直接在集群内另一台主

更进一步，Proxmox VE 提供了名为 ha-manager 的软件栈，可以自动完成这些工作。它能够自动检测

Proxmox VE ha-manager 的工作方式类似一位“自动化”的管理员。首先，配置它应管理哪些资源（虚拟机、容器等）。然后，ha-manager 会监控其正确运行状态，并在发生错误时将服务故障切换到 ha-manager 也可以处理普通用户请求，例如启动、停止、重定位和迁移服务。

但高可用性是有成本的。高质量组件更昂贵，而将它们冗余化至少会使成本翻倍。额外备件会进一步增

Tip
将可用性从 99% 提高到 99.9% 相对简单。但从 99.9999% 提高到 99.99999% 非常困难且成本高昂。ha-manager 的典型错误检测和故障切换时间约为 2 分钟，因此最多只能获得约 99.999% 的可用性。

15.1 要求

开始使用 HA 之前，必须满足以下要求：

- 至少三个集群节点（以获得可靠的 quorum）
- 用于虚拟机和容器的共享存储
- 硬件冗余（全范围）
- 使用可靠的“服务器”组件
- 硬件 watchdog - 如果不可用，则回退到 Linux 内核软件 watchdog（softdog）
- 可选的硬件 fencing 设备

15.2 资源

ha-manager 处理的主要管理单元称为资源。资源（也称为“服务”）由服务 ID（SID）唯一标识，SID 由资源类型和类型特定 ID 组成，例如 vm:100。该示例表示一个类型为 vm（虚拟机）、ID 为 100 的资源。

目前有两类重要资源类型：虚拟机和容器。这里的基本思路之一是，可以将相关软件打包到这类 虚拟机 rgmanager 那样由其他服务组合出一个大服务。一般来说，由 HA 管理的资源不应依赖其他资源。

15.3 管理任务

本节简要概述常见管理任务。第一步是为资源启用 HA。做法是将资源添加到 HA 资源配置中。 可以通过 GUI 完成，也可以直接使用命令行工具，例如：

```
# ha-manager add vm:100
```

HA 栈现在会尝试启动资源并保持其运行。请注意，可以配置资源的“请求”状态。 例如，可能希望 HA 栈停止该资源：

```
# ha-manager set vm:100 --state stopped
```

稍后再重新启动：

```
# ha-manager set vm:100 --state started
```

也可以使用普通的虚拟机和容器管理命令。它们会自动将命令转发给 HA 栈，因此

```
# qm start 100
```

只是将请求状态设置为 `started`。`qm stop` 同样如此，它会将请求状态设置为 `stopped`。

Note
HA 栈完全异步工作，并且需要与其他集群成员通信。因此，看到这些操作的结果需要几秒钟。

要查看当前 HA 资源配置，请使用：

```
# ha-manager config
vm:100
        state stopped
```

可以使用以下命令查看实际的 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)
```

也可以发起资源向其他节点迁移：

```
# ha-manager migrate vm:100 node2
```

这会使用在线迁移，并尝试保持虚拟机运行。在线迁移需要通过网络传输所有已使用内存， 因此有时先 `relocate` 命令完成：

```
# ha-manager relocate vm:100 node2
```

最后，可以使用以下命令从 HA 配置中移除资源：

```
# ha-manager remove vm:100
```

Note
这不会启动或停止该资源。

不过，所有 HA 相关任务都可以在 GUI 中完成，因此完全不需要使用命令行。

常用命令示例：

```
ha-manager status
ha-manager config
ha-manager set vm:100 --state started
```

15.4 工作原理

本节详细说明 Proxmox VE HA 管理器的内部机制。它描述所有相关守护进程以及它们如何协同工作。为了提供 HA，每个节点上会运行两个守护进程：

pve-ha-lrm
本地资源管理器（LRM），用于控制在本地节点上运行的服务。它从当前 manager status 文件中读取其服务的请求状态，并执行相应命令。

pve-ha-crm
集群资源管理器（CRM），用于做出集群范围内的决策。它向 LRM 发送命令、处理结果，并在发生故障时将资源移动到其他节点。CRM 还负责节点 fencing。

Note
锁由分布式配置文件系统（pmxcfs）提供。它们用于保证每个 LRM 只活动一次并正常工作。由于 LRM 只有在持有自己的锁时才会执行操作，如果能够取得某个失败节点的锁，就可以将该节点标记为已 fenced。这样便可安全恢复任何失败的 HA 服务，而不会受到当前状态未知的失败节点干扰。所有这些都由当前持有 manager master 锁的 CRM 监督。

15.4.1 服务状态

CRM 使用服务状态枚举来记录当前服务状态。该状态会显示在 GUI 中，也可以使用 ha-manager 命令行工具查询：

```
# 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)
```

可能的状态如下：

stopped
服务已停止（由 LRM 确认）。如果 LRM 检测到标记为停止的服务仍在运行，它会再次停止该服务

request_stop
服务应被停止。CRM 等待 LRM 确认。

stopping
停止请求待处理。但 CRM 目前尚未获得该请求的结果。

started
服务处于活动状态，如果尚未运行，LRM 应尽快启动它。如果服务失败并被检测为未运行，LRM 会重启它（见 启动失败策略）。

starting
启动请求待处理。但 CRM 尚未从 LRM 收到服务正在运行的确认。

fence
等待节点 fencing，因为服务所在节点不在具备 quorum 的集群分区内（见 Fencing）。一旦节点 fenced，服务会被置于 recovery 状态。

recovery
等待服务恢复。HA 管理器会尝试寻找一个可运行该服务的新节点。该搜索不仅取决于在线且具备 quorum 的节点列表，也取决于服务是否为某个组的成员，以及该组如何受限。 一旦找到新的可用 stopped 状态。 如果配置为运行，新节点会启动它。

freeze
不要修改服务状态。在重启节点或重启 LRM 守护进程时会使用该状态（见 软件包更新）。

ignored
表现得像该服务完全不由 HA 管理。当需要临时完全控制服务、但不想将其从 HA 配置中移除时，该状态很有用。

migrate
将服务（实时）迁移到其他节点。

error
服务因 LRM 错误而被禁用。需要人工干预（见 错误恢复）。

queued
服务是新添加的，CRM 目前尚未看到它。

disabled
服务已停止，并被标记为 `disabled`

15.4.2 本地资源管理器

本地资源管理器（`pve-ha-lrm`）会在启动时作为守护进程启动，并等待 HA 集群具备 quorum，从而确保集群范围锁可以工作。

它可以处于三种状态：

wait for agent lock
LRM 等待其独占锁。如果未配置任何服务，这也用作空闲状态。

active
LRM 持有其独占锁，并且已配置服务。

lost agent lock
LRM 丢失了锁，这表示发生了故障并丢失了 quorum。

LRM 进入 active 状态后，会读取 /etc/pve/ha/manager_status 中的 manager status 文件，并确定需要为其拥有的服务执行哪些命令。每个命令都会启动一个 worker，这些 worker 并行运行，默认最多限制为 4 个。该默认设置可以通过数据中心配置键 max_worker 修改。完成后，w 进程会被回收，其结果会保存给 CRM。

Note
最多 4 个并发 worker 的默认值可能不适合特定环境。例如，可能同时发生 4 次实时迁移，在较慢网络和/或大型（以内存计）服务的情况下会导致网络拥塞。还应确保即使在最坏情况下，拥塞也保持在最低水平，即使这意味着降低 max_worker 值。相反，如果环境特别强大、属于高端配置，也可以考虑提高该值。

CRM 请求的每个命令都可以通过 UID 唯一识别。worker 完成后，其结果会被处理并写入 LRM 状态文件 /etc/pve/nodes/<nodename>/lrm_status。CRM 可以在那里收集结果，并让其状态机

CRM 与 LRM 之间对每个服务的操作通常始终保持同步。这意味着 CRM 请求一个由 UID 唯一标记的状态随后 只执行一次 该操作，并写回同样可由该 UID 识别的结果。 这是为了避免 LRM 执行过期命令。该行 stop 和 error 命令例外； 这两者不依赖产生的结果，在 stopped 状态下总是执行，在 error 状态下执行一次。

Note
HA 栈会记录它执行的每个操作。这有助于理解集群中发生了什么以及为什么发生。这里重要的是查看 LRM 和 CRM 两个守护进程分别做了什么。可以在服务所在节点上使用 journalctl -u pve-ha-lrm，并在当前 master 节点上对 pve-ha-crm 使用相同命令。

15.4.3 集群资源管理器

集群资源管理器（pve-ha-crm）会在每个节点上启动，并等待 manager lock，该锁同一时间只能由 manager lock 的节点会被提升为 CRM master。

它可以处于三种状态：

wait for agent lock
CRM 等待其独占锁。如果未配置任何服务，这也用作空闲状态。

active
CRM 持有其独占锁，并且已配置服务。

lost agent lock
CRM 丢失了锁，这表示发生了故障并丢失了 quorum。

其主要任务是管理配置为高可用的服务，并始终尝试强制满足请求状态。例如， 请求状态为 started 的服务如果尚未运行，就会被启动。如果它崩溃， 则会自动再次启动。因此，CRM 会指示 LRM 需要执行的操作。

当某个节点离开集群 quorum 时，其状态会变为 unknown。如果当前 CRM 随后能够取得失败节点的锁，服务会被“窃取”并在另一节点上重启。

当某个集群成员确定自己不再处于集群 quorum 中时，LRM 会等待形成新的 quorum。 在存在集群 quorum 之前，该节点无法重置 watchdog。如果节点上有活动服务， 或者 LRM 或 CRM 进程未被调度， watchdog 超时后触发重启（这会在 60 秒后发生）。

请注意，如果某节点有活动 CRM 但 LRM 处于空闲状态，quorum 丢失不会触发 self-fence reset。 原因是 CRM 访问的所有状态文件和配置都由 集群配置文件系统 支撑， 而该文件系统会在 quorum 丢失时变为只读。这意味着 CRM 只需要防止自身进程长时间得不到调度； 否则另一个 CRM 可能在不了解情况的情况下接管，从而破坏 HA 状态。已打开的 watchdog 确保这种情况不会发生。

如果超过 15 分钟未配置任何服务，CRM 会自动返回空闲状态，并完全关闭 watchdog。

15.5 HA 模拟器

使用 HA 模拟器，可以测试并学习 Proxmox VE HA 方案的所有功能。

默认情况下，模拟器允许观察并测试一个真实场景中的 3 节点集群和 6 台虚拟机的行为。 也可以添加或

无需设置或配置真实集群，HA 模拟器开箱即可运行。

使用 dnf 安装：

```
dnf install pve-ha-simulator
```

甚至可以在没有任何其他 Proxmox VE 软件包的 Debian 系统上安装该软件包。 为此，需要下载该软件包并从本地文件系统使用 dnf 安装该软件包时，dnf 也会为你解析所需依赖。

要在远程机器上启动模拟器，必须将 X11 重定向到当前系统。

如果使用 Linux 机器，可以使用：

```
ssh root@<IPofPVE> -Y
```

在 Windows 上，可以使用 mobaxterm。

连接到已安装模拟器的现有 Proxmox VE，或在本地 Debian 系统上手动安装后， 可以按如下方式试用

首先需要创建一个工作目录，模拟器会在其中保存当前状态并写入默认配置：

```
mkdir working
```

然后，只需将创建的目录作为参数传递给 pve-ha-simulator：

```
pve-ha-simulator working/
```

随后可以启动、停止、迁移模拟的 HA 服务，甚至可以查看节点故障时会发生什么。

15.6 配置

HA 栈与 Proxmox VE API 深度集成。因此，例如可以通过 ha-manager 命令行界面或 Proxmox VE Web 界面配置 HA，这两种界面都提供了简单的 HA 管理方式。自动化工具可以直接使用 API。

所有 HA 配置文件都位于 /etc/pve/ha/ 中，因此会自动分发到集群节点，并由所有节点共享同一份 HA 配置。

15.6.1 资源

资源配置文件 /etc/pve/ha/resources.cfg 存储由 ha-manager 管理的资源列表。该列表中的资

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

它以资源类型开头，后接资源特定名称，中间用冒号分隔。两者共同组成 HA 资源 ID，所有 ha-manag 命令都使用该 ID 唯一标识资源（例如 vm:100 或 ct:101）。后续行包含其他属性：

comment: <string>
: 说明。

group: <string>
: HA 组标识符。

max_relocate: <integer> (0 - N) (*default* = 1)
: 服务启动失败时，尝试重定位服务的最大次数。

max_restart: <integer> (0 - N) (*default* = 1)
: 服务在节点上启动失败后，尝试重启该服务的最大次数。

state: <disabled | enabled | ignored | started | stopped> (*default* = started)
: 请求的资源状态。CRM 读取此状态并据此执行操作。请注意，enabled 只是 started 的别名。

  started
  : CRM 会尝试启动资源。成功启动后，服务状态会设置为 started。当节点故障或启动失败时 error。

  stopped
  : CRM 会尝试将资源保持在 stopped 状态，但在节点故障时仍会尝试重定位资源。

  disabled
  : CRM 会尝试将资源置于 stopped 状态，但在节点故障时不会尝试重定位资源。此状态的主要 error 状态的唯一方式。

  ignored
  : 该资源会从管理器状态中移除，因此 CRM 和 LRM 不再处理该资源。所有影响此资源的 {pve} API 调用都会直接执行，并绕过 HA 栈。当资源处于此状态时，CRM 命令会被丢弃。

下面是一个包含一台虚拟机和一个容器的真实示例。可以看到，这些文件的语法非常简单，甚至可以使

配置示例（**/etc/pve/ha/resources.cfg**）

```
vm: 501
    state started
    max_relocate 2

ct: 102
    # Note: use default settings for everything
```

上述配置由 ha-manager 命令行工具生成：

```
# ha-manager add vm:501 --state started --max_relocate 2
# ha-manager add ct:102
```

15.6.2 组

HA 组配置文件 /etc/pve/ha/groups.cfg 用于定义集群节点组。可以将资源限制为仅在这类组的成
组配置如下所示：

```
group: <group>
       nodes <node_list>
       <property> <value>
       ...
```

comment: <string>
说明。

nodes: <node>[:<pri>]{,<node>[:<pri>]}*
集群节点成员列表，可以为每个节点指定优先级。绑定到某个组的资源会在优先级最高的可用节点

nofailback: <boolean> (*default* = 0)
CRM 会尝试在优先级最高的节点上运行服务。如果优先级更高的节点上线，CRM 会将服务迁移到
nofailback 可阻止这种行为。

restricted: <boolean> (*default* = 0)
绑定到受限组的资源只能在该组定义的节点上运行。如果没有任何组成员节点在线，资源会被置于
stopped 状态。绑定到非受限组的资源在所有组成员都离线时可以在任意集群节点上运行，但只要

常见需求是让某个资源运行在特定节点上。通常该资源也能够在其他节点上运行， 因此可以定义一个只

```
# ha-manager groupadd prefer_node1 --nodes node1
```

对于较大的集群，定义更详细的故障切换行为是有意义的。例如，可能希望在可行时将一组服务运行在
node1 上。如果 node1 不可用，则希望它们平均分布在 node2 和 node3 上。 如果这些节点也失败，
node4 上。为实现这一点，可以将节点列表设置为：

```
# ha-manager groupadd mygroup1 -nodes "node1:2,node2:1,node3:1,node4"
```

另一种用例是，某个资源依赖仅在特定节点上可用的其他资源，例如 node1 和 node2。 需要确保
HA 管理器不会使用其他节点，因此需要使用这些节点创建一个受限组：

```
# ha-manager groupadd mygroup2 -nodes "node1,node2" -restricted
```

上述命令创建了以下组配置文件：

配置示例（**/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
```

nofailback 选项主要用于在管理任务期间避免不必要的资源移动。例如，如果需要将服务迁移到组中
nofailback 选项告诉 HA 管理器不要立即将该服务移回去。

另一种场景是某个服务被 fenced 后恢复到另一节点。管理员尝试修复被 fenced 的节点，并重新将其上
nofailback 标志可防止 已恢复的服务直接迁回被 fenced 的节点。

15.7 Fencing

节点故障时，fencing 确保出错节点一定处于离线状态。这是为了确保资源在另一节点上恢复时不会同时
这是一项非常重要的任务，因为没有它，就无法在另一节点上恢复资源。

如果某个节点没有被 fenced，它会处于未知状态，可能仍然能够访问共享资源。 这非常危险。设想除了
但仍在运行并写入共享存储。

如果此时简单地在另一节点上启动该虚拟机，就会出现危险的竞争条件，因为两个节点都会写入。
这种情况可能破坏所有虚拟机数据，并导致整个虚拟机不可用。如果存储防止多重挂载，恢复也可能失

15.7.1 Proxmox VE 如何执行 Fencing

对节点执行 fencing 有多种方法，例如使用 fence 设备切断节点电源，或完全禁用其通信。 这些设备通

因此，Proxmox VE 集成了一种更简单的 fencing 方法，不需要额外的外部硬件。 这可以通过 watchdog 定时器实现。

可能的 Fencing 方法

- 外部电源开关
- 通过在交换机上完全禁用网络流量来隔离节点
- 使用 watchdog 定时器执行 self fencing

从微控制器出现之初，watchdog 定时器就已广泛用于关键且可靠性要求高的系统中。 它们通常是简单

正常运行期间，ha-manager 会定期重置 watchdog 定时器，防止其超时。 如果由于硬件故障或程序错
watchdog，定时器就会超时， 并触发整个服务器重置（重启）。

较新的服务器主板通常包含这类硬件 watchdog，但需要进行配置。如果没有可用或已配置的 watch-
dog，则回退到 Linux 内核 softdog。它虽然仍然可靠，但并不独立于服务器硬件， 因此可靠性低于硬
watchdog。

15.7.2 配置硬件 Watchdog

默认情况下，出于安全原因，所有硬件 watchdog 模块都会被阻止。如果没有正确初始化， 它们就像一
watchdog，需要在 /etc/default/pve-ha-manager 中指定要加载的模块，例如：

```
# select watchdog module (default is softdog)
WATCHDOG_MODULE=iTCO_wdt
```

该配置由 watchdog-mux 服务读取，该服务会在启动时加载指定模块。

15.7.3 恢复已 Fenced 的服务

节点失败并成功完成 fencing 后，CRM 会尝试将服务从失败节点移动到仍在线的节点。

这些服务恢复到哪些节点，会受资源 group 设置、当前活动节点列表以及各节点活动服务数量影响。

CRM 首先根据用户选择的节点（来自 group 设置）与可用节点的交集构建集合。 然后选择优先级最高
这可以尽量降低节点过载的可能性。

Caution
节点故障时，CRM 会将服务分发到剩余节点。这会增加这些节点上的服务数量，
并可能导致高负载，尤其是在小型集群中。请设计集群，使其能够处理这类最坏情况。

15.8 启动失败策略

如果服务在某个节点上一次或多次启动失败，启动失败策略就会生效。它可用于配置在同一节点上
应触发多少次重启，以及服务应重定位多少次，以便尝试在另一节点上启动。 该策略的目标是规避特定
某个具备 quorum 的节点不再能够访问共享存储，但其他节点仍可访问， 则 relocate 策略仍允许服务

有两个可针对每个资源单独配置的服务启动恢复策略设置。

max_restart
在实际节点上重启失败服务的最大尝试次数。默认值为一。

max_relocate
将服务重定位到其他节点的最大尝试次数。只有在实际节点上超过 max_restart 值后， 才会发生
relocate。默认值为一。

Note
只有服务至少成功启动过一次，relocate 计数状态才会重置为零。
这意味着如果未修复错误就重新启动服务，只会重复执行重启策略。

15.9 错误恢复

如果在所有尝试之后仍无法恢复服务状态，它会被置于 error 状态。在该状态下， HA 栈不再触碰该服务

```
# ha-manager set vm:100 --state disabled
```

这也可以在 Web 界面中完成。

要从 error 状态恢复，应执行以下操作：

- 将资源恢复到安全且一致的状态（例如：如果服务无法停止，则 kill 其进程）
- 禁用资源以移除 error 标志
- 修复导致这些失败的错误
- 在修复所有错误 之后，才可以请求再次启动该服务

15.10 软件包更新

更新 ha-manager 时，应逐个节点更新，切勿一次性全部更新，原因有多个。首先， 尽管项目会全面测
bug。逐个节点更新， 并在每个节点更新完成后检查其功能，有助于从潜在问题中恢复；而一次性全部
通常也不是良好实践。

此外，Proxmox VE HA 栈使用请求确认协议在集群和本地资源管理器之间执行操作。重启时， LRM
会向 CRM 请求冻结其所有服务。这可以防止在 LRM 重启的短时间内这些服务被集群触碰。 随后，LRM
可以在重启期间安全关闭 watchdog。这类重启通常发生在软件包更新期间； 如前所述，需要一个活动
master CRM 来确认来自 LRM 的请求。如果不是这样， 更新过程可能耗时过长，在最坏情况下可能导
watchdog 触发重置。

15.11 节点维护

有时需要对节点执行维护，例如更换硬件或只是安装新的内核镜像。使用 HA 栈时同样如此。

HA 栈主要可以支持两类维护：

- 对于常规关机或重启，可以配置其行为，见 关机策略。
- 对于不需要关机或重启的维护，或不应在仅一次重启后自动关闭的维护， 可以启用手动维护模式。

15.11.1 维护模式

可以使用手动维护模式将节点标记为不适用于 HA 操作，从而促使 HA 管理的所有服务迁移到其他节点。

这些迁移的目标节点会从当前其他可用节点中选择，并由 HA 组配置和已配置的集群资源调度器（CRS
HA 管理器状态中， 以便在维护模式禁用且节点重新上线后，可以自动将服务移回。

目前可以使用 ha-manager CLI 工具启用或禁用维护模式。

为节点启用维护模式

```
# ha-manager crm-command node-maintenance enable NODENAME
```

这会将一个 CRM 命令加入队列，当管理器处理该命令时，会在 manager status 中记录维护模式请求。这允许在任意节点上提交该命令，而不必只在要进入或退出维护模式的节点上执行。

一旦相应节点上的 LRM 接收到该命令，它会将自身标记为不可用，但仍会处理所有迁移命令。 这意味着 LRM self-fencing watchdog 会保持活动，直到所有活动服务都已移动且所有正在运行的 worker 都已完成。

请注意，只要 LRM 接收到请求状态，LRM 状态就会显示 `maintenance` 模式， 而不是等到所有服务都目前，可以检查节点上是否仍有活动 HA 服务，或留意类似 `pve-ha-lrm[PID]: watchdog closed (disabled)` 的日志行，以判断节点何时完成进入维护模式的转换。

> **Note**
> 手动维护模式不会在节点重启时自动删除，只有通过 `ha-manager` CLI 手动停用， 或手动清除 manager-status 时才会删除。

为节点禁用维护模式

```
# ha-manager crm-command node-maintenance disable NODENAME
```

禁用手动维护模式的过程与启用类似。使用上面所示的 `ha-manager` CLI 命令会将一个 CRM 命令加入队列，LRM 节点会再次标记为可用。

如果停用维护模式，在维护模式激活时位于该节点上的所有服务都会被移回。

15.11.2 关机策略

下面描述节点关机时的不同 HA 策略。当前出于向后兼容性考虑，`Conditional` 是默认值。 部分用户可能 `Migrate` 的行为更符合预期。

关机策略可以在 Web UI（`Datacenter` → `Options` → `HA Settings`）中配置， 也可以直接在 `datacenter.cfg` 中配置：

```
ha: shutdown_policy=<value>
```

Migrate

本地资源管理器（LRM）收到关机请求且启用该策略后，会将自身标记为对当前 HA 管理器不可用。这会触发当前位于该节点上的所有 HA 服务迁移。LRM 会尝试延迟关机过程， 直到所有正在运行的服务可以 迁移到另一节点。 换言之，服务不能绑定在本地，例如不能使用硬件直通。由于在没有可用组成员非组成员节点会被视为可运行目标，因此即使使用只选择部分节点的 HA 组， 仍可使用该策略。但是，`restricted` 会告诉 HA 管理器该服务不能在所选节点集合之外运行。 如果所有这些节点都不可用，关之前被迁出的服务会被移回，除非它们在此期间已经被手动迁移。

Note
关机迁移过程中 watchdog 仍保持活动。如果节点丢失 quorum，它会被 fenced，服务也会被恢复。

如果在当前正在维护的节点上启动一个（此前已停止的）服务，则需要对该节点执行 fencing，以确保

Failover

该模式确保所有服务都会停止；如果当前节点短时间内没有重新上线，这些服务也会被恢复。 在集群规
但仍希望确保 HA 服务尽快恢复并重新启动。

Freeze

该模式确保所有服务都被停止并冻结，因此直到当前节点重新上线之前，它们不会被恢复。

Conditional

`Conditional` 关机策略会自动检测请求的是关机还是重启，并相应改变行为。

关机

如果计划让节点停机一段时间，通常会执行关机（`poweroff`）。在这种情况下，LRM 会停止所有受管

Note
现代硬件通常拥有大量内存（RAM）。因此，系统会停止所有资源，然后重新启动它们，以避免在线迁移全部 RAM。如果要使用在线迁移，需要在关闭节点前手动调用。

重启

节点重启通过 `reboot` 命令发起。这通常在安装新内核后执行。请注意，这不同于“关机”，因为节点会

LRM 会告知 CRM 它希望重启，并等待 CRM 将所有资源置于 `freeze` 状态（软件包更新 使用同一机制
这可以防止这些资源被移动到其他节点。相反，CRM 会在重启后在同一节点上启动这些资源。

手动资源移动

最后，也可以在关闭或重启节点之前，手动将资源移动到其他节点。其优势是拥有完全控制权，并且可

Note
请不要 `kill pve-ha-crm、pve-ha-lrm 或 watchdog-mux` 等服务。 它们管理并使用 watchdog，因此这样做可能导致节点立即重启甚至重置。

15.12 集群资源调度

集群资源调度器（CRS）模式控制 HA 如何为服务恢复以及由关机策略触发的迁移选择节点。 默认模式
basic，可以在 Web UI（Datacenter → Options）中更改， 也可以直接在 datacenter.cfg
中更改：

```
crs: ha=static
```

该变更会从下一轮 manager 运行开始生效（几秒钟后）。

对于每个需要恢复或迁移的服务，调度器会在该服务所属组中优先级最高的节点之间迭代选择最佳节点

Note
未来计划添加（静态和动态）负载均衡模式。

15.12.1 Basic 调度器

每个节点上的活动 HA 服务数量用于选择恢复节点。当前不统计非 HA 管理的服务。

15.12.2 Static-Load 调度器

Important
static 模式仍是技术预览。

每个节点上 HA 服务的静态使用信息用于选择恢复节点。当前不考虑非 HA 管理服务的使用情况。

在此选择过程中，会依次将每个节点视为该服务已经在其上运行，并使用关联客户机配置中的 CPU
和内存使用量。随后，对每个此类备选方案，都会考虑所有节点的 CPU 和内存使用量； 其中内存权重高
CPU 和内存，都会考虑节点中的最高使用量（权重更高，因为理想情况下不应有节点被过度承诺）以及
（以便在已经存在承诺更高的节点时仍能区分）。

Important
服务越多，可能的组合就越多，因此如果有数千个 HA 管理服务，
当前不建议使用该模式。

15.12.3 CRS 调度点

CRS 算法不会在每一轮都应用于每个服务，因为这会导致大量持续迁移。 根据工作负载不同，这可能给
因此，Proxmox VE HA 管理器倾向于让服务保持在当前节点上。

CRS 当前用于以下调度点：

- 服务恢复（始终活动）。当带有活动 HA 服务的节点失败时，其所有服务都需要恢复到其他节点。这里会使用 CRS 算法，在剩余节点之间均衡该恢复过程。
- HA 组配置变更（始终活动）。如果某个节点从组中移除，或其优先级降低，HA 栈会使用 CRS 算法为该组中的 HA 服务寻找新的目标节点，以匹配调整后的优先级约束。
- HA 服务从 stopped 到 start 的转换（可选启用）。请求启动已停止服务时，是根据 CRS 算法检查最尤其是其磁盘卷位于共享存储上时。可以通过在数据中心配置中设置 **ha-rebalance-on-start** CRS 选项来启用此行为。也可以在 Web UI 的 `Datacenter` → `Options` → `Cluster Resource Scheduling` 下更改该选项。

Chapter 16

备份与还原

备份是任何合理 IT 部署的必要组成部分，Proxmox VE 提供了一个完全集成的解决方案， 可利用各类存
mode 选项， 在备份一致性和客户机系统停机时间之间进行精细权衡。

Proxmox VE 备份始终是完整备份，包含 VM/CT 配置和所有数据。备份可以通过 GUI 启动，也可以通
vzdump 命令行工具启动。

备份存储

运行备份之前，必须先定义备份存储。有关如何添加存储，请参见 storage documentation。备份存
Proxmox Backup Server 存储，此时备份会以去重数据块和元数据的形式保存；也可以是文件级存储
此时备份会作为普通文件保存。由于 Proxmox Backup Server 具备高级功能， 建议在专用主机上使用
NFS 服务器也是一个不错的替代方案。在这两种 情况下，之后都可能需要将这些备份保存到磁带驱动器

计划备份

可以为备份作业设置计划，使其在指定日期和时间自动执行，并可选择节点和 客户机系统。更多信息请
备份作业 一节。

16.1 备份模式

根据客户机类型，可以通过多种方式提供一致性（选项 mode）。

虚拟机备份模式：

stop mode
: 该模式提供最高的备份一致性，代价是虚拟机运行会有短暂停机。其工作方式是先 对虚拟机执行
QEMU 进程备份虚拟机数据。备份开始后， 如果虚拟机之前处于运行状态，它会恢复到完整运行
保证一致性。

suspend mode
: 该模式出于兼容性原因提供，会在调用 snapshot 模式之前挂起虚拟机。由于挂起 虚拟机会导致
snapshot 模式。

snapshot mode

该模式提供最低的运行停机时间，代价是存在较小的一致性风险。其工作方式是执行 Proxmox VE 实时备份，即在虚拟机运行期间复制数据块。如果启用并运行了 guest agent（`agent: 1`），它会调用 `guest-fsfreeze-freeze` 和 `guest-fsfreeze-thaw` 以提高一致性。

> **Note**
> 对于 Windows 客户机，如果客户机内部还使用其他备份软件，则必须配置 guest agent。更多详情请参见 guest agent 章节中的 Freeze & Thaw。

可在线查看 QemuServer 的 Proxmox VE 实时备份技术概览： here。

> **Note**
> Proxmox VE 实时备份可在任何存储类型上提供类似快照的语义。它不要求底层存储支持快照。另请注意，由于备份通过后台 QEMU 进程完成，当 QEMU 读取虚拟机磁盘时，已停止的虚拟机会在短时间内显示为正在运行。但虚拟机本身并未启动，
> 只是其磁盘被读取。

容器备份模式：

stop mode

在备份期间停止容器。这可能导致很长的停机时间。

suspend mode

该模式使用 rsync 将容器数据复制到临时位置（见选项 `--tmpdir`）。随后挂起容器，并由第二次 rsync 复制已更改文件。之后再次启动（恢复）容器。这样停机时间最短，但需要额外空间保存容

当容器位于本地文件系统上，而备份目标存储是 NFS/CIFS 服务器时，也应将 `--tmpdir` 设置在本 NFS 服务器，并且要在 suspend 模式下备份使用 ACL 的本地容器，也必须使用本地 `tmpdir`。

snapshot mode

该模式使用底层存储的快照设施。首先会挂起容器以确保数据一致性。随后为容器卷 创建临时快照 tar 文件中。最后再次删除临时快照。

> **Note**
> `snapshot` 模式要求所有被备份卷都位于支持快照的存储上。可以使用 `backup=no` 挂载点选项将单个卷排除在备份之外（因此也排除该要求）。

> **Note**
> 默认情况下，除 Root Disk 挂载点外的其他挂载点不会包含在备份中。对于卷挂载点，可以设置 **Backup** 选项将该挂载点纳入备份。设备挂载和绑定挂载永远不会被备份，因为其内容在 Proxmox VE 存储库之外管理。

16.1.1 虚拟机备份 Fleecing

启动虚拟机备份时，QEMU 会在其块层安装一个 "copy-before-write" 过滤器。该过滤器 确保在客户机
阻塞，直到该操作完成，因此对尚未备份扇区的客户机 IO 会受备份目标速度限制。

使用 backup fleecing 时，这类旧数据会缓存在 fleecing 镜像中，而不是直接发送到 备份目标。这有助
IO 性能，在某些场景下甚至可以避免卡住，代价是 需要更多存储空间。

要手动启动虚拟机 123 的备份，并在存储 `local-lvm` 上创建 fleecing 镜像， 请运行：

```
vzdump 123 --fleecing enabled=1,storage=local-lvm
```

和往常一样，可以为特定备份作业设置该选项，也可以通过 配置选项 设置为节点范围的默认选项。
在 UI 中，编辑备份作业时可在 *Advanced* 选项卡中配置 fleecing。

fleecing 存储应是快速的本地存储，并支持精简置备和 discard。例如 LVM-thin、RBD、 存储配置中
`sparse 1` 的 ZFS，以及许多基于文件的存储。理想情况下， fleecing 存储应为专用存储，这样即使它
备份失败。fleecing 镜像中已经备份的部分会被 discard，以尽量降低空间使用量。

对于不支持 discard 的基于文件的存储（例如 4.2 之前版本的 NFS），应在存储配置中 设置 `prealloc
off`。配合 qcow2 使用时（当存储支持时会自动作为 fleecing 镜像格式使用），其优势是镜像中已分配
节省不少空间。

Warning
在非精简置备的存储上，例如未使用 `sparse` 选项的 LVM 或 ZFS，需要预先为 fleecing 镜像保留与原始磁盘相同的完整大小。在精简置备存储上，只有当备份正忙于另一块磁盘且客户机重写整块磁盘时，fleecing 镜像才可能增长到与原始镜像相同的大小。

16.1.2 CT 变更检测模式

设置变更检测模式会定义 pxar 归档的编码格式，以及在以 Proxmox Backup Server 作为目标的容器备

编辑作业时，可以在 *Advanced* 选项卡中为单个备份作业配置变更检测模式选项。 此外，也可以通过
配置选项 将该选项设置为 节点范围的默认选项。

有 3 种可用的变更检测模式：

模式	描述
`Default`	读取并编码所有文件到单个归档中，使用 pxar 格式版本 1。
`Data`	读取并编码所有文件，但将数据和元数据拆分到独立流中，使用 pxar 格式版本 2。
`Metadata`	与 `Data` 一样拆分流并使用归档格式版本 2，但会使用上一个快照的元数据归档（如果存在）来检测未变更文件，并在可能时复用其数据块，而不从磁盘读取文

要使用变更检测模式 `metadata` 执行备份，可以运行：

```
vzdump 123 --storage pbs-storage --pbs-change-detection-mode ↩
  metadata
```

Note
虚拟机备份，或目标为 Proxmox Backup Server 以外存储后端的备份， 不受该设置影响。

16.2 备份文件名

较新版本的 vzdump 会将客户机类型和备份时间编码到文件名中，例如：

```
vzdump-lxc-105-2009_10_09-11_04_43.tar
```

这样就可以在同一目录中存储多个备份。可以通过各种保留选项限制保留的备份数量，请参见下面的 Backup Retention 一节。

16.3 备份文件压缩

备份文件可以使用以下算法之一压缩：`lzo` [1]、`gzip` [2] 或 `zstd` [3]。

目前，Zstandard（zstd）是这三种算法中最快的。相较于 lzo 和 gzip，zstd 的另一个优势是支持多线
和 gzip 使用更广泛，也通常默认安装。

可以安装 pigz [4] 作为 gzip 的直接替代品，以通过多线程提供更好性能。对于 pigz 和 zstd，可以调整线
配置选项。

通常可以通过备份文件名的扩展名判断创建该备份时使用的压缩算法。

.zst	Zstandard (zstd) 压缩
.gz or .tgz	gzip 压缩
.lzo	lzo 压缩

如果备份文件名不是以上述文件扩展名之一结尾，则它不是由 vzdump 压缩的。

16.4 备份加密

对于 Proxmox Backup Server 存储，可以选择设置备份的客户端加密，请参见 相应章节。

16.5 备份作业

除了手动触发备份，也可以设置周期性作业，将所有或选定的虚拟客户机备份到存储。可以在 UI
的 *Datacenter → Backup* 下管理作业，也可以通过 `/cluster/backup` API 端点管理。两种方式都会
`/etc/pve/jobs.cfg` 中生成 作业条目，并由 `pvescheduler` 守护进程解析和执行。

作业可以配置为面向所有集群节点或某个特定节点，并按照给定计划执行。计划格式 与 `systemd`
calendar events 非常相似，详情请参见 calendar events 一节。UI 中的 *Schedule* 字段 可以自由编

可以配置特定于作业的 保留选项，覆盖存储或节点 配置中的设置；也可以配置 备注模板，用于保存随备
一起记录的附加信息。

[1] Lempel–Ziv–Oberhumer a lossless data compression algorithm LZO
[2] gzip - based on the DEFLATE algorithm gzip
[3] Zstandard a lossless data compression algorithm Zstandard
[4] pigz - parallel implementation of gzip https://zlib.net/pigz/

如果主机在计划时间离线，或 pvescheduler 在计划时间被禁用，计划备份就会错过 执行。因此可以配
Repeat missed 选项（UI 中的 *Advanced* 选项卡，配置中为 repeat-missed）后，可告知调度器尽

有一些设置可用于调优备份性能（其中部分在 UI 的 *Advanced* 选项卡中公开）。 最重要的是用于限制
IO 带宽的 bwlimit。压缩器使用的线程数量可分别通过 pigz（替代 gzip）和 zstd 设置控制。此外还
ionice（使用 BFQ 调度器时），以及作为 performance 设置一部分的 max-workers（仅影响虚拟
pbs-entries-max（仅影响容器备份）。详情请参见 配置选项。

16.6 备份保留

通过 prune-backups 选项，可以灵活指定要保留哪些备份。

可用的保留选项如下：

keep-all <boolean>
: 保留所有备份。如果该值为 true，则不能设置其他选项。

keep-last <N>
: 保留最后 <N> 个备份。

keep-hourly <N>
: 保留最近 <N> 小时的备份。如果单个小时内有多个备份，则只保留最新的一个。

keep-daily <N>
: 保留最近 <N> 天的备份。如果单日内有多个备份，则只保留最新的一个。

keep-weekly <N>
: 保留最近 <N> 周的备份。如果单周内有多个备份，则只保留最新的一个。

Note
一周从星期一开始，到星期日结束。软件使用 ISO week date 系统，并能 正确处理年末周。

---

keep-monthly <N>
: 保留最近 <N> 个月的备份。如果单月内有多个备份，则只保留最新的一个。

keep-yearly <N>
: 保留最近 <N> 年的备份。如果单年内有多个备份，则只保留最新的一个。

保留选项会按上述顺序处理。每个选项只覆盖其时间范围内的备份。下一个选项不会 处理已经覆盖的备

将要使用的保留选项指定为逗号分隔列表，例如：

```
# vzdump 777 --prune-backups keep-last=3,keep-daily=13,keep-yearly=9
```

虽然可以将 prune-backups 直接传递给 vzdump，但通常更合理的做法是在存储层级 配置该设置，这
Web 界面完成。

Note
旧的 maxfiles 选项已弃用，应替换为 keep-last；如果 maxfiles 为 0 表示无限保留，则应替换为 keep-all。

16.6.1 Prune 模拟器

可以使用 Proxmox Backup Server 文档中的 prune simulator，探索不同备份计划下各种保留选项的

16.6.2 保留设置示例

备份频率和旧备份保留时间可能取决于特定工作负载中数据变更频率，以及较旧状态的 重要性。当备份
法律要求。

在本示例中，假设每天执行备份，保留周期为 10 年，并且已保存备份之间的时间间隔 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重大升级前后额外创建 一个备份。设置 keep-last 可以确保这一点。

`keep-hourly` 未设置 - 对于每日备份而言这并不相关。通过 keep-last 已经覆盖了 额外的手动备份。

`keep-daily=13` - 与至少覆盖一天的 keep-last 一起使用，可确保至少拥有两周备份。

`keep-weekly=8` - 确保至少拥有完整两个月的每周备份。

`keep-monthly=11` - 与前面的 keep 设置一起使用，可确保至少拥有一年的每月备份。

`keep-yearly=9` - 这是用于长期归档的设置。由于前面的选项已经覆盖当前年份， 因此可将其设置为 9 来覆盖剩余年份，从而总计至少覆盖 10 年。

建议使用高于环境最低要求的保留周期；如果发现保留周期不必要地过高，随时可以 降低，但备份一旦

16.7 备份保护

可以将备份标记为 `protected` 以防止其被移除。尝试通过 Proxmox VE 的 UI、CLI 或 API 移除受保护
Proxmox VE 强制执行的，而不是文件系统本身； 这意味着任何对底层备份存储具有写访问权限的人，

> **Note**
> 受保护备份会被 pruning 忽略，并且不计入保留设置。

对于基于文件系统的存储，保护通过 sentinel 文件 `<backup-name>.protected` 实现。 对于 Proxmox Backup Server，该保护在服务器端处理（自 Proxmox Backup Server 版本 2.1 起可用）。

使用存储选项 `max-protected-backups` 控制每个客户机在该存储上允许多少个受保护 备份。使用 `-1` 表示无限制。默认情况下，对具有 `Datastore.Allocate` 权限的用户 无限制，对其他用户为 5。

16.8 备份备注

可以使用 UI 中的 *Edit Notes* 按钮，或通过存储内容 API，为备份添加备注。

也可以为备份作业和手动备份指定模板，以动态生成备注。模板字符串可以包含由 双大括号包围的变量

当前支持：

- {{cluster}} 集群名称（如果有）
- {{guestname}} 虚拟客户机分配的名称
- {{node}} 创建备份的节点主机名
- {{vmid}} 客户机的数字 VMID

通过 API 或 CLI 指定时，它必须是单行，其中换行符和反斜杠需要分别转义为字面量 \n 和 \\。

16.9 还原

备份归档可以通过 Proxmox VE Web GUI 还原，也可以通过以下 CLI 工具还原：

pct restore
: 容器还原工具

qmrestore
: 虚拟机还原工具

详情请参见相应手册页。

16.9.1 带宽限制

还原一个或多个大型备份可能需要大量资源，尤其是从备份存储读取以及写入目标存储 所需的存储带宽
为避免这种情况，可以为备份作业设置带宽限制。Proxmox VE 为还原和归档实现了两种限制：

- 每个还原任务限制：表示从备份归档读取时的最大带宽
- 每个存储写入限制：表示写入特定存储时使用的最大带宽

读取限制会间接影响写入限制，因为写入量不能超过读取量。较小的每作业限制会覆盖 较大的每存储限
'Data.Allocate' 权限时， 较大的每作业限制才会覆盖每存储限制。

可以使用还原 CLI 命令中的 '--bwlimit <integer>` 选项，为特定还原作业设置带宽 限制。限制单位为
KiB/s，这意味着传入`10240' 会将备份读取速度限制为 10 MiB/s， 从而确保其余可用存储带宽留给已

> **Note**
> 可以为 'bwlimit` 参数使用 0，以禁用某个特定还原作业的所有限制。
> 如果需要尽快还原非常重要的虚拟客户机，这会很有帮助。（需要在存储上拥有
> `Data.Allocate' 权限）

大多数情况下，存储通常可用的带宽会随时间保持稳定，因此系统实现了按已配置存储 设置默认带宽限

```
# pvesm set STORAGEID --bwlimit restore=KIBs
```

16.9.2 实时还原（Live-Restore）

还原大型备份可能需要很长时间，在此期间客户机仍不可用。对于存储在 Proxmox Backup Server 上的虚拟机备份，可以使用 live-restore 选项缓解该等待时间。

通过 GUI 中的复选框或 qmrestore 的 --live-restore 参数启用 live-restore 后， 虚拟机会在还原数据块。

注意，这有两个限制：

- 在 live-restore 期间，虚拟机的磁盘读取速度会受限，因为数据必须从备份服务器 加载（但一旦加载只会在第一次产生代价）。写入速度基本不受影响。
- 如果 live-restore 因任何原因失败，虚拟机会处于未定义状态，即并非所有数据都 可能已经从备份中很可能 无法保留失败还原操作期间写入的任何数据。

这种运行模式尤其适用于大型虚拟机，且初始运行只需要少量数据的场景，例如 Web 服务器。一旦 OS 和必要服务启动，虚拟机即可运行，同时后台任务继续复制较少使用的 数据。

16.9.3 单文件还原

存储 GUI 的 *Backups* 选项卡中的 *File Restore* 按钮可用于直接打开备份中所含 数据的文件浏览器。此 Proxmox Backup Server 上的备份。

对于容器，文件树第一层会显示所有包含的 *pxar* 归档，可以自由打开和浏览。 对于虚拟机，第一层会 存储技术列表。在最基本的情况下，会有一个名为 *part* 的条目，表示分区表， 其中包含驱动器上找到的（不支持的客户机文件系统、存储技术等）。

可以使用 *Download* 按钮下载文件和目录；下载目录时会即时压缩为 zip 归档。

为了安全访问可能包含不可信数据的虚拟机镜像，系统会启动一个临时虚拟机（不会作为 客户机可见）hypervisor 系统暴露于危险之中。该虚拟机会在超时后自行停止。从用户角度看， 整个过程是透明的。

Note
为便于排错，每个临时虚拟机实例都会在 /var/log/proxmox-backup/file-restore/ 中生成日志文件。当尝试还原单个文件或 访问备份归档中包含的文件系统失败时，该日志文件可能包

16.10 配置

全局配置存储在 /etc/vzdump.conf。该文件使用简单的冒号分隔键/值格式。 每一行格式如下：

```
OPTION: value
```

文件中的空行会被忽略，以 # 字符开头的行会被视为注释并同样被忽略。该文件中的 值会作为默认值使

当前支持以下选项：

bwlimit: <integer> (0 - N) (*default* = 0)
: 限制 I/O 带宽（单位 KiB/s）。

---

compress: <0 | 1 | gzip | lzo | zstd> (*default* = 0)
压缩转储文件。

dumpdir: <string>
将生成的文件保存到指定目录。

exclude-path: <array>
排除指定文件/目录（shell glob）。以 / 开头的路径锚定到容器根目录，其他路径则相对于各个子

fleecing: [[enabled=]<1|0>] [,storage=<storage ID>]
backup fleecing 选项（仅适用于虚拟机）。

enabled=<boolean> (*default* = 0)
启用 backup fleecing。当客户机发生新写入时，将相关块的备份数据缓存在指定存储上，而
I/O 性能，甚至避免卡顿，但代价是需要更多存储空间。

storage=<storage ID>
使用该存储保存 fleecing 镜像。为了高效利用空间，最好使用支持 discard 且支持 thin provisioning 或 sparse file 的本地存储。

ionice: <integer> (0 - 8) (*default* = 7)
使用 BFQ 调度器时设置 I/O 优先级。对于虚拟机的 snapshot 和 suspend 模式备份，该设置只影
8 表示使用 idle 优先级，否则使用带指定数值的 best-effort 优先级。

lockwait: <integer> (0 - N) (*default* = 180)
等待全局锁的最长时间（分钟）。

mailnotification: <always | failure> (*default* = always)
已弃用：请改用通知目标/匹配器。指定何时发送通知邮件。

mailto: <string>
已弃用：请改用通知目标/匹配器。应接收电子邮件通知的电子邮件地址或用户，以逗号分隔。

maxfiles: <integer> (1 - N)
已弃用：请改用 prune-backups。每个客户系统的最大备份文件数。

mode: <snapshot | stop | suspend> (*default* = snapshot)
备份模式。

notes-template: <string>
用于生成备份备注的模板字符串。可以包含变量，这些变量会被对应值替换。当前支持 {\{\cluster}}、{\{\guestname}}、{\{\node}} 和 {\{\vmid}}，未来可能会增加更多变量。必须为单
\n 和 \\。

> **Note**
> 需要选项：storage

notification-mode: <auto | legacy-sendmail | notification-system> (*default* = auto)
确定使用哪个通知系统。如果设置为 legacy-sendmail，vzdump 会使用 mailto/mailnotif
参数，并通过 sendmail 命令向指定地址发送电子邮件。如果设置为 notification-system，

PVE 通知系统发送通知，并忽略 mailto 和 mailnotification。如果设置为 auto（默认设置）mailto 时会发送电子邮件，否则使用通知系统。

notification-policy: <always | failure | never> (*default* = always)
已弃用：不要使用。

notification-target: <string>
已弃用：不要使用。

pbs-change-detection-mode: <data | legacy | metadata>
用于检测文件变化并为容器备份切换编码格式的 PBS 模式。

performance: [max-workers=<integer>] [,pbs-entries-max=<integer>]
其他性能相关设置。

max-workers=<integer> (1 - 256) (*default* = 16)
适用于虚拟机。允许同时使用的最大 I/O worker 数量。

pbs-entries-max=<integer> (1 - N) (*default* = 1048576)
适用于发送到 PBS 的容器备份。限制同一时间允许驻留在内存中的条目数量，以避免意外 OOM。对于包含大量文件的容器，可增大该值以支持备份。

pigz: <integer> (*default* = 0)
当 N>0 时使用 pigz 代替 gzip。N=1 表示使用一半核心数，N>1 表示使用 N 作为线程数。

pool: <string>
备份指定 pool 中包含的所有已知客户系统。

protected: <boolean>
如果为 true，则将备份标记为受保护。

Note
需要选项：storage

---

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。

keep-all=<boolean>
保留所有备份。为 true 时与其他选项冲突。

keep-daily=<N>
保留最近 <N> 个不同日期的备份。如果同一天有多个备份，只保留最新的一个。

keep-hourly=<N>
保留最近 <N> 个不同小时的备份。如果同一小时有多个备份，只保留最新的一个。

keep-last=<N>
保留最近 <N> 个备份。

keep-monthly=<N>
保留最近 <N> 个不同月份的备份。如果同一月份有多个备份，只保留最新的一个。

keep-weekly=<N>
保留最近 <N> 个不同周的备份。如果同一周有多个备份，只保留最新的一个。

keep-yearly=<N>
保留最近 <N> 个不同年份的备份。如果同一年有多个备份，只保留最新的一个。

remove: <boolean> (*default* = 1)
根据 prune-backups 清理较旧备份。

script: <string>
使用指定的 hook script。

stdexcludes: <boolean> (*default* = 1)
排除临时文件和日志。

stopwait: <integer> (0 - N) (*default* = 10)
等待客户系统停止的最长时间（分钟）。

storage: <storage ID>
将生成的文件保存到该存储。

tmpdir: <string>
将临时文件保存到指定目录。

zstd: <integer> (*default* = 1)
Zstd 线程数。N=0 表示使用一半可用核心数；如果 N 设置为大于 0 的值，则将 N 作为线程数。

示例 **vzdump.conf** 配置

```
tmpdir: /mnt/fast_local_disk
storage: my_backup_storage
mode: snapshot
bwlimit: 10000
```

16.11 钩子脚本

可以通过选项 --script 指定钩子脚本。该脚本会在备份过程的各个阶段被调用，并相应设置参数。可

16.12 文件排除

Note
该选项仅适用于容器备份。

vzdump 默认跳过以下文件（可通过选项 --stdexcludes 0 禁用）

```
/tmp/?*
/var/tmp/?*
/var/run/?*pid
```

也可以手动指定（额外的）排除路径，例如：

```
# vzdump 777 --exclude-path /tmp/ --exclude-path '/var/foo*'
```

这会排除目录 /tmp/，以及名为 /var/foo、/var/foobar 等的任何文件或目录。

Warning
对于备份到 Proxmox Backup Server（PBS）的备份以及 suspend 模式备份，带尾随斜杠的模式会匹配目录，但不会匹配文件。另一方面，对于非 PBS 的 snapshot 模式和 stop 模式备份，带尾随斜杠的模式目前完全不会匹配，因为 tar 命令不支持这种方式。

不以 / 开头的路径不会锚定到容器根目录，而是会相对于任意子目录进行匹配。 例如：

```
# vzdump 777 --exclude-path bar
```

这会排除名为 /bar、/var/bar、/var/foo/bar 等的任何文件或目录，但不会排除 /bar2。

配置文件也会存储在备份归档内部（位于 ./etc/vzdump/），并会被正确还原。

16.13 示例

简单转储客户机 777，不使用快照，只将客户机私有区域和配置文件归档到默认转储目录（通常为 /var/lib/vz/dump/）。

```
# vzdump 777
```

使用 rsync 和挂起/恢复创建快照（最短停机时间）。

```
# vzdump 777 --mode suspend
```

备份所有客户机系统，并向 root 和 admin 发送通知邮件。由于设置了 mailto，且 notification-默认为 auto，通知邮件会通过系统的 sendmail 命令发送， 而不是通过通知系统发送。

```
# vzdump --all --mode suspend --mailto root --mailto admin
```

使用 snapshot 模式（无停机）和非默认转储目录。

```
# vzdump 777 --dumpdir /mnt/backup --mode snapshot
```

备份多个客户机（选择性备份）

```
# vzdump 101 102 103 --mailto root
```

备份除 101 和 102 以外的所有客户机

```
# vzdump --mode suspend --exclude 101,102
```

将容器还原为新的 CT 600

```
# pct restore 600 /mnt/backup/vzdump-lxc-777.tar
```

将 QemuServer 虚拟机还原为 VM 601

```
# qmrestore /mnt/backup/vzdump-qemu-888.vma 601
```

使用管道将现有容器 101 克隆为新的容器 300，并设置 4GB 根文件系统

```
# vzdump 101 --stdout | pct restore --rootfs 4 300 -
```

常用命令示例：

```
# vzdump 777 --mode snapshot
# vzdump --all --mode suspend
# qmrestore /mnt/backup/vzdump-qemu-888.vma 601
```

Chapter 17

通知

17.1 概述

- 当发生存储复制失败、节点 fencing、备份完成/失败以及其他事件时，Proxmox VE 会发出 通知事件
- 通知匹配器 会将通知事件路由到一个或多个通知目标。匹配器可以配置匹配规则，根据通知事件的元
- 通知目标 是匹配器将通知事件路由到的目的地。目标有多种类型，包括基于邮件的 Sendmail 和 SMTP，以及 Gotify。

备份任务可以配置 通知模式。该模式允许在通知系统和旧版通知邮件发送方式之间选择。旧版模式等同 Proxmox VE 8.1 之前处理通知的方式。

通知系统可以在 GUI 的 Datacenter → Notifications 下配置。配置存储在 `/etc/pve/notificatio` 和 `/etc/pve/priv/notifications.cfg` 中，后者包含通知目标的密码或认证 token 等敏感配置选 `root` 读取。

17.2 通知目标

Proxmox VE 提供多种类型的通知目标。

17.2.1 Sendmail

`sendmail` 二进制程序常见于类 Unix 操作系统，用于处理电子邮件消息的发送。它是一个命令行工具

Sendmail 通知目标使用 `sendmail` 二进制程序向已配置的用户或电子邮件地址列表发送邮件。如果选 `root@pam` 用户，这就是安装期间输入的电子邮件地址。用户的电子邮件地址可以在 `Datacenter → Permissions → Users` 中配置。如果用户没有关联电子邮件地址，则不会发送邮件。

Note
在标准 Proxmox VE 安装中，`sendmail` 二进制程序由 Postfix 提供。可能需要配置 Postfix 才能正确投递邮件，例如设置外部邮件中继（smart host）。如果投递失败，请检查系统日志中 Postfix 守护进程记录的消息。

Sendmail 目标插件配置包含以下选项：

- mailto: 通知应发送到的电子邮件地址。可设置多次以支持多个收件人。
- mailto-user: 应接收邮件的用户。用户的电子邮件地址会从 users.cfg 中查找。可设置多次以支
- author: 设置电子邮件作者。默认为 Proxmox VE。
- from-address: 设置电子邮件发件地址。如果未设置该参数，插件会回退到 datacenter.cfg 中的 email_from 设置。如果该设置也未设置，插件默认使用 root@$hostname，其中 $hostnam 是节点主机名。电子邮件中的 From 头会设置为 $author <$from-address>。
- comment: 此目标的注释。

配置示例（/etc/pve/notifications.cfg）：

```
sendmail: example
        mailto-user root@pam
        mailto-user admin@pve
        mailto max@example.com
        from-address pve1@example.com
        comment Send to multiple users/addresses
```

17.2.2 SMTP

SMTP 通知目标可以直接向 SMTP 邮件中继发送电子邮件。该目标不使用系统的 MTA 投递邮件。与 sendmail 目标类似，如果选择用户作为收件人，则使用该用户配置的电子邮件地址。

Note
与 sendmail 目标不同，SMTP 目标在邮件投递失败时没有任何排队/重试机制。

SMTP 目标插件配置包含以下选项：

- mailto: 通知应发送到的电子邮件地址。可设置多次以支持多个收件人。
- mailto-user: 应接收邮件的用户。用户的电子邮件地址会从 users.cfg 中查找。可设置多次以支
- author: 设置电子邮件作者。默认为 Proxmox VE。
- from-address: 设置电子邮件的 From 地址。SMTP 中继可能要求该地址属于对应用户，以避免欺 From 头会设置为 $author <$from-address>。
- username: 认证时使用的用户名。如果未设置用户名，则不执行认证。支持 PLAIN 和 LOGIN 认证方法。
- password: 认证时使用的密码。
- mode: 设置加密模式（insecure、starttls 或 tls）。默认为 tls。
- server: SMTP 中继的地址/IP。

- port: 要连接的端口。如果未设置，根据 mode 的取值，默认分别使用 25（insecure）、465（t
  587（starttls）。
- comment: 此目标的注释。

配置示例（/etc/pve/notifications.cfg）：

```
smtp: example
        mailto-user root@pam
        mailto-user admin@pve
        mailto max@example.com
        from-address pve1@example.com
        username pve1
        server mail.example.com
        mode starttls
```

/etc/pve/priv/notifications.cfg 中包含密钥 token 的对应条目：

```
smtp: example
        password somepassword
```

17.2.3 Gotify

Gotify 是一个开源的自托管通知服务器，允许向各种设备和应用程序发送并接收推送通知。它提供简单
API 和 Web 界面，便于与不同平台和服务集成。

Gotify 目标插件配置包含以下选项：

- server: Gotify 服务器的基础 URL，例如 http://<ip>:8888
- token: 认证 token。可以在 Gotify Web 界面中生成 token。
- comment: 此目标的注释。

> **Note**
> Gotify 目标插件会遵循 数据中心配置 中的 HTTP 代理设置。

配置示例（/etc/pve/notifications.cfg）：

```
gotify: example
        server http://gotify.example.com:8888
        comment Send to multiple users/addresses
```

/etc/pve/priv/notifications.cfg 中包含密钥 token 的对应条目：

```
gotify: example
        token somesecrettoken
```

17.2.4 Webhook

Webhook 通知目标会向可配置的 URL 执行 HTTP 请求。

可用配置选项如下：

- `url`: 执行 HTTP 请求的 URL。支持通过模板注入消息内容、元数据和 secret。
- `method`: 要使用的 HTTP Method（POST/PUT/GET）。
- `header`: 请求应设置的 HTTP 头数组。支持通过模板注入消息内容、元数据和 secret。
- `body`: 应发送的 HTTP body。支持通过模板注入消息内容、元数据和 secret。
- `secret`: secret 键值对数组。它们会存储在仅 root 可读的受保护配置文件中。可以在 body/-header/URL 模板中通过 `secrets` 命名空间访问 secret。
- `comment`: 此目标的注释。

对于支持模板的配置选项，可以使用 Handlebars 语法访问以下属性：

- `{{ title }}`: 渲染后的通知标题。
- `{{ message }}`: 渲染后的通知正文。
- `{{ severity }}`: 通知严重级别（`info`、`notice`、`warning`、`error`、`unknown`）。
- `{{ timestamp }}`: 通知时间戳，以 UNIX epoch 表示（秒）。
- `{{ fields.<name> }}`: 通知任意元数据字段的子命名空间。例如，`fields.type` 包含通知类型 通知事件。
- `{{ secrets.<name> }}`: secret 的子命名空间。例如，名为 `token` 的 secret 可通过 `secrets.` 访问。

为方便使用，提供以下 helper：

- `{{ url-encode <value/property> }}`: 对属性/字面量进行 URL 编码。
- `{{ escape <value/property> }}`: 转义无法安全表示为 JSON 字符串的控制字符。
- `{{ json <value/property> }}`: 将值渲染为 JSON。将整个子命名空间（例如 `fields`）作为 JSON payload 的一部分传递时很有用，例如 `{{ json fields }}`。

示例

- Method: `POST`
- URL: `https://ntfy.sh/{{ secrets.channel }}`
- Headers:
  - `Markdown: Yes`

- Body:

```
```
{{ message }}
```
```

- Secrets:
  - channel: <your ntfy.sh channel>
- Method: POST
- URL: https://discord.com/api/webhooks/{{ secrets.token }}
- Headers:
  - Content-Type: application/json
- Body:

```
{
  "content": "``` {{ escape message }}```"
}
```

- Secrets:
  - token: <token>
- Method: POST
- URL: https://hooks.slack.com/services/{{ secrets.token }}
- Headers:
  - Content-Type: application/json
- Body:

```
{
  "text": "``` {{escape message}}```",
  "type": "mrkdwn"
}
```

- Secrets:
  - token: <token>

17.3 通知匹配器

通知匹配器根据匹配规则将通知路由到通知目标。这些规则可以匹配通知的特定属性，例如时间戳（ma
如果通知被某个匹配器匹配，该匹配器配置的所有目标都会收到通知。

可以创建任意数量的匹配器，每个匹配器都可以有自己的匹配规则和要通知的目标。即使某个目标被多

没有任何匹配规则的匹配器始终为 true；其配置的目标始终会被通知。

```
matcher: always-matches
        target admin
        comment b''该b''b''匹b''b''配b''b''器b''b''始b''b''终b''b''匹b''b' ↩
           '配b''
```

17.3.1 匹配器选项

- target: 确定匹配器命中时应通知哪个目标。可多次使用以通知多个目标。
- invert-match: 反转整个匹配器的结果。
- mode: 确定如何评估各个匹配规则以计算整个匹配器的结果。设置为 all 时，所有匹配规则都必须匹
  any 时，至少一个规则匹配即可。默认为 all。
- match-calendar: 将通知时间戳与日程匹配。
- match-field: 匹配通知的元数据字段。
- match-severity: 匹配通知严重级别。
- comment: 此匹配器的注释。

17.3.2 日历匹配规则

日历匹配器会将通知发送时间与可配置日程进行匹配。

- match-calendar 8-12
- match-calendar 8:00-15:30
- match-calendar mon-fri 9:00-17:00
- match-calendar sun,tue-wed,fri 9-17

17.3.3 字段匹配规则

通知包含若干可匹配的元数据字段。使用 exact 作为匹配模式时，可以使用，作为分隔符。只要元数据

- match-field exact:type=vzdump 仅匹配与备份相关的通知。
- match-field exact:type=replication,fencing 匹配 replication 和 fencing 通知。

- `match-field regex:hostname=^.+\.example\.com$` 匹配节点主机名。

如果被匹配的元数据字段不存在，则通知不会匹配。例如，`match-field regex:hostname=.*` 指令只会匹配具有任意 `hostname` 元数据字段的通知；如果该字段不存在，则不会匹配。

17.3.4 严重级别匹配规则

通知具有关联的严重级别，可用于匹配。

- `match-severity error`: 仅匹配错误。
- `match-severity warning,error`: 匹配警告和错误。

当前使用以下严重级别： `info`, `notice`, `warning`, `error`, `unknown`.

17.3.5 示例

```
matcher: workday
        match-calendar mon-fri 9-17
        target admin
        comment b''’工b’'b''’作b’'b''’时b’'b''’间b’'b''’通b’'b''’知b’'b''’管b’'b' ↩
           ’理b’'b''’员b’'
```

```
matcher: night-and-weekend
        match-calendar mon-fri 9-17
        invert-match true
        target on-call-admins
        comment b''’非b’'b''’工b’'b''’作b’'b''’时b’'b''’间b’'b''’使b’'b''’用b’'b' ↩
           ’独b’'b''’立b’'b''’目b’'b''’标b’'
```

```
matcher: backup-failures
        match-field exact:type=vzdump
        match-severity error
        target backup-admins
        comment b''’将b’'b''’备b’'b''’份b’'b''’失b’'b''’败b’'b''’通b’'b''’知b’'b' ↩
           ’发b’'b''’送b’'b''’给b’'b''’一b’'b''’组b’'b''’管b’'b''’理b’'b''’员b’'

matcher: cluster-failures
        match-field exact:type=replication,fencing
        target cluster-admins
        comment b''’将b’'b''’集b’'b''’群b’'b''’相b’'b''’关b’'b''’通b’'b''’知b’'b' ↩
           ’发b’'b''’送b’'b''’给b’'b''’另b’'b''’一b’'b''’组b’'b''’管b’'b''’理b’'b' ↩
           ’员b’'
```

17.4 通知事件

事件	**type**	严重级别	元数据字段（除 **type** 外）
系统更新可用	package-updates	info	hostname
集群节点被 fencing	fencing	error	hostname
存储复制任务失败	replication	error	hostname, job-id
备份成功	vzdump	info	hostname, job-id（仅备份任务）
备份失败	vzdump	error	hostname, job-id（仅备份任务）
root 邮件	system-mail	unknown	hostname

字段名	说明
type	通知类型
hostname	不含域名的主机名（例如 pve1）
job-id	任务 ID

Note
只有根据调度自动执行的备份任务通知才会设置 job-id；如果通过 UI 中的 Run now 按钮手动触发，则不会设置该字段。

17.5 系统邮件转发

某些本地系统守护进程（例如 smartd）会生成通知邮件，这些邮件最初发送给本地 root 用户。Proxmox VE 会将这些邮件送入通知系统，并作为类型为 system-mail、严重级别为 unknown 的通知处理。

当电子邮件转发到 sendmail 目标时，邮件内容和头部会按原样转发。对于所有其他目标，系统会尝试提取 HTML 内容，则会在此过程中转换为纯文本格式。

17.6 权限

要修改/查看通知目标配置，需要在 /mapping/notifications ACL 节点上具备 Mapping.Modify/Audit 权限。

测试目标需要在 /mapping/notifications 上具备 Mapping.Use、Mapping.Audit 或 Mapping.Modify 权限。

17.7 通知模式

备份任务配置包含 notification-mode 选项，可取以下三个值之一。

- auto: 如果在 mailto/Send email to 字段中输入了电子邮件地址，则使用 legacy-sendmail 模式。如果未输入电子邮件地址，则使用 notification-system 模式。
- legacy-sendmail: 通过系统的 sendmail 命令发送通知邮件。通知系统会被绕过，所有已配置的 Proxmox VE 8.1 之前版本的通知行为。
- notification-system: 使用新的灵活通知系统。

如果未设置 notification-mode 选项，Proxmox VE 默认使用 auto。

legacy-sendmail 模式可能会在 Proxmox VE 未来版本中移除。

17.8 覆盖通知模板

Proxmox VE 使用 Handlebars 模板渲染通知。Proxmox VE 提供的原始模板存储在 /usr/share/pv

可以通过在覆盖目录 /etc/pve/notification-templates/default/ 中提供自定义模板文件来覆 VE 会首先尝试从覆盖目录加载模板。如果该模板不存在或渲染失败，则使用原始模板。

模板文件遵循 <type>-<body|subject>.<html|txt>.hbs 命名约定。例如，vzdump-body.htm 包含用于渲染备份通知 HTML 版本的模板，而 package-updates-subject.txt.hbs 用于渲染可用

基于电子邮件的通知目标（例如 sendmail 和 smtp）始终发送同时包含 HTML 和纯文本部分的 multi-part 消息。因此，渲染电子邮件消息时会同时使用 <type>-body.html.hbs 和 <type>-body 模板。所有其他通知目标类型只使用 <type>-body.txt.hbs 模板。

17.9 常用命令示例

查看通知配置文件：

```
# cat /etc/pve/notifications.cfg
```

查看包含敏感字段的私有通知配置文件权限：

```
# ls -l /etc/pve/priv/notifications.cfg
```

Chapter 18

重要服务守护进程

18.1 pvedaemon - Proxmox VE API 守护进程

该守护进程在 127.0.0.1:85 上公开完整的 Proxmox VE API。它以 root 身份运行，并有权限执行所

Note
该守护进程仅监听本地地址，因此无法从外部访问。pveproxy 守护进程负责向外部 公开 API。

18.2 pveproxy - Proxmox VE API 代理守护进程

该守护进程通过 HTTPS 在 TCP 端口 8006 上公开完整的 Proxmox VE API。它以 www-data 用户身份运行，权限非常有限。需要更高权限的操作会转发给本地 pvedaemon。

发往其他节点的请求会自动转发到相应节点。这意味着只需连接到单个 Proxmox VE 节点，就可以管理

18.2.1 基于主机的访问控制

可以配置类似“apache2”的访问控制列表。相关值从 /etc/default/pveproxy 文件读取。例如：

```
ALLOW_FROM="10.0.0.1-10.0.0.5,192.168.0.0/22"
DENY_FROM="all"
POLICY="allow"
```

IP 地址可以使用 Net::IP 能识别的任意语法指定。名称 all 是 0/0 和 ::/0 的别名，表示所有 IPv4 和 IPv6 地址。

默认策略为 allow。

匹配	**POLICY=deny**	**POLICY=allow**
仅匹配 Allow	allow	allow
仅匹配 Deny	deny	deny

匹配	**POLICY=deny**	**POLICY=allow**
无匹配	deny	allow
同时匹配 Allow 与 Deny	deny	allow

18.2.2 监听 IP 地址

默认情况下，pveproxy 和 spiceproxy 守护进程监听通配地址，并接受来自 IPv4 和 IPv6 客户端的连

通过在 /etc/default/pveproxy 中设置 LISTEN_IP，可以控制 pveproxy 和 spiceproxy 守护进程绑定到哪个 IP 地址。该 IP 地址需要已在系统上配置。

将 sysctl net.ipv6.bindv6only 设置为非默认值 1 会导致这些守护进程只接受 IPv6 客户端连接，sysctl 设置，或将 LISTEN_IP 设置为 0.0.0.0（这将只允许 IPv4 客户端）。

LISTEN_IP 可用于将套接字限制到内部接口，从而减少对公网的暴露，例如：

```
LISTEN_IP="192.0.2.1"
```

同样，也可以设置 IPv6 地址：

```
LISTEN_IP="2001:db8:85a3::1"
```

请注意，如果要指定 link-local IPv6 地址，需要同时提供接口名称。例如：

```
LISTEN_IP="fe80::c463:8cff:feb9:6a4e%vmbr0"
```

Warning
集群中的节点需要访问 pveproxy 进行通信，并且可能位于不同子网中。不建议在集群系统上设置 LISTEN_IP。

要应用该变更，需要重启节点，或完整重启 pveproxy 和 spiceproxy 服务：

```
systemctl restart pveproxy.service spiceproxy.service
```

Note
与 reload 不同，restart pveproxy 服务可能中断一些长时间运行的 worker 进程，例如来自虚拟客户机的运行中控制台或 shell。因此，请在维护窗口内应用此变更。

常用命令示例

以下命令可用于查看相关服务状态：

```
systemctl status pveproxy.service
systemctl status spiceproxy.service
```

18.2.3 SSL 加密套件

可以通过 /etc/default/pveproxy 中的 CIPHERS（TLS ⇐ 1.2）和 CIPHERSUITES （TLS >= 1.3）键定义加密算法列表。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"
CIPHERSUITES="TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256: ↩
   TLS_AES_128_GCM_SHA256"
```

上面是默认值。所有可用选项请参见 openssl 软件包中的 ciphers(1) 手册页。

此外，可以在 /etc/default/pveproxy 中设置由客户端选择使用的加密算法（默认使用客户端和 pveproxy 都可用的列表中的第一个加密算法）：

```
HONOR_CIPHER_ORDER=0
```

18.2.4 支持的 TLS 版本

pveproxy 无条件禁用不安全的 SSL 2 和 SSL 3。较新的 OpenSSL 版本默认禁用低于 1.1 的 TLS 版本，pveproxy 会遵循该设置（见 /etc/ssl/openssl.cnf）。

要禁用 TLS 1.2 或 1.3，请在 /etc/default/pveproxy 中设置以下内容：

```
DISABLE_TLS_1_2=1
```

或者分别设置：

```
DISABLE_TLS_1_3=1
```

Note
除非有特定原因，否则不建议手动调整支持的 TLS 版本。

18.2.5 Diffie-Hellman 参数

可以在 /etc/default/pveproxy 中将 DHPARAMS 设置为包含 PEM 格式 DH 参数的文件路径，以定义要使用的 Diffie-Hellman 参数，例如：

```
DHPARAMS="/path/to/dhparams.pem"
```

如果未设置该选项，将使用内置的 skip2048 参数。

Note
只有在协商出使用 DH 密钥交换算法的加密套件时，才会使用 DH 参数。

18.2.6 替代 HTTPS 证书

可以将使用的证书更改为外部证书，或更改为通过 ACME 获取的证书。

如果存在，pveproxy 会使用 /etc/pve/local/pveproxy-ssl.pem 和 /etc/pve/local/pvepr
/etc/pve/local/pve-ssl.pem 和 /etc/pve/local/pve-ssl.key。私钥不能使用 passphras

可以通过在 /etc/default/pveproxy 中设置 TLS_KEY_FILE，覆盖证书私钥 /etc/pve/local/p
的位置，例如：

```
TLS_KEY_FILE="/secrets/pveproxy.key"
```

Note
内置 ACME 集成不会遵循该设置。

详细信息请参见文档中的主机系统管理章节。

18.2.7 响应压缩

默认情况下，如果客户端支持，pveproxy 会对可压缩内容使用 gzip HTTP 级压缩。 可在 /etc/defa
中禁用该功能：

```
COMPRESSION=0
```

18.2.8 真实客户端 IP 日志记录

默认情况下，pveproxy 会记录发送请求的客户端 IP 地址。当 pveproxy 前面有代理服务器时，
可能希望记录发起请求的客户端 IP，而不是代理 IP。

要启用对代理设置的 HTTP 头的处理以用于日志记录，请将 PROXY_REAL_IP_HEADER 设置为用于获取
IP 的头名称。例如：

```
PROXY_REAL_IP_HEADER="X-Forwarded-For"
```

该头中传入的任何无效值都会被忽略。

默认行为是在所有传入请求上记录该头中的值。要定义可信任、允许设置上述 HTTP 头的代理服务器列
请设置 PROXY_REAL_IP_ALLOW_FROM，例如：

```
PROXY_REAL_IP_ALLOW_FROM="192.168.0.2"
```

PROXY_REAL_IP_ALLOW_FROM 设置也支持与 ALLOW_FROM 和 DENY_FROM 类似的值。

IP 地址可以使用 Net::IP 能识别的任意语法指定。名称 all 是 0/0 和 ::/0 的别名，表示所有
IPv4 和 IPv6 地址。

18.3 pvestatd - Proxmox VE 状态守护进程

该守护进程会定期查询虚拟机、存储和容器的状态。查询结果会发送到集群中的所有节点。

18.3.1 常用命令示例

```
systemctl status pvestatd.service
journalctl -u pvestatd.service
```

18.4 spiceproxy - SPICE 代理服务

SPICE（Simple Protocol for Independent Computing Environments）是一种开放的远程计算解决
（例如键盘、鼠标、音频）。主要使用场景是远程访问虚拟机和容器。

该守护进程监听 TCP 端口 3128，并实现一个 HTTP 代理，用于将 SPICE 客户端发出的 *CONNECT* 请求转发到正确的 Proxmox VE VM。它以用户 `www-data` 运行，并且权限非常 有限。

18.4.1 基于主机的访问控制

可以配置类似 "apache2" 的访问控制列表。其值从 `/etc/default/pveproxy` 文件读取。 详细信息
`pveproxy` 文档。

常用命令示例

```
# spiceproxy status
# systemctl status spiceproxy.service
```

18.5 pvescheduler - Proxmox VE 调度器守护进程

此守护进程负责按照计划启动作业，例如复制作业和 vzdump 作业。

对于 vzdump 作业，它会从 `/etc/pve/jobs.cfg` 文件读取配置。

18.5.1 常用命令示例

查看调度器服务状态和相关日志时，可使用标准 Debian 系统命令：

```
# systemctl status pvescheduler
# journalctl -u pvescheduler
```

Chapter 19

实用命令行工具

19.1 pvesubscription - 订阅管理

此工具用于管理 Proxmox VE 订阅。

19.1.1 常用命令示例

查看当前节点的订阅信息：

```
pvesubscription get
```

刷新本地订阅状态缓存：

```
pvesubscription update
```

19.2 pveperf - Proxmox VE 基准测试脚本

尝试收集挂载在 PATH（默认使用 /）上的硬盘对应的 CPU/硬盘性能数据：

CPU BOGOMIPS
: 所有 CPU 的 bogomips 总和

REGEX/SECOND
: 每秒正则表达式处理次数（perl 性能测试），应高于 300000

HD SIZE
: 硬盘大小

BUFFERED READS
: 简单的硬盘读取测试。现代硬盘应至少达到 40 MB/sec

AVERAGE SEEK TIME
: 测试平均寻道时间。快速 SCSI 硬盘可达到 < 8 毫秒。 常见 IDE/SATA 磁盘的数值通常为 15 到 20 ms。

FSYNCS/SECOND
数值应大于 200（应在 RAID 控制器上启用 `write back` 缓存模式，该模式需要电池备份缓存（

DNS EXT
解析外部 DNS 名称的平均时间

DNS INT
解析本地 DNS 名称的平均时间

常用命令示例：

```
# pveperf
# pveperf /var/lib/vz
```

19.3 pvebcache - bcache 管理工具

pvebcache 是 PXVIRT 的 bcache 管理工具，用于创建、注册、停止 bcache 后端设备，以及创建、
bcache 缓存设备。

bcache 可以将高速块设备（例如 NVMe SSD）作为缓存设备，为较慢的后端块设备提供缓存能力。

Caution
创建 bcache 后端设备或缓存设备会修改目标块设备元数据。执行相关命令前，请确认设备上没有需要保留的数据，并确认设备没有被系统或虚拟机使用。

19.3.1 后端设备管理

创建后端设备

`pvebcache create` 用于将一个磁盘设备转换为 bcache 后端设备。

```
# pvebcache create /dev/sdb
```

以上命令会将 `/dev/sdb` 转换为 bcache 后端设备。

查看设备列表

`pvebcache list` 用于查看当前 bcache 后端设备和缓存设备。

```
# pvebcache list
name       type       backend-dev          cache-dev             state  ↩
             size            cachemode
bcache0    backend    sdc                  unknown               Running  ↩
           80GB            writeback
nvme0n1    cache      unknown              unknown               Running  ↩
           32GB            unknown
```

停止后端设备

pvebcache stop 用于停止指定 bcache 后端设备。

```
# pvebcache stop bcache0
```

重新注册后端设备

pvebcache register 用于重新注册已经创建过的 bcache 后端设备。

```
# pvebcache register /dev/sdb
```

19.3.2 缓存设备管理

创建缓存设备

pvebcache cache create 用于将指定块设备创建为 bcache 缓存设备。

```
# pvebcache cache create /dev/nvme0n1p1
```

停止缓存设备

pvebcache cache stop 用于停止指定缓存设备。如果该缓存设备正在被后端设备使用，则无法停止

```
# pvebcache cache stop /dev/nvme0n1p1
```

绑定缓存设备

pvebcache cache attach 用于将缓存设备绑定到 bcache 后端设备。

```
# pvebcache cache attach bcache0 --cache /dev/nvme0n1p1
```

分离缓存设备

pvebcache cache detach 用于将 bcache 后端设备与当前缓存设备分离。

```
# pvebcache cache detach bcache0
```

19.3.3 缓存策略

pvebcache cache set 用于设置 bcache 后端设备的缓存策略和相关参数。

cachemode
缓存模式。支持以下取值：

- writethrough
- writeback
- writearound
- none

sequential
顺序 IO 阈值，单位为 KiB。

wb-percent
回写缓存比例。

clear-stats
清除统计数据。

以下示例将 bcache0 的缓存模式设置为 writeback，顺序 IO 阈值设置为 4096 KiB，回写缓存比例设
20：

```
# pvebcache cache set bcache0 --cachemode writeback --sequential 4096 --wb- ↩
   percent 20
```

19.3.4 常用命令示例

```
# pvebcache list
# pvebcache create /dev/sdb
# pvebcache cache create /dev/nvme0n1p1
# pvebcache cache attach bcache0 --cache /dev/nvme0n1p1
# pvebcache cache set bcache0 --cachemode writeback --sequential 4096 --wb- ↩
   percent 20
```

19.4 Proxmox VE API 的 Shell 接口

Proxmox VE 管理工具（pvesh）允许直接调用 API 函数，而无需使用 REST/HTTPS 服务器。

Note
只有 *root* 允许执行该操作。

19.4.1 示例

获取集群中的节点列表

```
# pvesh get /nodes
```

获取数据中心可用选项列表

```
# pvesh usage cluster/options -v
```

将 HTML5 NoVNC 控制台设置为数据中心的默认控制台

```
# pvesh set cluster/options -console html5
```

常用命令示例

```
# pvesh get /version
# pvesh get /nodes
# pvesh usage cluster/options -v
```

Chapter 20

常见问题

Note
新的 FAQ 会追加到本节底部。

1. *Proxmox VE* 基于哪个发行版？
   Proxmox VE 基于 OpenEuler GNU/Linux。
2. *Proxmox VE* 项目使用什么许可证？
   Proxmox VE 代码采用 GNU Affero General Public License 第 3 版授权。
3. *Proxmox VE* 能在 *32* 位处理器上运行吗？
   Proxmox VE 仅适用于 64 位 CPU（AMD 或 Intel）。该平台没有支持 32 位的计划。

   Note
   虚拟机和容器可以是 32 位或 64 位。

4. 我的 *CPU* 是否支持虚拟化？
   要检查 CPU 是否兼容虚拟化，请在以下命令输出中查找 `vmx` 或 `svm` 标记：
   ```
   egrep '(vmx|svm)' /proc/cpuinfo
   ```
5. 支持的 *Intel CPU*
   带有 Intel Virtualization Technology (Intel VT-x) 支持的 64 位处理器。（支持 Intel VT 和 64 位的处理器列表）
6. 支持的 *AMD CPU*
   带有 AMD Virtualization Technology (AMD-V) 支持的 64 位处理器。
7. 什么是容器/虚拟环境（*VE*）/虚拟专用服务器（*VPS*）？
   在容器语境中，这些术语都指操作系统级虚拟化的概念。操作系统级虚拟化是一种虚拟化方法，其中操作系统内核允许多个隔离实例共享同一个内核。谈到 LXC 时，我们称这类实例为容器。由于容器使用宿主机内核，而不是模拟完整操作系统，因此开销更低，但仅限于 Linux 客户机。

8. 什么是 *QEMU/KVM* 客户机（或虚拟机）？

   QEMU/KVM 客户机（或虚拟机）是使用 QEMU 和 Linux KVM 内核模块在 Proxmox VE 下虚拟化运行的客户机系统。

9. 什么是 *QEMU*？

   QEMU 是一个通用的开源机器模拟器和虚拟化程序。QEMU 使用 Linux KVM 内核模块，通过直接在宿主机 CPU 上执行客户机代码来实现接近原生的性能。它不限于 Linux 客户机，而是允许运行任意操作系统。

10. 我的 *Proxmox VE* 版本会支持多久？

    Proxmox VE 版本的支持周期至少会持续到对应 OpenEuler 版本进入 stable 阶段。Proxmox VE 使用滚动发布模型， 始终建议使用最新稳定版本。

Proxmox VE 版本	**OpenEuler** 版本	首次发布	**OpenEuler EOL**	**Proxmox EOL**
Proxmox VE 8	OpenEuler 24.03	2024-05	tba	tba

11. 如何将 *Proxmox VE* 升级到下一个小版本？

    小版本升级，例如从 Proxmox VE 7.1 升级到 7.2 或 7.3，可以像普通更新一样完成。 但仍应查看发行说明，了解任何相关的重要变更或破坏性变更。

    执行更新时，可以使用 Web UI 的 *Node → Updates* 面板，或通过 CLI 执行：

    ```
    dnf makecache
    dnf update
    ```

    Note
    始终确保已正确设置 软件包仓库， 并且只有在 `dnf makecache` 没有遇到任何错误时，才继续执行实际升级。

12. 如何将 *Proxmox VE* 升级到下一个主版本？

    也支持主版本升级，例如从 Proxmox VE 4.4 升级到 5.0。这类升级必须仔细规划和测试，并且在没有当前备份可用时 绝不 应启动。

    尽管具体升级步骤取决于各自环境，项目仍提供了关于如何执行升级的一般说明和建议：

    - Upgrade from Proxmox VE 7 to 8
    - Upgrade from Proxmox VE 6 to 7
    - Upgrade from Proxmox VE 5 to 6
    - Upgrade from Proxmox VE 4 to 5
    - Upgrade from Proxmox VE 3 to 4

13. *LXC vs LXD vs Proxmox Containers vs Docker*

    LXC 是 Linux 内核容器隔离功能的用户空间接口。通过强大的 API 和简单工具， 它让 Linux 用户可以轻松创建和管理系统容器。LXC 与早期的 OpenVZ 一样， 目标是 系统虚拟化。因此，它 ssh 登录、 添加用户、运行 apache 等。

LXD 构建在 LXC 之上，用于提供新的、更好的用户体验。在底层，LXD 通过 `liblxc` 及其 Go 绑定使用 LXC 来创建和管理容器。它基本上是 LXC 工具和发行版模板系统的替代方案，并增加了可通过网络控制所带来的功能。

Proxmox Containers 指使用 Proxmox Container Toolkit（`pct`）创建和管理的容器。它们同样面向 系统虚拟化，并以 LXC 作为容器能力的基础。Proxmox Container Toolkit（`pct`）与 Proxmox VE 紧密集成。这意味着它了解集群设置，并可以使用与 QEMU 虚拟机（VM）相同的网络和存储资源。甚至可以使用 Proxmox VE 防火墙、创建和还原备份，或通过 HA 框架管理容器。所有内容都可以通过 Proxmox VE API 经由网络控制。

Docker 的目标是在隔离、自包含的环境中运行 单个 应用程序。这类容器通常称为“应用容器”，可 Docker Engine 命令行界面从宿主机管理 Docker 实例。不建议直接在 Proxmox VE 宿主机上运行 docker。

Note
如果要运行应用容器，例如 `Docker` 镜像，最佳做法是在 Proxmox QEMU 虚拟机内运行它们。

---

14. *PXVIRT* 和 *PVE* 是什么关系？

    PXVIRT 是基于 Proxmox VE 开源代码和相关生态组件适配、独立维护的虚拟化平台。 它继承了 PVE 中经过验证的虚拟机、容器、集群、存储和 Web 管理等核心机制， 同时面向 openEuler 和多架构环境进行适配、打包和扩展，属于Proxmox VE的衍生品，使用AGPL-3.0协议开源。

    您可以在https://github.com/jiangcuo/pxvirt 仓库中，获取我们的全部源码. 从 https://gitea.pxcloud/proxmox-rpms，获取我们的OpenEuler版本的全部源码。

    PXVIRT 不是 Proxmox Server Solutions GmbH 的官方产品，也不隶属于 Proxmox 项目。Proxmox 和 Proxmox VE 是 Proxmox Server Solutions GmbH 的商标。

Chapter 21

参考文献

21.1 关于 Proxmox VE 的书籍

[1] [Ahmed16] Wasim Ahmed. Mastering Proxmox - Third Edition. Packt Publishing, 2017. ISBN 978-1788397605

[2] [Ahmed15] Wasim Ahmed. Proxmox Cookbook. Packt Publishing, 2015. ISBN 978-1783980901

[3] [Cheng14] Simon M.C. Cheng. Proxmox High Availability. Packt Publishing, 2014. ISBN 978-1783980888

[4] [Goldman16] Rik Goldman. Learning Proxmox VE. Packt Publishing, 2016. ISBN 978-1783981786

[5] [Surber16]] Lee R. Surber. Virtualization Complete: Business Basic Edition. Linux Solutions (LRS-TEK), 2016. ASIN B01BBVQZT6

21.2 关于相关技术的书籍

[6] [Hertzog13] Raphaël Hertzog, Roland Mas., Freexian SARL The Debian Administrator's Handbook: Debian Bullseye from Discovery to Mastery, Freexian, 2021. ISBN 979-10-91414-20-3

[7] [Bir96] Kenneth P. Birman. Building Secure and Reliable Network Applications. Manning Publications Co, 1996. ISBN 978-1884777295

[8] [Walsh10] Norman Walsh. DocBook 5: The Definitive Guide. O'Reilly & Associates, 2010. ISBN 978-0596805029

[9] [Richardson07] Leonard Richardson & Sam Ruby. RESTful Web Services. O'Reilly Media, 2007. ISBN 978-0596529260

[10] [Singh15] Karan Singh. Learning Ceph. Packt Publishing, 2015. ISBN 978-1783985623

[11] [Singh16] Karan Signh. Ceph Cookbook Packt Publishing, 2016. ISBN 978-1784393502

[12] [Mauerer08] Wolfgang Mauerer. Professional Linux Kernel Architecture. John Wiley & Sons, 2008. ISBN 978-0470343432

[13] [Loshin03] Pete Loshin, IPv6: Theory, Protocol, and Practice, 2nd Edition. Morgan Kaufmann, 2003. ISBN 978-1558608108

[14] [Loeliger12] Jon Loeliger & Matthew McCullough. Version Control with Git: Powerful tools and techniques for collaborative software development. O'Reilly and Associates, 2012. ISBN 978-1449316389

[15] [Kreibich10] Jay A. Kreibich. Using SQLite, O'Reilly and Associates, 2010. ISBN 978-0596521189

21.3 关于相关主题的书籍

[16] [Bessen09] James Bessen & Michael J. Meurer, Patent Failure: How Judges, Bureaucrats, and Lawyers Put Innovators at Risk. Princeton Univ Press, 2009. ISBN 978-0691143217

Appendix A

命令行界面

A.1 常规

关于选项在历史上大小写风格不统一的问题，请参见 配置文件的相关章节。

A.2 输出格式选项 [FORMAT_OPTIONS]

可以使用 `--output-format` 参数指定输出格式。默认格式 `text` 使用 ASCII 字符为表格绘制边框，并

- Unix epoch 会显示为 ISO 8601 日期字符串。
- 持续时间会显示为周/天/小时/分钟/秒的组合，例如 `1d 5h`。
- 字节大小会包含单位（B、KiB、MiB、GiB、TiB、PiB）。
- 小数会显示为百分比，例如 1.0 会显示为 100%。

也可以使用 `--quiet` 选项完全抑制输出。

--human-readable <boolean> (*default* = 1)
调用输出渲染函数以生成人类可读的文本。

--noborder <boolean> (*default* = 0)
不绘制边框（用于 *text* 格式）。

--noheader <boolean> (*default* = 0)
不显示列标题（用于 *text* 格式）。

--output-format <json | json-pretty | text | yaml> (*default* = text)
输出格式。

--quiet <boolean>
禁止打印结果。

A.3 pvesm - Proxmox VE 存储管理器

pvesm <COMMAND> [ARGS] [OPTIONS]

pvesm add <type> <storage> [OPTIONS]

创建新的存储。

<type>: <btrfs | cephfs | cifs | dir | esxi | glusterfs | iscsi | iscsidirect | lvm | lvmthin | nfs | pbs | rbd | zfs | zfspool>
存储类型。

<storage>: <storage ID>
存储标识符。

--authsupported <string>
支持的认证方式。

--base <string>
基础卷。此卷会自动激活。

--blocksize <string>
块大小

--bwlimit [clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>] [,move=<LIMIT>] [,restore=<LIMIT>]
为各种操作设置 I/O 带宽限制（单位为 KiB/s）。

--comstar_hg <string>
comstar views 的主机组

--comstar_tg <string>
comstar views 的目标组

--content <string>
允许的内容类型。

> **Note**
> 值 *rootdir* 用于容器，值 *images* 用于虚拟机。

--content-dirs <string>
覆盖默认内容类型目录。

--create-base-path <boolean> (*default* = yes)
如果基础目录不存在，则创建它。

--create-subdirs <boolean> (*default* = yes)
使用默认结构填充目录。

--data-pool <string>
数据池（仅用于纠删码）

--datastore <string>
Proxmox Backup Server datastore 名称。

--disable <boolean>
用于禁用该存储的标志。

--domain <string>
CIFS 域。

--encryption-key a file containing an encryption key, or the special value "autogen"
加密密钥。使用 *autogen* 可自动生成一个不带口令的密钥。

--export <string>
NFS 导出路径。

--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}
证书 SHA 256 指纹。

--format <qcow2 | raw | subvol | vmdk>
默认镜像格式。

--fs-name <string>
Ceph 文件系统名称。

--fuse <boolean>
通过 FUSE 挂载 CephFS。

--is_mountpoint <string> (*default* = no)
假定给定路径是由外部管理的挂载点；如果未挂载，则认为该存储离线。使用布尔值（yes/no）

--iscsiprovider <string>
iscsi 提供程序

--keyring file containing the keyring to authenticate in the Ceph cluster
客户端 keyring 内容（用于外部集群）。

--krbd <boolean> (*default* = 0)
始终通过 krbd 内核模块访问 rbd。

--lio_tpg <string>
Linux LIO targets 的 target portal group

--master-pubkey a file containing a PEM-formatted master public key

Base64 编码、PEM 格式的 RSA 公钥。用于加密 encryption-key 的副本，该副本会添加到每个加密备份中。

--max-protected-backups <integer> (-1 - N) (*default* = Unlimited for users with Datastore.Allocate privilege, 5 for other users)
每个客户机的最大受保护备份数量。使用 *-1* 表示无限制。

--maxfiles <integer> (0 - N)
已弃用：请改用 *prune-backups*。每台虚拟机的最大备份文件数量。使用 *0* 表示 无限制。

--mkdir <boolean> (*default* = yes)
如果目录不存在则创建它，并填充默认子目录。NOTE: 已弃用，请改用 *create-base-path* 和 *create-subdirs* 选项。

--monhost <string>
monitor 的 IP 地址（用于外部集群）。

--mountpoint <string>
挂载点

--namespace <string>
命名空间。

--nocow <boolean> (*default* = 0)
在文件上设置 NOCOW 标志。它会禁用数据校验和，并在允许 direct I/O 的同时导致数据错误无法恢复。仅当数据安全性不需要高于没有底层 RAID 系统的单块 ext4 格式化磁盘 时才使用

--nodes <string>
该存储配置适用的节点列表。

--nowritecache <boolean>
禁用目标上的写缓存

--options <string>
NFS/CIFS 挂载选项（参见 *man nfs* 或 *man mount.cifs*）

--password <password>
访问共享/datastore 的密码。

--path <string>
文件系统路径。

--pool <string>
池。

--port <integer> (1 - 65535)
使用此端口而不是默认端口连接存储（例如用于 PBS 或 ESXi）。对于 NFS 和 CIFS，请使用 *options* 选项通过挂载选项配置端口。

--portal <string>
iSCSI portal（IP 或 DNS 名称，可带端口）。

--preallocation <falloc | full | metadata | off> (*default* = metadata)
raw 和 qcow2 镜像的预分配模式。对 raw 镜像使用 *metadata* 会得到 preallocation=off。

--prune-backups [keep-all=<1|0>] [,keep-daily=<N>] [,keep-hourly=<N>] [,keep-last=<N>] [,keep-monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>]
保留选项会按较短时间间隔优先处理，其中 --keep-last 最先处理。每个选项覆盖一个 特定时间段
备份，只会考虑更早的备份。

--saferemove <boolean>
移除 LV 时将数据清零。

--saferemove_throughput <string>
擦除吞吐量（cstream -t 参数值）。

--server <string>
服务器 IP 或 DNS 名称。

--server2 <string>
备用 volfile 服务器 IP 或 DNS 名称。

> **Note**
> 需要选项：server

--share <string>
CIFS 共享。

--shared <boolean>
表示这是一个在所有节点（或 *nodes* 选项列出的所有节点）上内容相同的单一存储。 它不会让本

--skip-cert-verification <boolean> (*default* = false)
禁用 TLS 证书验证；仅应在完全可信的网络中启用。

--smbversion <2.0 | 2.1 | 3 | 3.0 | 3.11 | default> (*default* = default)
SMB 协议版本。未设置时为 *default*，会协商客户端和服务器双方均支持的最高 SMB2+ 版本。

--sparse <boolean>
使用稀疏卷

--subdir <string>
要挂载的子目录。

--tagged_only <boolean>
仅使用带有 *pve-vm-ID* 标签的逻辑卷。

--target <string>
iSCSI target。

--thinpool <string>
LVM thin pool LV 名称。

--transport <rdma | tcp | unix>
Gluster 传输：tcp 或 rdma

--username <string>
RBD ID。

--vgname <string>
卷组名称。

--volume <string>
Glusterfs 卷。

pvesm alloc <storage> <vmid> <filename> <size> [OPTIONS]

分配磁盘镜像。

<storage>: <storage ID>
存储标识符。

<vmid>: <integer> (100 - 999999999)
指定所属虚拟机

<filename>: <string>
要创建的文件名称。

<size>: \d+[MG]?
大小，单位为 kilobyte（1024 字节）。可选后缀 *M*（megabyte，1024K）和 *G*（gigabyte，

--format <qcow2 | raw | subvol | vmdk>
镜像格式。

> **Note**
> 需要选项：size

pvesm apiinfo

返回 APIVER 和 APIAGE。

pvesm cifsscan

pvesm scan cifs 的别名。

pvesm export <volume> <format> <filename> [OPTIONS]

内部用于导出卷。

<volume>: <string>
卷标识符

<format>: <btrfs | qcow2+size | raw+size | tar+size | vmdk+size | zfs>
导出流格式

<filename>: <string>
目标文件名

--base (?^i:[a-z0-9_\-]{1,40})
增量流的起始快照

--snapshot (?^i:[a-z0-9_\-]{1,40})
要导出的快照

--snapshot-list <string>
要传输的有序快照列表

--with-snapshots <boolean> (*default* = 0)
是否在流中包含中间快照

pvesm extractconfig <volume>

从 vzdump 备份归档中提取配置。

<volume>: <string>
卷标识符

pvesm free <volume> [OPTIONS]

删除卷

<volume>: <string>
卷标识符

--delay <integer> (1 - 30)
等待任务完成的时间。如果任务在该时间内完成，则返回 *null*。

--storage <storage ID>
存储标识符。

pvesm glusterfsscan

pvesm scan glusterfs 的别名。

pvesm help [OPTIONS]

获取指定命令的帮助信息。

--extra-args <array>
显示特定命令的帮助信息。

--verbose <boolean>
详细输出格式。

pvesm import <volume> <format> <filename> [OPTIONS]

内部用于导入卷。

<volume>: <string>
卷标识符

<format>: <btrfs | qcow2+size | raw+size | tar+size | vmdk+size | zfs>
导入流格式

<filename>: <string>
源文件名。使用 - 表示 stdin；tcp://<IP-or-CIDR> 格式允许使用 TCP 连接； unix://PATH-TO-SOCKET 格式表示使用 UNIX socket 作为输入。否则，该文件会被视为 普通文件。

--allow-rename <boolean> (*default* = 0)
如果请求的卷 ID 已存在，则选择新的卷 ID，而不是抛出错误。

--base (?^i:[a-z0-9_\-]{1,40})
增量流的基础快照

--delete-snapshot (?^i:[a-z0-9_\-]{1,80})
成功后要删除的快照

--snapshot (?^i:[a-z0-9_\-]{1,40})
如果流包含快照，则表示当前状态快照

--with-snapshots <boolean> (*default* = 0)
流是否包含中间快照

pvesm iscsiscan

pvesm scan iscsi 的别名。

pvesm list <storage> [OPTIONS]

列出存储内容。

<storage>: <storage ID>
存储标识符。

--content <string>
仅列出此类型的内容。

--vmid <integer> (100 - 999999999)
仅列出此虚拟机的镜像

pvesm lvmscan

pvesm scan lvm 的别名。

pvesm lvmthinscan

pvesm scan lvmthin 的别名。

pvesm nfsscan

pvesm scan nfs 的别名。

pvesm path <volume>

获取指定卷的文件系统路径

<volume>: <string>
卷标识符

pvesm prune-backups <storage> [OPTIONS]

修剪备份。仅考虑使用标准命名方案的备份。如果未指定 keep 选项，则使用存储配置中 的选项。

<storage>: <storage ID>
存储标识符。

--dry-run <boolean>
仅显示将被修剪的内容，不删除任何内容。

--keep-all <boolean>
保留所有备份。为 true 时与其他选项冲突。

--keep-daily <N>
保留最近 <N> 个不同日期的备份。如果某一天有多个备份，则只保留最新的一个。

--keep-hourly <N>
保留最近 <N> 个不同时段（小时）的备份。如果某一小时有多个备份，则只保留最新的 一个。

--keep-last <N>
保留最后 <N> 个备份。

--keep-monthly <N>
保留最近 <N> 个不同月份的备份。如果某一月份有多个备份，则只保留最新的一个。

--keep-weekly <N>
保留最近 <N> 个不同周的备份。如果某一周有多个备份，则只保留最新的一个。

--keep-yearly <N>
保留最近 <N> 个不同年份的备份。如果某一年有多个备份，则只保留最新的一个。

--type <lxc | qemu>
qemu 或 *lxc*。仅考虑此类型客户机的备份。

--vmid <integer> (100 - 999999999)
仅考虑此客户机的备份。

pvesm remove <storage>

删除存储配置。

<storage>: <storage ID>
存储标识符。

pvesm scan cifs <server> [OPTIONS]

扫描远程 CIFS 服务器。

<server>: <string>
服务器地址（名称或 IP）。

--domain <string>
SMB 域（Workgroup）。

--password <password>
用户密码。

--username <string>
用户名。

pvesm scan glusterfs <server>

扫描远程 GlusterFS 服务器。

<server>: <string>

服务器地址（名称或 IP）。

pvesm scan iscsi <portal>

扫描远程 iSCSI 服务器。

<portal>: <string>

iSCSI portal（IP 或 DNS 名称，可带端口）。

pvesm scan lvm

列出本地 LVM 卷组。

pvesm scan lvmthin <vg>

列出本地 LVM Thin Pool。

<vg>: [a-zA-Z0-9\.\+_][a-zA-Z0-9\.\+_\-]+

无可用描述

pvesm scan nfs <server>

扫描远程 NFS 服务器。

<server>: <string>

服务器地址（名称或 IP）。

pvesm scan pbs <server> <username> --password <string> [OPTIONS] [FORMAT_OPTI

扫描远程 Proxmox Backup Server。

<server>: <string>

服务器地址（名称或 IP）。

<username>: <string>

用户名或 API token-ID。

--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}

证书 SHA 256 指纹。

--password <string>

用户密码或 API token secret。

--port <integer> (1 - 65535) (*default* = 8007)

可选端口。

pvesm scan zfs

扫描本地节点上的 zfs pool 列表。

pvesm set <storage> [OPTIONS]

更新存储配置。

<storage>: <storage ID>
存储标识符。

--blocksize <string>
块大小

--bwlimit [clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>] [,move=<LIMIT>] [,restore=<LIMIT>]
为各种操作设置 I/O 带宽限制（单位为 KiB/s）。

--comstar_hg <string>
comstar views 的主机组

--comstar_tg <string>
comstar views 的目标组

--content <string>
允许的内容类型。

> **Note**
> 值 *rootdir* 用于容器，值 *images* 用于虚拟机。

--content-dirs <string>
覆盖默认内容类型目录。

--create-base-path <boolean> (*default* = yes)
如果基础目录不存在，则创建它。

--create-subdirs <boolean> (*default* = yes)
使用默认结构填充目录。

--data-pool <string>
数据池（仅用于纠删码）

--delete <string>
要删除的设置列表。

--digest <string>
如果当前配置文件的 digest 不同，则阻止变更。这可用于防止并发修改。

--disable <boolean>
用于禁用该存储的标志。

--domain <string>
CIFS 域。

--encryption-key a file containing an encryption key, or the special value "autogen"
加密密钥。使用 *autogen* 可自动生成一个不带口令的密钥。

--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}
证书 SHA 256 指纹。

--format <qcow2 | raw | subvol | vmdk>
默认镜像格式。

--fs-name <string>
Ceph 文件系统名称。

--fuse <boolean>
通过 FUSE 挂载 CephFS。

--is_mountpoint <string> (*default* = no)
假定给定路径是由外部管理的挂载点；如果未挂载，则认为该存储离线。使用布尔值（yes/no）

--keyring file containing the keyring to authenticate in the Ceph cluster
客户端 keyring 内容（用于外部集群）。

--krbd <boolean> (*default* = 0)
始终通过 krbd 内核模块访问 rbd。

--lio_tpg <string>
Linux LIO targets 的 target portal group

--master-pubkey a file containing a PEM-formatted master public key

Base64 编码、PEM 格式的 RSA 公钥。用于加密 encryption-key 的副本，该副本会添加到每个加密备份中。

--max-protected-backups <integer> (-1 - N) (*default* = Unlimited for users with Datastore.Allocate privilege, 5 for other users)
每个客户机的最大受保护备份数量。使用 *-1* 表示无限制。

--maxfiles <integer> (0 - N)
已弃用：请改用 *prune-backups*。每台虚拟机的最大备份文件数量。使用 *0* 表示 无限制。

--mkdir <boolean> (*default* = yes)
如果目录不存在则创建它，并填充默认子目录。NOTE: 已弃用，请改用 *create-base-path* 和 *create-subdirs* 选项。

--monhost <string>
monitor 的 IP 地址（用于外部集群）。

--mountpoint <string>
挂载点

--namespace <string>
命名空间。

--nocow <boolean> (*default* = 0)
在文件上设置 NOCOW 标志。它会禁用数据校验和，并在允许 direct I/O 的同时导致数据错误无法恢复。仅当数据安全性不需要高于没有底层 RAID 系统的单块 ext4 格式化磁盘 时才使用

--nodes <string>
该存储配置适用的节点列表。

--nowritecache <boolean>
禁用目标上的写缓存

--options <string>
NFS/CIFS 挂载选项（参见 *man nfs* 或 *man mount.cifs*）

--password <password>
访问共享/datastore 的密码。

--pool <string>
池。

--port <integer> (1 - 65535)
使用此端口而不是默认端口连接存储（例如用于 PBS 或 ESXi）。对于 NFS 和 CIFS，请使用 *options* 选项通过挂载选项配置端口。

--preallocation <falloc | full | metadata | off> (*default* = metadata)
raw 和 qcow2 镜像的预分配模式。对 raw 镜像使用 *metadata* 会得到 preallocation=off。

--prune-backups [keep-all=<1|0>] [,keep-daily=<N>] [,keep-hourly=<N>] [,keep-last=<N>] [,keep-monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>]
保留选项会按较短时间间隔优先处理，其中 --keep-last 最先处理。每个选项覆盖一个 特定时间段
备份，只会考虑更早的备份。

--saferemove <boolean>
移除 LV 时将数据清零。

--saferemove_throughput <string>
擦除吞吐量（cstream -t 参数值）。

--server <string>
服务器 IP 或 DNS 名称。

--server2 <string>
备用 volfile 服务器 IP 或 DNS 名称。

> **Note**
> 需要选项：server

--shared <boolean>
表示这是一个在所有节点（或 *nodes* 选项列出的所有节点）上内容相同的单一存储。 它不会让本

--skip-cert-verification <boolean> (*default* = false)
禁用 TLS 证书验证；仅应在完全可信的网络中启用。

--smbversion <2.0 | 2.1 | 3 | 3.0 | 3.11 | default> (*default* = default)
SMB 协议版本。未设置时为 *default*，会协商客户端和服务器双方均支持的最高 SMB2+ 版本。

--sparse <boolean>
使用稀疏卷

--subdir <string>
要挂载的子目录。

--tagged_only <boolean>
仅使用带有 *pve-vm-ID* 标签的逻辑卷。

--transport <rdma | tcp | unix>
Gluster 传输：tcp 或 rdma

--username <string>
RBD ID。

pvesm status [OPTIONS]

获取所有 datastore 的状态。

--content <string>
仅列出支持此内容类型的存储。

--enabled <boolean> (*default* = 0)
仅列出已启用的存储（未在配置中禁用）。

--format <boolean> (*default* = 0)
包含格式相关信息

--storage <storage ID>
仅列出指定存储的状态

--target <string>
如果 target 不同于 *node*，则只列出内容可被此 *node* 和指定 *target* 节点访问的 共享存储。

pvesm zfsscan

pvesm scan zfs 的别名。

常用命令示例

```
# pvesm status
# pvesm list <storage>
# pvesm prune-backups <storage> --dry-run
# pvesm scan nfs <server>
```

A.4 pvesubscription - Proxmox VE 订阅管理器

pvesubscription <COMMAND> [ARGS] [OPTIONS]

pvesubscription delete

删除此节点的订阅密钥。

pvesubscription get

读取订阅信息。

pvesubscription help [OPTIONS]

获取指定命令的帮助信息。

--extra-args <array>
显示特定命令的帮助信息。

--verbose <boolean>
详细输出格式。

pvesubscription set <key>

设置订阅密钥。

<key>: \s*pve([1248])([cbsp])-[0-9a-f]{10}\s*
Proxmox VE 订阅密钥

pvesubscription set-offline-key <data>

仅供内部使用！要设置离线密钥，请改用 proxmox-offline-mirror-helper 软件包。

<data>: <string>
已签名的订阅信息 blob

pvesubscription update [OPTIONS]

更新订阅信息。

--force <boolean> (*default* = 0)
即使本地缓存仍然有效，也始终连接到服务器。

常用命令示例

```
# pvesubscription get
# pvesubscription update
```

A.5 pveperf - Proxmox VE 基准测试脚本

pveperf [PATH]

常用命令示例

```
# pveperf
# pveperf /var/lib/vz
```

A.6 pvebcache - PXVIRT bcache 管理工具

pvebcache <COMMAND> [ARGS] [OPTIONS]

pvebcache create <device>

将指定磁盘设备转换为 bcache 后端设备。

<device>: <string>
要转换为 bcache 后端设备的块设备，例如 /dev/sdb。

pvebcache list

列出当前 bcache 后端设备和缓存设备。

pvebcache stop <name>

停止指定 bcache 后端设备。

<name>: <string>
bcache 后端设备名称，例如 bcache0。

pvebcache register <device>

重新注册已经创建过的 bcache 后端设备。

<device>: <string>
要重新注册的后端块设备，例如 /dev/sdb。

pvebcache cache create <device>

创建 bcache 缓存设备。

<device>: <string>
要转换为缓存设备的块设备，例如 /dev/nvme0n1p1。

pvebcache cache stop <device>

停止指定缓存设备。如果缓存设备正在使用，则无法停止。

<device>: <string>
要停止的缓存设备，例如 /dev/nvme0n1p1。

pvebcache cache attach <name> --cache <device>

将缓存设备绑定到 bcache 后端设备。

<name>: <string>
bcache 后端设备名称，例如 bcache0。

--cache <device>
要绑定的缓存设备，例如 /dev/nvme0n1p1。

pvebcache cache detach <name>

将 bcache 后端设备与缓存设备分离。

<name>: <string>
bcache 后端设备名称，例如 bcache0。

pvebcache cache set <name> [OPTIONS]

设置 bcache 后端设备的缓存策略和相关参数。

<name>: <string>
bcache 后端设备名称，例如 bcache0。

--cachemode <none | writearound | writeback | writethrough>
缓存模式。

--sequential <integer>
顺序 IO 阈值，单位为 KiB。

--wb-percent <integer>
回写缓存比例。

--clear-stats <boolean>
清除统计数据。

常用命令示例

```
pvebcache list
pvebcache create /dev/sdb
pvebcache cache create /dev/nvme0n1p1
pvebcache cache attach bcache0 --cache /dev/nvme0n1p1
pvebcache cache set bcache0 --cachemode writeback --sequential 4096 --wb- ↩
   percent 20
```

A.7 pveceph - 管理 Proxmox VE 节点上的 CEPH 服务

pveceph <COMMAND> [ARGS] [OPTIONS]

常用命令示例：

```
pveceph status
pveceph pool ls
```

pveceph createmgr

pveceph mgr create 的别名。

pveceph createmon

pveceph mon create 的别名。

pveceph createosd

pveceph osd create 的别名。

pveceph createpool

pveceph pool create 的别名。

pveceph destroymgr

pveceph mgr destroy 的别名。

pveceph destroymon

pveceph mon destroy 的别名。

pveceph destroyosd

pveceph osd destroy 的别名。

pveceph destroypool

pveceph pool destroy 的别名。

pveceph fs create [OPTIONS]

创建 Ceph 文件系统。

--add-storage <boolean> (*default* = 0)
将已创建的 CephFS 配置为此集群的存储。

--name (?^:^[^:/\s]+$) (*default* = cephfs)
Ceph 文件系统名称。

--pg_num <integer> (8 - 32768) (*default* = 128)
后端数据池的 placement group 数量。元数据池将使用该数量的四分之一。

pveceph fs destroy <name> [OPTIONS]

销毁 Ceph 文件系统。

<name>: <string>
Ceph 文件系统名称。

--remove-pools <boolean> (*default* = 0)
移除此 fs 配置的数据池和元数据池。

--remove-storages <boolean> (*default* = 0)
移除此 fs 配置的所有由 pveceph 管理的存储。

pveceph help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助。

--verbose <boolean>
详细输出格式。

pveceph init [OPTIONS]

创建初始 Ceph 默认配置并设置符号链接。

--cluster-network <string>
声明一个独立的集群网络，OSD 会通过该网络传输 heartbeat、对象复制和恢复流量。

> **Note**
> 需要选项：network

--disable_cephx <boolean> (*default* = 0)
禁用 cephx 认证。

> **Warning**
> cephx 是用于防范中间人攻击的安全功能。只有在网络为私有网络时，才应考虑禁用 cephx。

--min_size <integer> (1 - 7) (*default* = 2)
允许 I/O 时每个对象所需的最少可用副本数。

--network <string>
为所有 Ceph 相关流量使用指定网络。

--pg_bits <integer> (6 - 14) (*default* = 6)
placement group 位数，用于指定默认 placement group 数量。

已弃用。此设置已在较新的 Ceph 版本中弃用。

--size <integer> (1 - 7) (*default* = 3)
每个对象的目标副本数。

pveceph install [OPTIONS]

安装 Ceph 相关软件包。

--allow-experimental <boolean> (*default* = 0)
允许使用实验版本。请谨慎使用。

--repository <enterprise | no-subscription | test> (*default* = enterprise)
要使用的 Ceph 仓库。

--version <quincy | reef | squid> (*default* = quincy)
要安装的 Ceph 版本。

pveceph lspools

pveceph pool ls 的别名。

pveceph mds create [OPTIONS]

创建 Ceph Metadata Server (MDS) 。

--hotstandby <boolean> (*default* = 0)
决定 ceph-mds 守护进程是否应轮询并重放活动 MDS 的日志。MDS 故障时切换更快，但需要更

--name [a-zA-Z0-9]([a-zA-Z0-9\-]*[a-zA-Z0-9])? (*default* = nodename)
mds 的 ID；省略时与 nodename 相同。

pveceph mds destroy <name>

销毁 Ceph Metadata Server。

<name>: [a-zA-Z0-9]([a-zA-Z0-9\-]*[a-zA-Z0-9])?
mds 的名称 (ID) 。

pveceph mgr create [OPTIONS]

创建 Ceph Manager。

--id [a-zA-Z0-9]([a-zA-Z0-9\-]*[a-zA-Z0-9])?
manager 的 ID；省略时与 nodename 相同。

pveceph mgr destroy <id>

销毁 Ceph Manager。

<id>: [a-zA-Z0-9]([a-zA-Z0-9\-]*[a-zA-Z0-9])?
manager 的 ID。

pveceph mon create [OPTIONS]

创建 Ceph Monitor 和 Manager。

--mon-address <string>
覆盖自动检测到的 monitor IP 地址。必须位于 Ceph 的 public network 中。

--monid [a-zA-Z0-9]([a-zA-Z0-9\-]*[a-zA-Z0-9])?
monitor 的 ID；省略时与 nodename 相同。

pveceph mon destroy <monid>

销毁 Ceph Monitor 和 Manager。

<monid>: [a-zA-Z0-9]([a-zA-Z0-9\-]*[a-zA-Z0-9])?
Monitor ID。

pveceph osd create <dev> [OPTIONS]

创建 OSD。

<dev>: <string>
块设备名称。

--crush-device-class <string>
在 CRUSH 中设置 OSD 的设备类别。

--db_dev <string>
block.db 的块设备名称。

--db_dev_size <number> (1 - N) (*default* = bluestore_block_db_size or 10% of OSD size)
block.db 的大小，单位为 GiB。

> **Note**
> 需要选项：db_dev

--encrypted <boolean> (*default* = 0)
启用 OSD 加密。

--osds-per-device <integer> (1 - N)
每个物理设备上的 OSD 服务数量。仅对高速 NVMe 设备有用，可更好地利用其性能。

--wal_dev <string>
block.wal 的块设备名称。

--wal_dev_size <number> (0.5 - N) (*default* = bluestore_block_wal_size or 1% of OSD size)
block.wal 的大小，单位为 GiB。

> **Note**
> 需要选项：wal_dev

pveceph osd destroy <osdid> [OPTIONS]

销毁 OSD。

<osdid>: <integer>
OSD ID。

--cleanup <boolean> (*default* = 0)
设置后，会移除分区表条目。

pveceph osd details <osdid> [OPTIONS] [FORMAT_OPTIONS]

获取 OSD 详细信息。

<osdid>: <string>
OSD 的 ID。

--verbose <boolean> (*default* = 0)
打印详细信息，与 json-pretty 输出格式相同。

pveceph pool create <name> [OPTIONS]

创建 Ceph pool。

<name>: (?^:^[^:/\s]+$)
pool 的名称。必须唯一。

--add_storages <boolean> (*default* = 0; for erasure coded pools: 1)
使用新的 pool 配置虚拟机和容器存储。

--application <cephfs | rbd | rgw> (*default* = rbd)
pool 的应用类型。

--crush_rule <string>
用于在集群中映射对象放置位置的规则。

--erasure-coding k=<integer> ,m=<integer> [,device-class=<class>] [,failure-domain=<domain>] [,profile=<profile>]
为 RBD 创建 erasure coded pool，并创建配套的 replicated pool 用于元数据存储。使用 EC 时，通用 Ceph 选项 *size*、*min_size* 和 *crush_rule* 参数会应用到元数据池。

--min_size <integer> (1 - 7) (*default* = 2)
每个对象的最少副本数。

--pg_autoscale_mode <off | on | warn> (*default* = warn)
pool 的自动 PG 扩缩模式。

--pg_num <integer> (1 - 32768) (*default* = 128)
placement group 数量。

--pg_num_min <integer> (-N - 32768)
最小 placement group 数量。

--size <integer> (1 - 7) (*default* = 3)
每个对象的副本数。

--target_size ^(\d+(\.\d+)?)([KMGT])?$
PG autoscaler 使用的 pool 估算目标大小。

--target_size_ratio <number>
PG autoscaler 使用的 pool 估算目标比例。

pveceph pool destroy <name> [OPTIONS]

销毁 pool。

<name>: <string>
pool 的名称。必须唯一。

--force <boolean> (*default* = 0)
如果为 true，即使 pool 正在使用也会销毁。

--remove_ecprofile <boolean> (*default* = 1)
移除 erasure code profile。适用时默认为 true。

--remove_storages <boolean> (*default* = 0)
移除此 pool 配置的所有由 pveceph 管理的存储。

pveceph pool get <name> [OPTIONS] [FORMAT_OPTIONS]

显示当前 pool 状态。

<name>: <string>
pool 的名称。必须唯一。

--verbose <boolean> (*default* = 0)
启用后，将显示额外数据（例如统计信息）。

pveceph pool ls [FORMAT_OPTIONS]

列出所有 pool 及其设置（这些设置可通过 POST/PUT 端点设置）。

pveceph pool set <name> [OPTIONS]

更改 POOL 设置。

<name>: (?^:^[^:/\s]+$)
pool 的名称。必须唯一。

--application <cephfs | rbd | rgw>
pool 的应用类型。

--crush_rule <string>
用于在集群中映射对象放置位置的规则。

--min_size <integer> (1 - 7)
每个对象的最少副本数。

--pg_autoscale_mode <off | on | warn>
pool 的自动 PG 扩缩模式。

--pg_num <integer> (1 - 32768)
placement group 数量。

--pg_num_min <integer> (-N - 32768)
最小 placement group 数量。

--size <integer> (1 - 7)
每个对象的副本数。

--target_size ^(\d+(\.\d+)?)([KMGT])?$
PG autoscaler 使用的 pool 估算目标大小。

--target_size_ratio <number>
PG autoscaler 使用的 pool 估算目标比例。

pveceph purge [OPTIONS]

销毁 Ceph 相关数据和配置文件。

--crash <boolean>
同时清除 Ceph 崩溃日志 /var/lib/ceph/crash。

--logs <boolean>
同时清除 Ceph 日志 /var/log/ceph。

pveceph start [OPTIONS]

启动 Ceph 服务。

--service (ceph|mon|mds|osd|mgr)(\.[a-zA-Z0-9]([a-zA-Z0-9\-]*[a-zA-Z0-9])?)? (*default* = ceph.target)
Ceph 服务名称。

pveceph status

获取 Ceph 状态。

pveceph stop [OPTIONS]

停止 Ceph 服务。

--service (ceph|mon|mds|osd|mgr)(\.[a-zA-Z0-9]([a-zA-Z0-9\-]*[a-zA-Z0-9])?)? (*default* = ceph.target)
Ceph 服务名称。

A.8 pvenode - Proxmox VE 节点管理

pvenode <COMMAND> [ARGS] [OPTIONS]

pvenode acme account deactivate [<name>]

在 CA 处停用现有 ACME 账户。

<name>: <name> (*default* = default)
ACME 账户配置文件名。

pvenode acme account info [<name>] [FORMAT_OPTIONS]

返回现有 ACME 账户信息。

<name>: <name> (*default* = default)
ACME 账户配置文件名。

pvenode acme account list

ACMEAccount 索引。

pvenode acme account register [<name>] {<contact>} [OPTIONS]

向兼容的 CA 注册新的 ACME 账户。

<name>: <name> (*default* = default)
ACME 账户配置文件名。

<contact>: <string>
联系电子邮件地址。

--directory ^https?://.*
ACME CA directory endpoint 的 URL。

pvenode acme account update [<name>] [OPTIONS]

向 CA 更新现有 ACME 账户信息。注意：不指定任何新账户信息时会触发刷新。

<name>: <name> (*default* = default)
ACME 账户配置文件名。

--contact <string>
联系电子邮件地址。

pvenode acme cert order [OPTIONS]

从兼容 ACME 的 CA 申请新证书。

--force <boolean> (*default* = 0)
覆盖现有自定义证书。

pvenode acme cert renew [OPTIONS]

从 CA 续订现有证书。

--force <boolean> (*default* = 0)
即使距离过期还有 30 天以上，也强制续订。

pvenode acme cert revoke

从 CA 吊销现有证书。

pvenode acme plugin add <type> <id> [OPTIONS]

添加 ACME 插件配置。

<type>: <dns | standalone>
ACME challenge 类型。

<id>: <string>
ACME 插件 ID 名称

--api <1984hosting | acmedns | acmeproxy | active24 | ad | ali | alviy | anx | artfiles | arvan | aurora | autodns | aws | azion | azure | bookmyname | bunny | cf | clouddns | cloudns | cn | conoha | constellix | cpanel | curanet | cyon | da | ddnss | desec | df | dgon | dnsexit | dnshome | dnsimple | dnsservices | doapi | domeneshop | dp | dpi | dreamhost | duckdns | durabledns | dyn | dynu | dynv6 | easydns | edgedns | euserv | exoscale | fornex | freedns | gandi_livedns | gcloud | gcore | gd | geoscaling | googledomains | he | hetzner | hexonet | hostingde | huaweicloud | infoblox | infomaniak | internetbs | inwx | ionos | ionos_cloud | ipv64 | ispconfig | jd | joker | kappernet | kas | kinghost | knot | la | leaseweb | lexicon | limacity | linode | linode_v4 | loopia | lua | maradns | me | miab | misaka | myapi | mydevil | mydnsjp | mythic_beasts | namecheap | namecom | namesilo | nanelo | nederhost | neodigit | netcup | netlify | nic | njalla | nm | nsd | nsone | nsupdate | nw | oci | omglol | one | online | openprovider | openstack | opnsense | ovh | pdns | pleskxml | pointhq | porkbun | rackcorp | rackspace | rage4 | rcode0 | regru | scaleway | schlundtech | selectel | selfhost | servercow | simply | technitium | tele3 | tencent | timeweb | transip | udr | ultra | unoeuro | variomedia | veesp | vercel | vscale | vultr | websupport | west_cn | world4you | yandex360 | yc | zilore | zone | zoneedit | zonomi>
API 插件名称

--data 每行一个键值对的文件，存入插件配置时会进行 **base64url** 编码。
DNS 插件数据（base64 编码）。

--disable <boolean>
用于禁用该配置的标志。

--nodes <string>
集群节点名称列表。

--validation-delay <integer> (0 - 172800) (*default* = 30)
请求验证前额外等待的秒数。可用于应对 DNS 记录 TTL 较长的情况。

pvenode acme plugin config <id> [FORMAT_OPTIONS]

获取 ACME 插件配置。

<id>: <string>
ACME 插件实例的唯一标识符。

pvenode acme plugin list [OPTIONS] [FORMAT_OPTIONS]

ACME 插件索引。

--type <dns | standalone>
仅列出指定类型的 ACME 插件

pvenode acme plugin remove <id>

删除 ACME 插件配置。

<id>: <string>
ACME 插件实例的唯一标识符。

pvenode acme plugin set <id> [OPTIONS]

更新 ACME 插件配置。

<id>: <string>
ACME 插件 ID 名称

--api <1984hosting | acmedns | acmeproxy | active24 | ad | ali | alviy | anx | artfiles | arvan | aurora | autodns | aws | azion | azure | bookmyname | bunny | cf | clouddns | cloudns | cn | conoha | constellix | cpanel | curanet | cyon | da | ddnss | desec | df | dgon | dnsexit | dnshome | dnsimple | dnsservices | doapi | domeneshop | dp | dpi | dreamhost | duckdns | durabledns | dyn | dynu | dynv6 | easydns | edgedns | euserv | exoscale | fornex | freedns | gandi_livedns | gcloud | gcore | gd | geoscaling | googledomains | he | hetzner | hexonet | hostingde | huaweicloud | infoblox | infomaniak | internetbs | inwx | ionos | ionos_cloud | ipv64 | ispconfig | jd | joker | kappernet | kas | kinghost | knot | la | leaseweb | lexicon | limacity | linode | linode_v4 | loopia | lua | maradns | me | miab | misaka | myapi | mydevil | mydnsjp | mythic_beasts | namecheap | namecom | namesilo | nanelo | nederhost | neodigit | netcup | netlify | nic | njalla | nm | nsd | nsone | nsupdate | nw | oci | omglol | one | online | openprovider | openstack | opnsense | ovh | pdns | pleskxml | pointhq | porkbun | rackcorp | rackspace | rage4 | rcode0 | regru | scaleway | schlundtech | selectel | selfhost | servercow | simply | technitium | tele3 | tencent | timeweb | transip | udr | ultra | unoeuro | variomedia | veesp | vercel | vscale | vultr | websupport | west_cn | world4you | yandex360 | yc | zilore | zone | zoneedit | zonomi>
API 插件名称

--data 每行一个键值对的文件，存入插件配置时会进行 **base64url** 编码。
DNS 插件数据（base64 编码）。

--delete <string>
要删除的设置列表。

--digest <string>
如果当前配置文件具有不同 digest，则阻止更改。可用于防止并发修改。

--disable <boolean>
用于禁用该配置的标志。

--nodes <string>
集群节点名称列表。

--validation-delay <integer> (0 - 172800) (*default* = 30)
请求验证前额外等待的秒数。可用于应对 DNS 记录 TTL 较长的情况。

pvenode cert delete [<restart>]

删除自定义证书链和密钥。

<restart>: <boolean> (*default* = 0)
重启 pveproxy。

pvenode cert info [FORMAT_OPTIONS]

获取节点证书信息。

pvenode cert set <certificates> [<key>] [OPTIONS] [FORMAT_OPTIONS]

上传或更新自定义证书链和密钥。

<certificates>: <string>
PEM 编码的证书（链）。

<key>: <string>
PEM 编码的私钥。

--force <boolean> (*default* = 0)
覆盖现有自定义或 ACME 证书文件。

--restart <boolean> (*default* = 0)
重启 pveproxy。

pvenode config get [OPTIONS]

获取节点配置选项。

--property <acme | acmedomain0 | acmedomain1 | acmedomain2 | acmedomain3 | acmedomain4 | acmedomain5 | ballooning-target | description | startall-onboot-delay | wakeonlan> (*default* = all)
仅返回节点配置中的指定属性。

pvenode config set [OPTIONS]

设置节点配置选项。

--acme [account=<name>] [,domains=<domain[;domain;...]>]
节点专用 ACME 设置。

--acmedomain[n] [domain=]<domain> [,alias=<domain>] [,plugin=<name of the plugin configuration>]
ACME 域名和验证插件

--ballooning-target <integer> (0 - 100) (*default* = 80)
ballooning 的 RAM 使用目标（占总内存百分比）

--delete <string>
要删除的设置列表。

--description <string>
节点描述。显示在 Web 界面的节点备注面板中，并作为注释保存在配置文件内。

--digest <string>
如果当前配置文件具有不同 SHA1 digest，则阻止更改。可用于防止并发修改。

--startall-onboot-delay <integer> (0 - 300) (*default* = 0)
启动所有已启用 on-boot 的虚拟客户机之前的初始延迟秒数。

--wakeonlan [mac=]<MAC address> [,bind-interface=<bind interface>] [,broadcast-address=<IPv4 broadcast address>]
节点专用 Wake-on-LAN 设置。

pvenode help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助

--verbose <boolean>
详细输出格式。

pvenode migrateall <target> [OPTIONS]

迁移所有虚拟机和容器。

<target>: <string>
目标节点。

--maxworkers <integer> (1 - N)
并行迁移任务的最大数量。如果未设置，则使用 datacenter.cfg 中的 max_workers。两者必

--vms <string>
仅考虑这些 ID 对应的客户机。

--with-local-disks <boolean>
为本地磁盘启用在线存储迁移

pvenode startall [OPTIONS]

启动位于该节点上的所有虚拟机和容器（默认仅启动 onboot=1 的客户机）。

--force <boolean> (*default* = off)
即使虚拟客户机未设置 onboot 或将其设置为 off，也发出启动命令。

--vms <string>
仅考虑此逗号分隔 VMID 列表中的客户机。

pvenode stopall [OPTIONS]

停止所有虚拟机和容器。

--force-stop <boolean> (*default* = 1)
超时后强制硬停止。

--timeout <integer> (0 - 7200) (*default* = 180)
每个客户机关闭任务的超时时间。根据 force-stop 设置，超时后关闭操作会被直接中止或强制

--vms <string>
仅考虑这些 ID 对应的客户机。

pvenode task list [OPTIONS] [FORMAT_OPTIONS]

读取某个节点的任务列表（已完成任务）。

--errors <boolean> (*default* = 0)
仅列出状态为 ERROR 的任务。

--limit <integer> (0 - N) (*default* = 50)
仅列出该数量的任务。

--since <integer>
仅列出此 UNIX epoch 之后的任务。

--source <active | all | archive> (*default* = archive)
列出归档任务、活动任务或全部任务。

--start <integer> (0 - N) (*default* = 0)
从该偏移位置开始列出任务。

--statusfilter <string>
应返回的任务状态列表。

--typefilter <string>
仅列出此类型的任务（例如 vzstart、vzdump）。

--until <integer>
仅列出此 UNIX epoch 之前的任务。

--userfilter <string>
仅列出此用户的任务。

--vmid <integer> (100 - 999999999)
仅列出此虚拟机的任务。

pvenode task log <upid> [OPTIONS]

读取任务日志。

<upid>: <string>
任务的唯一 ID。

--download <boolean>
是否下载任务日志文件。该参数不能与其他参数同时使用。

--start <integer> (0 - N) (*default* = 0)
读取任务日志时从该行开始。

pvenode task status <upid> [FORMAT_OPTIONS]

读取任务状态。

<upid>: <string>
任务的唯一 ID。

pvenode wakeonlan <node>

尝试通过 Wake-on-LAN 网络包唤醒节点。

<node>: <string>
Wake-on-LAN 网络包的目标节点

A.9 pvesh - Proxmox VE API 的 Shell 接口

pvesh <COMMAND> [ARGS] [OPTIONS]

pvesh create <api_path> [OPTIONS] [FORMAT_OPTIONS]

对 <api_path> 调用 API POST。

<api_path>: <string>
API 路径。

--noproxy <boolean>
禁用自动代理。

pvesh delete <api_path> [OPTIONS] [FORMAT_OPTIONS]

对 <api_path> 调用 API DELETE。

<api_path>: <string>
API 路径。

--noproxy <boolean>
禁用自动代理。

pvesh get <api_path> [OPTIONS] [FORMAT_OPTIONS]

对 <api_path> 调用 API GET。

<api_path>: <string>
API 路径。

--noproxy <boolean>
禁用自动代理。

pvesh help [OPTIONS]
获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助。

--verbose <boolean>
详细输出格式。

pvesh ls <api_path> [OPTIONS] [FORMAT_OPTIONS]
列出 <api_path> 下的子对象。

<api_path>: <string>
API 路径。

--noproxy <boolean>
禁用自动代理。

pvesh set <api_path> [OPTIONS] [FORMAT_OPTIONS]
对 <api_path> 调用 API PUT。

<api_path>: <string>
API 路径。

--noproxy <boolean>
禁用自动代理。

pvesh usage <api_path> [OPTIONS]
打印 <api_path> 的 API 用法信息。

<api_path>: <string>
API 路径。

--command <create | delete | get | set>
API 命令。

--returns <boolean>
包含返回数据的 schema。

--verbose <boolean>
详细输出格式。

常用命令示例

```
pvesh get /nodes
pvesh get /nodes/<node>/status
pvesh ls /nodes/<node>
```

A.10 qm - QEMU/KVM 虚拟机管理器

qm <COMMAND> [ARGS] [OPTIONS]

qm agent

qm guest cmd 的别名。

qm cleanup <vmid> <clean-shutdown> <guest-requested>

清理 tap 设备、vgpu 等资源。在虚拟机关机、崩溃等情况后调用。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<clean-shutdown>: <boolean>
表示 qemu 是否正常关机。

<guest-requested>: <boolean>
表示关机是由客户机请求，还是通过 qmp 请求。

qm clone <vmid> <newid> [OPTIONS]

创建虚拟机/模板的副本。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<newid>: <integer> (100 - 999999999)
克隆的 VMID。

--bwlimit <integer> (0 - N) (*default* = clone limit from datacenter or storage config)
覆盖 I/O 带宽限制（单位 KiB/s）。

--description <string>
新虚拟机的描述。

--format <qcow2 | raw | vmdk>
文件存储的目标格式。仅对完整克隆有效。

--full <boolean>
创建所有磁盘的完整副本。克隆普通虚拟机时总是执行此操作。对于虚拟机模板，默认会尝试创建

--name <string>
设置新虚拟机的名称。

--pool <string>
将新虚拟机添加到指定资源池。

--snapname <string>
快照名称。

--storage <storage ID>
完整克隆的目标存储。

--target <string>
目标节点。仅当原虚拟机位于共享存储上时允许。

qm cloudinit dump <vmid> <type>
获取自动生成的 cloudinit 配置。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<type>: <meta | network | user>
配置类型。

qm cloudinit pending <vmid>
获取包含当前值和待应用值的 cloudinit 配置。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm cloudinit update <vmid>
重新生成并更改 cloudinit 配置驱动器。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm config <vmid> [OPTIONS]
获取已应用待处理配置更改后的虚拟机配置。设置 current 参数可改为获取当前配置。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--current <boolean> (*default* = 0)
获取当前值（而不是待应用值）。

--snapshot <string>
从给定快照获取配置值。

qm create <vmid> [OPTIONS]
创建或还原虚拟机。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--gicversion: <2 | 3 | 4 | max | host > (*default* = host)
设置 ARM 虚拟机的 gicversion。

--virtualization: <boolean> (*default* = 0)
为 arm64 虚拟机启用嵌套虚拟化。

--snapshot=<boolean>
控制 qemu 的快照模式功能。启用后，对磁盘所做的更改是临时的，并会在虚拟机关机时丢

--pxvditemplate <boolean> (*default* = 0)
启用/禁用 PXVDI Template。

--acpi <boolean> (*default* = 1)
启用/禁用 ACPI。

--affinity <string>
用于执行客户机进程的主机核心列表，例如：0,5,8-11

**--agent [enabled=]<1|0> [,freeze-fs-on-backup=<1|0>]
[,fstrim_cloned_disks=<1|0>] [,type=<virtio|isa>]**
启用/禁用与 QEMU Guest Agent 及其属性的通信。

**--amd-sev [type=]<sev-type> [,allow-smt=<1|0>]
[,kernel-hashes=<1|0>] [,no-debug=<1|0>] [,no-key-sharing=<1|0>]**
AMD CPU 提供的 Secure Encrypted Virtualization（SEV）功能。

--arch <aarch64 | x86_64 | riscv64 | loongarch64>
虚拟处理器架构。默认为主机架构。

--archive <string>
备份归档。可以是 .tar 或 .vma 文件的文件系统路径（使用 - 从 stdin 管道传入数据），也可以是
proxmox 存储备份卷标识符。

--args <string>
传递给 kvm 的任意参数。

**--audio0 device=<ich9-intel-hda|intel-hda|AC97>
[,driver=<spice|none>]**
配置音频设备，与 QXL/Spice 结合使用时很有用。

--autostart <boolean> (*default* = 0)
崩溃后自动重启（当前会被忽略）。

--balloon <integer> (0 - N)
虚拟机目标 RAM 容量，单位 MiB。使用 0 会禁用 balloon 驱动。

--bios <ovmf | seabios> (*default* = seabios)
选择 BIOS 实现。

--boot [[legacy=]<[acdn]{1,4}>] [,order=<device[;device...]>]
指定客户机启动顺序。请使用 order= 子属性；不带键或使用 legacy= 的用法已弃用。

--bootdisk (ide|sata|scsi|virtio)\d+
启用从指定磁盘启动。已弃用：请改用 boot: order=foo;bar。

--bwlimit <integer> (0 - N) (*default* = restore limit from datacenter or storage config)
覆盖 I/O 带宽限制（单位 KiB/s）。

--cdrom <volume>
这是选项 -ide2 的别名。

--cicustom [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>]
cloud-init：指定自定义文件，以替换启动时自动生成的文件。

--cipassword <password>
cloud-init：要分配给用户的密码。通常不建议使用此项，请改用 ssh 密钥。另请注意，较旧版本
cloud-init 不支持哈希密码。

--citype <configdrive2 | nocloud | opennebula>
指定 cloud-init 配置格式。默认值取决于配置的操作系统类型（ostype）。Linux 使用 noclou
格式，windows 使用 configdrive2。

--ciupgrade <boolean> (*default* = 1)
cloud-init：首次启动后自动执行软件包升级。

--ciuser <string>
cloud-init：要更改 ssh 密钥和密码的用户名，用于替代镜像中配置的默认用户。

--cores <integer> (1 - N) (*default* = 1)
每个 socket 的核心数。

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

--cpulimit <number> (0 - 128) (*default* = 0)
CPU 使用限制。

--cpuunits <integer> (1 - 262144) (*default* = cgroup v1: 1024, cgroup v2: 100)
虚拟机的 CPU 权重，在 cgroup v2 中会限制到 [1, 10000] 范围内。

--description <string>
虚拟机描述。显示在 Web 界面的虚拟机摘要中，并作为注释保存到配置文件内。

--efidisk0 [file=]<volume> [,efitype=<2m|4m>] [,format=<enum>] [,import-from=<source volume>] [,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。注意，此处

SIZE_IN_GiB，并会将默认 EFI 变量复制到该卷中。使用 STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

--force <boolean>
允许覆盖现有虚拟机。

> **Note**
> 需要选项：`archive`

--freeze <boolean>
启动时冻结 CPU（使用 c monitor 命令开始执行）。

--hookscript <string>
将在虚拟机生命周期的各个步骤中执行的脚本。

--hostpci[n] [[host=]<HOSTPCIID[;HOSTPCIID2...]>] [,device-id=<hex id>] [,legacy-igd=<1|0>] [,mapping=<mapping-id>] [,mdev=<string>] [,pcie=<1|0>] [,rombar=<1|0>] [,romfile=<string>] [,sub-device-id=<hex id>] [,sub-vendor-id=<hex id>] [,vendor-id=<hex id>] [,x-vga=<1|0>] [,ramfb=<1|0>]
将主机 PCI 设备映射到客户机。

--hotplug <string> (*default* = network,disk,usb)
选择性启用热插拔功能。这是一个逗号分隔的热插拔功能列表：network、disk、cpu、memory 和 cloudinit。使用 0 可完全禁用热插拔。使用 1 作为值是默认值 network,disk,usb 的别名。对于 machine version >= 7.1 且 ostype 为 l26 或 windows > 7 的客户机，可以使用 USB 热插拔。

--hugepages <1024 | 2 | any>
启用/禁用 hugepages 内存。

--ide[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>] [,import-from=<source volume>] [,iops=<iops>] [,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>] [,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>] [,iops_wr=<iops>] [,iops_wr_max=<iops>] [,iops_wr_max_length=<seconds>] [,mbps=<mbps>] [,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>] [,media=<cdrom|disk>] [,model=<model>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,secs=<integer>] [,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>]
将卷用作 IDE 硬盘或 CD-ROM（n 为 0 到 3）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。使用 STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

--import-working-storage <storage ID>
启用了 images 内容类型的基于文件的存储，在导入期间用作中间解压存储。默认为源存储。

--ipconfig[n] [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>]
cloud-init：指定对应接口的 IP 地址和网关。

IP 地址使用 CIDR 表示法；网关是可选项，但需要指定同类型 IP。

IP 地址可以使用特殊字符串 dhcp 以使用 DHCP，此时不应显式提供网关。 对于 IPv6，可以使用 auto 以使用无状态自动配置。这需要 cloud-init 19.4 或更新版本。

如果启用了 cloud-init 且既未指定 IPv4 地址也未指定 IPv6 地址，则默认在 IPv4 上使用 dhcp。

--ivshmem size=<integer> [,name=<string>]
虚拟机间共享内存。适用于虚拟机之间或虚拟机与主机之间的直接通信。

--keephugepages <boolean> (*default* = 0)
与 hugepages 一起使用。启用后，虚拟机关机后 hugepages 不会被删除，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>
VNC 服务器的键盘布局。通常不需要此选项，且往往更适合在客户机 OS 内处理。

--kvm <boolean> (*default* = 1)
启用/禁用 KVM 硬件虚拟化。

--live-restore <boolean>
在后台导入或还原期间立即启动虚拟机。

--localtime <boolean>
将实时时钟（RTC）设置为本地时间。如果 ostype 表示 Microsoft Windows OS，则默认启用。

--lock <backup | clone | create | migrate | rollback | snapshot | snapshot-delete | suspended | suspending>
锁定/解锁虚拟机。

--machine [[type=]<machine type>] [,enable-s3=<1|0>] [,enable-s4=<1|0>] [,viommu=<intel|virtio>]
指定 QEMU machine。

--memory [current=]<integer>
内存属性。

--migrate_downtime <number> (0 - N) (*default* = 0.1)
设置迁移可容忍的最大停机时间（秒）。如果迁移在最后阶段因需要传输太多新脏 RAM 而无法收

--migrate_speed <integer> (0 - N) (*default* = 0)
设置迁移最大速度（MB/s）。值为 0 表示无限制。

--name <string>
设置虚拟机名称。仅用于配置 Web 界面。

--nameserver <string>
cloud-init：设置容器的 DNS 服务器 IP 地址。如果既未设置 searchdomain 也未设置 nameserver，创建时会自动使用主机上的设置。

--net[n] [model=]<enum> [,bridge=<bridge>] [,firewall=<1|0>] [,link_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX:XX>] [,mtu=<integer>] [,queues=<integer>] [,rate=<number>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,<model>=<macaddr>]
指定网络设备。

--numa <boolean> (*default* = 0)
启用/禁用 NUMA。

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

--onboot <boolean> (*default* = 0)
指定虚拟机是否在系统启动期间启动。

--ostype <l24 | l26 | other | solaris | w2k | w2k3 | w2k8 | win10 | win11 | win7 | win8 | wvista | wxp>
指定客户机操作系统。

--parallel[n] /dev/parport\d+|/dev/usb/lp\d+
映射主机并口设备（n 为 0 到 2）。

--pool <string>
将虚拟机添加到指定资源池。

--protection <boolean> (*default* = 0)
设置虚拟机的保护标志。这会禁用删除虚拟机和删除磁盘操作。

--reboot <boolean> (*default* = 1)
允许重启。如果设置为 0，虚拟机会在重启时退出。

--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。

--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>] [,import-from=<source volume>] [,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>]

将卷用作 SATA 硬盘或 CD-ROM（n 为 0 到 5）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。使用 STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

--nvme[n] [file=]<volume> [,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>] [,format=<enum>]
[,heads=<integer>] [,import-from=<source volume>] [,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>] [,size=<DiskSize>] [,snapshot=<1|0>]

将卷用作 NVME 硬盘或 CD-ROM（n 为 0 到 5）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。使用 STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

--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>] [,import-from=<source volume>] [,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>] [,product=<product>] [,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>] [,vendor=<vendor>] [,werror=<enum>] [,wwn=<wwn>]

将卷用作 SCSI 硬盘或 CD-ROM（n 为 0 到 30）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。使用 STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

--scsihw <lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci | virtio-scsi-single> (*default* = lsi)

SCSI 控制器型号。

--searchdomain <string>

cloud-init：设置容器的 DNS 搜索域。如果既未设置 searchdomain 也未设置 nameserver，创

--serial[n] (/dev/.+|socket)

在虚拟机内部创建串口设备（n 为 0 到 3）。

--shares <integer> (0 - 50000) (*default* = 1000)

自动 ballooning 的内存 share 数量。数值越大，该虚拟机获得的内存越多。该数值相对于所有其 0 会禁用自动 ballooning。自动 ballooning 由 pvestatd 执行。

--smbios1 [base64=<1|0>] [,family=<Base64 encoded string>] [,manufacturer=<Base64 encoded string>] [,product=<Base64 encoded string>] [,serial=<Base64 encoded string>] [,sku=<Base64 encoded string>] [,uuid=<UUID>] [,version=<Base64 encoded string>]

指定 SMBIOS type 1 字段。

--smp <integer> (1 - N) (*default* = 1)

CPU 数量。请改用选项 -sockets。

--sockets <integer> (1 - N) (*default* = 1)

CPU socket 数量。

--spice_enhancements [foldersharing=<1|0>] [,videostreaming=<off|all|filter>]

配置 SPICE 的其他增强功能。

--sshkeys <filepath>
cloud-init：设置公共 SSH 密钥（每行一个密钥，OpenSSH 格式）。

--start <boolean> (*default* = 0)
虚拟机成功创建后启动。

--startdate (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (*default* = now)
设置实时时钟的初始日期。有效日期格式为：now、2006-06-17T16:01:21 或 2006-06-17。

--startup `[[order=]\d+] [,up=\d+] [,down=\d+]`
启动和关机行为。Order 是定义通用启动顺序的非负数。关机按相反顺序完成。此外，可以设置 up 或 down 延迟（秒），用于指定启动或停止下一台虚拟机之前等待的延迟。

--storage <storage ID>
默认存储。

--tablet <boolean> (*default* = 1)
启用/禁用 USB tablet 设备。

--tags <string>
虚拟机标签。该信息仅为元信息。

--tdf <boolean> (*default* = 0)
启用/禁用时间漂移修复。

--template <boolean> (*default* = 0)
启用/禁用 Template。

--tpmstate0 [file=]<volume> [,import-from=<source volume>] [,size=<DiskSize>] [,version=<v1.2|v2.0>]
配置用于存储 TPM 状态的磁盘。格式固定为 raw。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。注意，此处会忽略 SIZE_IN_GiB，并改用 4 MiB。使用 STORAGE_ID:0 和 import-f 参数可从现有卷导入。

--unique <boolean>
分配唯一的随机以太网地址。

> **Note**
> 需要选项：archive

--unused[n] [file=]<volume>
对未使用卷的引用。该项供内部使用，不应手动修改。

--usb[n] [[host=]<HOSTUSBDEVICE|spice>] [,mapping=<mapping-id>] [,usb3=<1|0>]
配置 USB 设备（n 为 0 到 4；对于 machine version >= 7.1 且 ostype 为 l26 或 windows > 7 的客户机，n 可达 14）。

--vcpus <integer> (1 - N) (*default* = 0)
热插拔 vcpu 数量。

--vga [[type=]<enum>] [,clipboard=<vnc>] [,memory=<integer>]
配置 VGA 硬件。

**--virtio[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>] [,import-from=<source volume>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>]
[,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>] [,media=<cdrom|disk>]
[,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,ro=<1|0>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,trans=<none|lba|auto>]
[,werror=<enum>]**
将卷用作 VIRTIO 硬盘（n 为 0 到 15）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。
STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

**--virtiofs[n] [dirid=]<mapping-id> [,cache=<enum>]
[,direct-io=<1|0>] [,expose-acl=<1|0>] [,expose-xattr=<1|0>]**
使用 Virtio-fs 在主机和客户机之间共享目录的配置。

--vmgenid <UUID> (*default* = 1 (autogenerated))
设置 VM Generation ID。使用 1 可在创建或更新时自动生成，传入 0 可显式禁用。

--vmstatestorage <storage ID>
VM 状态卷/文件的默认存储。

--watchdog [[model=]<i6300esb|ib700>] [,action=<enum>]
创建虚拟硬件 watchdog 设备。

qm delsnapshot <vmid> <snapname> [OPTIONS]

删除虚拟机快照。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<snapname>: <string>
快照名称。

--force <boolean>
即使删除磁盘快照失败，也从配置文件中移除。

qm destroy <vmid> [OPTIONS]

销毁虚拟机以及所有已使用/归属的卷。移除所有虚拟机专用权限和防火墙规则。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--destroy-unreferenced-disks <boolean> (*default* = 0)
如果设置，还会从所有已启用存储中销毁未被配置引用但 VMID 匹配的所有磁盘。

--purge <boolean>
从备份、复制作业和 HA 等配置中移除 VMID。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

qm disk import <vmid> <source> <storage> [OPTIONS]

将外部磁盘镜像作为虚拟机中的未使用磁盘导入。镜像格式必须受 qemu-img(1) 支持。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<source>: <string>
要导入的磁盘镜像路径。

<storage>: <storage ID>
目标存储 ID。

--format <qcow2 | raw | vmdk>
目标格式。

--target-disk <efidisk0 | ide0 | ide1 | ide2 | ide3 | nvme0 |nvme1 |nvme2 |nvme3 |nvme4 |nvme5 |sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | scsi16 | scsi17 | scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25 | scsi26 | scsi27 | scsi28 | scsi29 | scsi3 | scsi30 | scsi4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | unused0 | unused1 | unused10 | unused100 | unused101 | unused102 | unused103 | unused104 | unused105 | unused106 | unused107 | unused108 | unused109 | unused11 | unused110 | unused111 | unused112 | unused113 | unused114 | unused115 | unused116 | unused117 | unused118 | unused119 | unused12 | unused120 | unused121 | unused122 | unused123 | unused124 | unused125 | unused126 | unused127 | unused128 | unused129 | unused13 | unused130 | unused131 | unused132 | unused133 | unused134 | unused135 | unused136 | unused137 | unused138 | unused139 | unused14 | unused140 | unused141 | unused142 | unused143 | unused144 | unused145 | unused146 | unused147 | unused148 | unused149 | unused15 | unused150 | unused151 | unused152 | unused153 | unused154 | unused155 | unused156 | unused157 | unused158 | unused159 | unused16 | unused160 | unused161 | unused162 | unused163 | unused164 | unused165 | unused166 | unused167 | unused168 | unused169 | unused17 | unused170 | unused171 | unused172 | unused173 | unused174 | unused175 | unused176 | unused177 | unused178 | unused179 | unused18 | unused180 | unused181 | unused182 | unused183 | unused184 | unused185 | unused186 | unused187 | unused188 | unused189 | unused19 | unused190 | unused191 | unused192 | unused193 | unused194 | unused195 | unused196 | unused197 | unused198 | unused199 | unused2 | unused20 | unused200 | unused201 | unused202 | unused203 | unused204 | unused205 | unused206 | unused207 | unused208 | unused209 | unused21 | unused210 | unused211 | unused212 | unused213 | unused214 | unused215 | unused216 | unused217 | unused218 | unused219 | unused22 | unused220 | unused221 | unused222 | unused223 | unused224 | unused225 | unused226 | unused227 | unused228 | unused229 | unused23 | unused230 | unused231 | unused232 | unused233 | unused234 | unused235 | unused236 | unused237 | unused238 | unused239 | unused24 | unused240 | unused241 | unused242 | unused243 | unused244 | unused245 | unused246 | unused247 | unused248 | unused249 | unused25 | unused250 | unused251 | unused252 | unused253 | unused254 | unused255 | unused26 | unused27 | unused28 | unused29 | unused3 | unused30 | unused31 | unused32 | unused33 | unused34 | unused35 | unused36 | unused37 | unused38 | unused39 | unused4 | unused40 | unused41 | unused42 | unused43 | unused44 | unused45 | unused46 | unused47 | unused48 | unused49 | unused5 | unused50 | unused51 | unused52 | unused53 | unused54 | unused55 | unused56 | unused57 | unused58 | unused59 | unused6 | unused60 | unused61 | unused62 | unused63 | unused64 | unused65 | unused66 | unused67 | unused68 | unused69 | unused7 | unused70 | unused71 | unused72 | unused73 | unused74 | unused75 | unused76 | unused77 | unused78 | unused79 | unused8 | unused80 | unused81 | unused82 | unused83 | unused84 | unused85 | unused86 | unused87 | unused88 | unused89 | unused9 |

卷将导入到的磁盘名称（例如 scsi1）。

qm disk move <vmid> <disk> [<storage>] [OPTIONS]

将卷移动到其他存储或其他虚拟机。

<vmid>: <integer> (100 - 999999999)

虚拟机的（唯一）ID。

<disk>: <efidisk0 | ide0 | ide1 | ide2 | ide3 | nvme0 |nvme1 |nvme2 |nvme3 |nvme4 |nvme5 |sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | scsi16 | scsi17 | scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25 | scsi26 | scsi27 | scsi28 | scsi29 | scsi3 | scsi30 | scsi4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | unused0 | unused1 | unused10 | unused100 | unused101 | unused102 | unused103 | unused104 | unused105 | unused106 | unused107 | unused108 | unused109 | unused11 | unused110 | unused111 | unused112 | unused113 | unused114 | unused115 | unused116 | unused117 | unused118 | unused119 | unused12 | unused120 | unused121 | unused122 | unused123 | unused124 | unused125 | unused126 | unused127 | unused128 | unused129 | unused13 | unused130 | unused131 | unused132 | unused133 | unused134 | unused135 | unused136 | unused137 | unused138 | unused139 | unused14 | unused140 | unused141 | unused142 | unused143 | unused144 | unused145 | unused146 | unused147 | unused148 | unused149 | unused15 | unused150 | unused151 | unused152 | unused153 | unused154 | unused155 | unused156 | unused157 | unused158 | unused159 | unused16 | unused160 | unused161 | unused162 | unused163 | unused164 | unused165 | unused166 | unused167 | unused168 | unused169 | unused17 | unused170 | unused171 | unused172 | unused173 | unused174 | unused175 | unused176 | unused177 | unused178 | unused179 | unused18 | unused180 | unused181 | unused182 | unused183 | unused184 | unused185 | unused186 | unused187 | unused188 | unused189 | unused19 | unused190 | unused191 | unused192 | unused193 | unused194 | unused195 | unused196 | unused197 | unused198 | unused199 | unused2 | unused20 | unused200 | unused201 | unused202 | unused203 | unused204 | unused205 | unused206 | unused207 | unused208 | unused209 | unused21 | unused210 | unused211 | unused212 | unused213 | unused214 | unused215 | unused216 | unused217 | unused218 | unused219 | unused22 | unused220 | unused221 | unused222 | unused223 | unused224 | unused225 | unused226 | unused227 | unused228 | unused229 | unused23 | unused230 | unused231 | unused232 | unused233 | unused234 | unused235 | unused236 | unused237 | unused238 | unused239 | unused24 | unused240 | unused241 | unused242 | unused243 | unused244 | unused245 | unused246 | unused247 | unused248 | unused249 | unused25 | unused250 | unused251 | unused252 | unused253 | unused254 | unused255 | unused26 | unused27 | unused28 | unused29 | unused3 | unused30 | unused31 | unused32 | unused33 | unused34 | unused35 | unused36 | unused37 | unused38 | unused39 | unused4 | unused40 | unused41 | unused42 | unused43 | unused44 | unused45 | unused46 | unused47 | unused48 | unused49 | unused5 | unused50 | unused51 | unused52 | unused53 | unused54 | unused55 | unused56 | unused57 | unused58 | unused59 | unused6 | unused60 | unused61 | unused62 | unused63 | unused64 | unused65 | unused66 | unused67 | unused68 | unused69 | unused7 | unused70 | unused71 | unused72 | unused73 | unused74 | unused75 | unused76 | unused77 | unused78 | unused79 | unused8 | unused80 | unused81 | unused82 | unused83 | unused84 | unused85 | unused86 | unused87 | unused88 | unused89 | unused9 |

要移动的磁盘。

<storage>: <storage ID>
目标存储。

--bwlimit <integer> (0 - N) (*default* = move limit from datacenter or storage config)
覆盖 I/O 带宽限制（单位 KiB/s）。

--delete <boolean> (*default* = 0)
复制成功后删除原始磁盘。默认情况下，原始磁盘会保留为未使用磁盘。

--digest <string>
如果当前配置文件具有不同的 SHA1 digest，则阻止更改。这可用于防止并发修改。

--format <qcow2 | raw | vmdk>
目标格式。

--target-digest <string>
如果目标虚拟机的当前配置文件具有不同的 SHA1 digest，则阻止更改。这可用于检测并发修改。

--target-disk <efidisk0 | ide0 | ide1 | ide2 | ide3 | nvme0 |nvme1 |nvme2 |nvme3 |nvme4 |nvme5 |sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | scsi16 | scsi17 | scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25 | scsi26 | scsi27 | scsi28 | scsi29 | scsi3 | scsi30 | scsi4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | unused0 | unused1 | unused10 | unused100 | unused101 | unused102 | unused103 | unused104 | unused105 | unused106 | unused107 | unused108 | unused109 | unused11 | unused110 | unused111 | unused112 | unused113 | unused114 | unused115 | unused116 | unused117 | unused118 | unused119 | unused12 | unused120 | unused121 | unused122 | unused123 | unused124 | unused125 | unused126 | unused127 | unused128 | unused129 | unused13 | unused130 | unused131 | unused132 | unused133 | unused134 | unused135 | unused136 | unused137 | unused138 | unused139 | unused14 | unused140 | unused141 | unused142 | unused143 | unused144 | unused145 | unused146 | unused147 | unused148 | unused149 | unused15 | unused150 | unused151 | unused152 | unused153 | unused154 | unused155 | unused156 | unused157 | unused158 | unused159 | unused16 | unused160 | unused161 | unused162 | unused163 | unused164 | unused165 | unused166 | unused167 | unused168 | unused169 | unused17 | unused170 | unused171 | unused172 | unused173 | unused174 | unused175 | unused176 | unused177 | unused178 | unused179 | unused18 | unused180 | unused181 | unused182 | unused183 | unused184 | unused185 | unused186 | unused187 | unused188 | unused189 | unused19 | unused190 | unused191 | unused192 | unused193 | unused194 | unused195 | unused196 | unused197 | unused198 | unused199 | unused2 | unused20 | unused200 | unused201 | unused202 | unused203 | unused204 | unused205 | unused206 | unused207 | unused208 | unused209 | unused21 | unused210 | unused211 | unused212 | unused213 | unused214 | unused215 | unused216 | unused217 | unused218 | unused219 | unused22 | unused220 | unused221 | unused222 | unused223 | unused224 | unused225 | unused226 | unused227 | unused228 | unused229 | unused23 | unused230 | unused231 | unused232 | unused233 | unused234 | unused235 | unused236 | unused237 | unused238 | unused239 | unused24 | unused240 | unused241 | unused242 | unused243 | unused244 | unused245 | unused246 | unused247 | unused248 | unused249 | unused25 | unused250 | unused251 | unused252 | unused253 | unused254 | unused255 | unused26 | unused27 | unused28 | unused29 | unused3 | unused30 | unused31 | unused32 | unused33 | unused34 | unused35 | unused36 | unused37 | unused38 | unused39 | unused4 | unused40 | unused41 | unused42 | unused43 | unused44 | unused45 | unused46 | unused47 | unused48 | unused49 | unused5 | unused50 | unused51 | unused52 | unused53 | unused54 | unused55 | unused56 | unused57 | unused58 | unused59 | unused6 | unused60 | unused61 | unused62 | unused63 | unused64 | unused65 | unused66 | unused67 | unused68 | unused69 | unused7 | unused70 | unused71 | unused72 | unused73 | unused74 | unused75 | unused76 | unused77 | unused78 | unused79 | unused8 | unused80 | unused81 | unused82 | unused83 | unused84 | unused85 | unused86 | unused87 | unused88 | unused89 | unused9 |

磁盘将移动到目标虚拟机上的配置键（例如 ide0 或 scsi1）。默认为源磁盘键。

--target-vmid <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm disk clone <vmid> [OPTIONS]

将链接克隆卷移动到其他存储或其他虚拟机。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<disk>: <efidisk0 | ide0 | ide1 | ide2 | ide3 | nvme0 |nvme1 |nvme2 |nvme3 |nvme4 |nvme5 |sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | scsi16 | scsi17 | scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25 | scsi26 | scsi27 | scsi28 | scsi29 | scsi3 | scsi30 | scsi4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | unused0 | unused1 | unused10 | unused100 | unused101 | unused102 | unused103 | unused104 | unused105 | unused106 | unused107 | unused108 | unused109 | unused11 | unused110 | unused111 | unused112 | unused113 | unused114 | unused115 | unused116 | unused117 | unused118 | unused119 | unused12 | unused120 | unused121 | unused122 | unused123 | unused124 | unused125 | unused126 | unused127 | unused128 | unused129 | unused13 | unused130 | unused131 | unused132 | unused133 | unused134 | unused135 | unused136 | unused137 | unused138 | unused139 | unused14 | unused140 | unused141 | unused142 | unused143 | unused144 | unused145 | unused146 | unused147 | unused148 | unused149 | unused15 | unused150 | unused151 | unused152 | unused153 | unused154 | unused155 | unused156 | unused157 | unused158 | unused159 | unused16 | unused160 | unused161 | unused162 | unused163 | unused164 | unused165 | unused166 | unused167 | unused168 | unused169 | unused17 | unused170 | unused171 | unused172 | unused173 | unused174 | unused175 | unused176 | unused177 | unused178 | unused179 | unused18 | unused180 | unused181 | unused182 | unused183 | unused184 | unused185 | unused186 | unused187 | unused188 | unused189 | unused19 | unused190 | unused191 | unused192 | unused193 | unused194 | unused195 | unused196 | unused197 | unused198 | unused199 | unused2 | unused20 | unused200 | unused201 | unused202 | unused203 | unused204 | unused205 | unused206 | unused207 | unused208 | unused209 | unused21 | unused210 | unused211 | unused212 | unused213 | unused214 | unused215 | unused216 | unused217 | unused218 | unused219 | unused22 | unused220 | unused221 | unused222 | unused223 | unused224 | unused225 | unused226 | unused227 | unused228 | unused229 | unused23 | unused230 | unused231 | unused232 | unused233 | unused234 | unused235 | unused236 | unused237 | unused238 | unused239 | unused24 | unused240 | unused241 | unused242 | unused243 | unused244 | unused245 | unused246 | unused247 | unused248 | unused249 | unused25 | unused250 | unused251 | unused252 | unused253 | unused254 | unused255 | unused26 | unused27 | unused28 | unused29 | unused3 | unused30 | unused31 | unused32 | unused33 | unused34 | unused35 | unused36 | unused37 | unused38 | unused39 | unused4 | unused40 | unused41 | unused42 | unused43 | unused44 | unused45 | unused46 | unused47 | unused48 | unused49 | unused5 | unused50 | unused51 | unused52 | unused53 | unused54 | unused55 | unused56 | unused57 | unused58 | unused59 | unused6 | unused60 | unused61 | unused62 | unused63 | unused64 | unused65 | unused66 | unused67 | unused68 | unused69 | unused7 | unused70 | unused71 | unused72 | unused73 | unused74 | unused75 | unused76 | unused77 | unused78 | unused79 | unused8 | unused80 | unused81 | unused82 | unused83 | unused84 | unused85 | unused86 | unused87 | unused88 | unused89 | unused9 |

要移动的磁盘。

--delete <boolean> (*default* = 1)

复制成功后删除原始磁盘。默认会删除原始磁盘。

--target-disk <efidisk0 | ide0 | ide1 | ide2 | ide3 | nvme0 |nvme1 |nvme2 |nvme3 |nvme4 |nvme5 |sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | scsi16 | scsi17 | scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25 | scsi26 | scsi27 | scsi28 | scsi29 | scsi3 | scsi30 | scsi4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | unused0 | unused1 | unused10 | unused100 | unused101 | unused102 | unused103 | unused104 | unused105 | unused106 | unused107 | unused108 | unused109 | unused11 | unused110 | unused111 | unused112 | unused113 | unused114 | unused115 | unused116 | unused117 | unused118 | unused119 | unused12 | unused120 | unused121 | unused122 | unused123 | unused124 | unused125 | unused126 | unused127 | unused128 | unused129 | unused13 | unused130 | unused131 | unused132 | unused133 | unused134 | unused135 | unused136 | unused137 | unused138 | unused139 | unused14 | unused140 | unused141 | unused142 | unused143 | unused144 | unused145 | unused146 | unused147 | unused148 | unused149 | unused15 | unused150 | unused151 | unused152 | unused153 | unused154 | unused155 | unused156 | unused157 | unused158 | unused159 | unused16 | unused160 | unused161 | unused162 | unused163 | unused164 | unused165 | unused166 | unused167 | unused168 | unused169 | unused17 | unused170 | unused171 | unused172 | unused173 | unused174 | unused175 | unused176 | unused177 | unused178 | unused179 | unused18 | unused180 | unused181 | unused182 | unused183 | unused184 | unused185 | unused186 | unused187 | unused188 | unused189 | unused19 | unused190 | unused191 | unused192 | unused193 | unused194 | unused195 | unused196 | unused197 | unused198 | unused199 | unused2 | unused20 | unused200 | unused201 | unused202 | unused203 | unused204 | unused205 | unused206 | unused207 | unused208 | unused209 | unused21 | unused210 | unused211 | unused212 | unused213 | unused214 | unused215 | unused216 | unused217 | unused218 | unused219 | unused22 | unused220 | unused221 | unused222 | unused223 | unused224 | unused225 | unused226 | unused227 | unused228 | unused229 | unused23 | unused230 | unused231 | unused232 | unused233 | unused234 | unused235 | unused236 | unused237 | unused238 | unused239 | unused24 | unused240 | unused241 | unused242 | unused243 | unused244 | unused245 | unused246 | unused247 | unused248 | unused249 | unused25 | unused250 | unused251 | unused252 | unused253 | unused254 | unused255 | unused26 | unused27 | unused28 | unused29 | unused3 | unused30 | unused31 | unused32 | unused33 | unused34 | unused35 | unused36 | unused37 | unused38 | unused39 | unused4 | unused40 | unused41 | unused42 | unused43 | unused44 | unused45 | unused46 | unused47 | unused48 | unused49 | unused5 | unused50 | unused51 | unused52 | unused53 | unused54 | unused55 | unused56 | unused57 | unused58 | unused59 | unused6 | unused60 | unused61 | unused62 | unused63 | unused64 | unused65 | unused66 | unused67 | unused68 | unused69 | unused7 | unused70 | unused71 | unused72 | unused73 | unused74 | unused75 | unused76 | unused77 | unused78 | unused79 | unused8 | unused80 | unused81 | unused82 | unused83 | unused84 | unused85 | unused86 | unused87 | unused88 | unused89 | unused9 |

磁盘将移动到目标虚拟机上的配置键（例如 ide0 或 scsi1）。默认为源磁盘键。

--target-vmid <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--snapname <string>
虚拟机快照名称。

qm disk rescan [OPTIONS]

重新扫描所有存储，并更新磁盘大小和未使用磁盘镜像。

--dryrun <boolean> (*default* = 0)
不实际将更改写入虚拟机配置。

--vmid <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm disk resize <vmid> <disk> <size> [OPTIONS]

扩展卷大小。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<disk>: <efidisk0 | ide0 | ide1 | ide2 | ide3 | nvme0 |nvme1 |nvme2 |nvme3 |nvme4 |nvme5 |sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | scsi16 | scsi17 | scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25 | scsi26 | scsi27 | scsi28 | scsi29 | scsi3 | scsi30 | scsi4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | virtio0 | virtio1 | virtio10 | virtio11 | virtio12 | virtio13 | virtio14 | virtio15 | virtio2 | virtio3 | virtio4 | virtio5 | virtio6 | virtio7 | virtio8 | virtio9>
要调整大小的磁盘。

<size>: \+?\d+(\.\d+)?[KMGT]?
新的大小。带 + 号时，该值会加到卷的实际大小上；不带 + 号时，该值会作为绝对大小使用。不支

--digest <string>
如果当前配置文件具有不同的 SHA1 digest，则阻止更改。这可用于防止并发修改。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

qm disk unlink <vmid> --idlist <string> [OPTIONS]

取消链接/删除磁盘镜像。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--force <boolean>
强制物理移除。未设置时，仅从配置文件中移除磁盘，并创建名为 unused[n] 的附加配置条目，ID。取消链接 unused[n] 始终会导致物理移除。

--idlist <string>
要删除的磁盘 ID 列表。

qm guest cmd <vmid> <command>

执行 QEMU Guest Agent 命令。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<command>: <fsfreeze-freeze | fsfreeze-status | fsfreeze-thaw | fstrim | get-fsinfo | get-host-name | get-memory-block-info | get-memory-blocks | get-osinfo | get-time | get-timezone | get-users | get-vcpus | info | network-get-interfaces | ping | shutdown | suspend-disk | suspend-hybrid | suspend-ram>
QGA 命令。

qm guest exec <vmid> [<extra-args>] [OPTIONS]

通过 guest agent 执行给定命令。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<extra-args>: <array>
作为数组传入的额外参数。

--pass-stdin <boolean> (*default* = 0)
设置后，会读取 STDIN 直到 EOF，并通过 input-data 转发给 guest agent（通常被 guest agent 启动的进程视为 STDIN）。最大允许 1 MiB。

--synchronous <boolean> (*default* = 1)
如果设置为 off，则立即返回 pid，而不是等待命令完成或超时。

--timeout <integer> (0 - N) (*default* = 30)
同步等待命令完成的最长时间。如果达到该时间，则返回 pid。设置为 0 可停用。

qm guest exec-status <vmid> <pid>

获取由 guest-agent 启动的给定 pid 的状态。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<pid>: <integer>
要查询的 PID。

qm guest passwd <vmid> <username> [OPTIONS]

将给定用户的密码设置为给定密码。

<vmid>: <integer> (100 - 999999999)

虚拟机的（唯一）ID。

<username>: <string>

要设置密码的用户。

--crypted <boolean> (*default* = 0)

如果密码已经通过 crypt() 处理，则设置为 1。

qm help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>

显示特定命令的帮助。

--verbose <boolean>

详细输出格式。

qm import <vmid> <source> --storage <string> [OPTIONS]

从受支持的导入源（例如 ESXi 存储）导入外部虚拟客户机。

<vmid>: <integer> (100 - 999999999)

虚拟机的（唯一）ID。

gicversion: <2 | 3 | 4 | max | host > (*default* = host)

设置 ARM 虚拟机的 gicversion。

virtualization: <boolean> (*default* = 0)

为 arm64 虚拟机启用嵌套虚拟化。

snapshot=<boolean>

控制 qemu 的快照模式功能。启用后，对磁盘所做的更改是临时的，并会在虚拟机关机时丢

pxvditemplate <boolean> (*default* = 0)

启用/禁用 PXVDI Template。

<source>: <string>

导入源卷 ID。

--acpi <boolean> (*default* = 1)

启用/禁用 ACPI。

--affinity <string>

用于执行客户机进程的主机核心列表，例如：0,5,8-11

--agent [enabled=]<1|0> [,freeze-fs-on-backup=<1|0>] [,fstrim_cloned_disks=<1|0>] [,type=<virtio|isa>]

启用/禁用与 QEMU Guest Agent 及其属性的通信。

--amd-sev [type=]<sev-type> [,allow-smt=<1|0>] [,kernel-hashes=<1|0>] [,no-debug=<1|0>] [,no-key-sharing=<1|0>]
AMD CPU 提供的 Secure Encrypted Virtualization（SEV）功能。

--arch <aarch64 | x86_64 | loongarch64 | riscv64>
虚拟处理器架构。默认为主机架构。

--args <string>
传递给 kvm 的任意参数。

--audio0 device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>]
配置音频设备，与 QXL/Spice 结合使用时很有用。

--autostart <boolean> (*default* = 0)
崩溃后自动重启（当前会被忽略）。

--balloon <integer> (0 - N)
虚拟机目标 RAM 容量，单位 MiB。使用 0 会禁用 balloon 驱动。

--bios <ovmf | seabios> (*default* = seabios)
选择 BIOS 实现。

--boot [[legacy=]<[acdn]{1,4}>] [,order=<device[;device...]>]
指定客户机启动顺序。请使用 `order=` 子属性；不带键或使用 `legacy=` 的用法已弃用。

--bootdisk (ide|sata|scsi|virtio)\d+
启用从指定磁盘启动。已弃用：请改用 `boot: order=foo;bar`。

--cdrom <volume>
这是选项 -ide2 的别名。

--cicustom [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>]
cloud-init：指定自定义文件，以替换启动时自动生成的文件。

--cipassword <string>
cloud-init：要分配给用户的密码。通常不建议使用此项，请改用 ssh 密钥。另请注意，较旧版本 cloud-init 不支持哈希密码。

--citype <configdrive2 | nocloud | opennebula>
指定 cloud-init 配置格式。默认值取决于配置的操作系统类型（`ostype`）。Linux 使用 `nocloud` 格式，windows 使用 `configdrive2`。

--ciupgrade <boolean> (*default* = 1)
cloud-init：首次启动后自动执行软件包升级。

--ciuser <string>
cloud-init：要更改 ssh 密钥和密码的用户名，用于替代镜像中配置的默认用户。

--cores <integer> (1 - N) (*default* = 1)
每个 socket 的核心数。

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

--cpulimit <number> (0 - 128) (*default* = 0)
CPU 使用限制。

--cpuunits <integer> (1 - 262144) (*default* = cgroup v1: 1024, cgroup v2: 100)
虚拟机的 CPU 权重，在 cgroup v2 中会限制到 [1, 10000] 范围内。

--delete <string>
要删除的设置列表。

--description <string>
虚拟机描述。显示在 Web 界面的虚拟机摘要中，并作为注释保存到配置文件内。

--dryrun <boolean> (*default* = 0)
显示创建命令并退出，不执行任何操作。

--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。

--format <qcow2 | raw | vmdk>
目标格式。

--freeze <boolean>
启动时冻结 CPU（使用 c monitor 命令开始执行）。

--hookscript <string>
将在虚拟机生命周期的各个步骤中执行的脚本。

--hostpci[n] [[host=]<HOSTPCIID[;HOSTPCIID2...]>] [,device-id=<hex id>] [,legacy-igd=<1|0>] [,mapping=<mapping-id>] [,mdev=<string>] [,pcie=<1|0>] [,rombar=<1|0>] [,romfile=<string>] [,sub-device-id=<hex id>] [,sub-vendor-id=<hex id>] [,vendor-id=<hex id>] [,x-vga=<1|0>] [,ramfb=<1|0>]
将主机 PCI 设备映射到客户机。

--hotplug <string> (*default* = network,disk,usb)
选择性启用热插拔功能。这是一个逗号分隔的热插拔功能列表：network、disk、cpu、memory 和 cloudinit。使用 0 可完全禁用热插拔。使用 1 作为值是默认值 network,disk,usb 的别名。对于 machine version >= 7.1 且 ostype 为 l26 或 windows > 7 的客户机，可以使用 USB 热插拔。

--hugepages <1024 | 2 | any>
启用/禁用 hugepages 内存。

**--ide[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>] [,model=<model>]
[,replicate=<1|0>] [,rerror=<ignore|report|stop>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>]**

将卷用作 IDE 硬盘或 CD-ROM（n 为 0 到 3）。

**--ipconfig[n] [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>]
[,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>]**

cloud-init：指定对应接口的 IP 地址和网关。

IP 地址使用 CIDR 表示法；网关是可选项，但需要指定同类型 IP。

IP 地址可以使用特殊字符串 dhcp 以使用 DHCP，此时不应显式提供网关。 对于 IPv6，可以使用 auto 以使用无状态自动配置。这需要 cloud-init 19.4 或更新版本。

如果启用了 cloud-init 且既未指定 IPv4 地址也未指定 IPv6 地址，则默认在 IPv4 上使用 dhcp。

--ivshmem size=<integer> [,name=<string>]

虚拟机间共享内存。适用于虚拟机之间或虚拟机与主机之间的直接通信。

--keephugepages <boolean> (*default* = 0)

与 hugepages 一起使用。启用后，虚拟机关机后 hugepages 不会被删除，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>**

VNC 服务器的键盘布局。通常不需要此选项，且往往更适合在客户机 OS 内处理。

--kvm <boolean> (*default* = 1)

启用/禁用 KVM 硬件虚拟化。

--live-import <boolean> (*default* = 0)

立即启动虚拟机，并在后台复制数据。

--localtime <boolean>

将实时时钟（RTC）设置为本地时间。如果 ostype 表示 Microsoft Windows OS，则默认启用。

**--lock <backup | clone | create | migrate | rollback | snapshot |
snapshot-delete | suspended | suspending>**

锁定/解锁虚拟机。

--machine [[type=]<machine type>] [,enable-s3=<1|0>] [,enable-s4=<1|0>] [,viommu=<intel|virtio>]
指定 QEMU machine。

--memory [current=]<integer>
内存属性。

--migrate_downtime <number> (0 - N) (*default* = 0.1)
设置迁移可容忍的最大停机时间（秒）。如果迁移在最后阶段因需要传输太多新脏 RAM 而无法收

--migrate_speed <integer> (0 - N) (*default* = 0)
设置迁移最大速度（MB/s）。值为 0 表示无限制。

--name <string>
设置虚拟机名称。仅用于配置 Web 界面。

--nameserver <string>
cloud-init：设置容器的 DNS 服务器 IP 地址。如果既未设置 searchdomain 也未设置 nameserver，创建时会自动使用主机上的设置。

--net[n] [model=]<enum> [,bridge=<bridge>] [,firewall=<1|0>] [,link_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX:XX>] [,mtu=<integer>] [,queues=<integer>] [,rate=<number>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,<model>=<macaddr>]
指定网络设备。

--numa <boolean> (*default* = 0)
启用/禁用 NUMA。

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

--onboot <boolean> (*default* = 0)
指定虚拟机是否在系统启动期间启动。

--ostype <l24 | l26 | other | solaris | w2k | w2k3 | w2k8 | win10 | win11 | win7 | win8 | wvista | wxp>
指定客户机操作系统。

--parallel[n] /dev/parport\d+|/dev/usb/lp\d+
映射主机并口设备（n 为 0 到 2）。

--protection <boolean> (*default* = 0)
设置虚拟机的保护标志。这会禁用删除虚拟机和删除磁盘操作。

--reboot <boolean> (*default* = 1)
允许重启。如果设置为 0，虚拟机会在重启时退出。

--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。

**--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>]**

将卷用作 SATA 硬盘或 CD-ROM（n 为 0 到 5）。

**--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>]
[,product=<product>] [,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>] [,vendor=<vendor>] [,werror=<enum>]
[,wwn=<wwn>]**

将卷用作 SCSI 硬盘或 CD-ROM（n 为 0 到 30）。

**--scsihw <lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci |
virtio-scsi-single> (*default* = lsi)**

SCSI 控制器型号。

--searchdomain <string>

cloud-init：设置容器的 DNS 搜索域。如果既未设置 searchdomain 也未设置 nameserver，创

--serial[n] (/dev/.+|socket)

在虚拟机内部创建串口设备（n 为 0 到 3）。

--shares <integer> (0 - 50000) (*default* = 1000)

自动 ballooning 的内存 share 数量。数值越大，该虚拟机获得的内存越多。该数值相对于所有其
0 会禁用自动 ballooning。自动 ballooning 由 pvestatd 执行。

--smbios1 [base64=<1|0>] [,family=<Base64 encoded string>] [,manufacturer=<Base64 encoded string>] [,product=<Base64 encoded string>] [,serial=<Base64 encoded string>] [,sku=<Base64 encoded string>] [,uuid=<UUID>] [,version=<Base64 encoded string>]
指定 SMBIOS type 1 字段。

--smp <integer> (1 - N) (*default* = 1)
CPU 数量。请改用选项 -sockets。

--UUID <UUID> (*default* = 1 (autogenerated))
VM uuid。

--sockets <integer> (1 - N) (*default* = 1)
CPU socket 数量。

--spice_enhancements [foldersharing=<1|0>] [,videostreaming=<off|all|filter>]
配置 SPICE 的其他增强功能。

--sshkeys <string>
cloud-init：设置公共 SSH 密钥（每行一个密钥，OpenSSH 格式）。

--startdate (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (*default* = now)
设置实时时钟的初始日期。有效日期格式为：now、2006-06-17T16:01:21 或 2006-06-17。

--startup `[[order=]\d+] [,up=\d+] [,down=\d+]`
启动和关机行为。Order 是定义通用启动顺序的非负数。关机按相反顺序完成。此外，可以设置 up 或 down 延迟（秒），用于指定启动或停止下一台虚拟机之前等待的延迟。

--storage <storage ID>
默认存储。

--tablet <boolean> (*default* = 1)
启用/禁用 USB tablet 设备。

--tags <string>
虚拟机标签。该信息仅为元信息。

--tdf <boolean> (*default* = 0)
启用/禁用时间漂移修复。

--template <boolean> (*default* = 0)
启用/禁用 Template。

--tpmstate0 [file=]<volume> [,size=<DiskSize>] [,version=<v1.2|v2.0>]
配置用于存储 TPM 状态的磁盘。格式固定为 raw。

--unused[n] [file=]<volume>
对未使用卷的引用。该项供内部使用，不应手动修改。

--usb[n] [[host=]<HOSTUSBDEVICE|spice>] [,mapping=<mapping-id>] [,usb3=<1|0>]

配置 USB 设备（n 为 0 到 4；对于 machine version >= 7.1 且 ostype 为 l26 或 windows > 7 的客户机，n 可达 14）。

--vcpus <integer> (1 - N) (*default* = 0)

热插拔 vcpu 数量。

--vga [[type=]<enum>] [,clipboard=<vnc>] [,memory=<integer>]

配置 VGA 硬件。

--virtio[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>] [,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,ro=<1|0>] [,secs=<integer>] [,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>] [,snapshot=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>]

将卷用作 VIRTIO 硬盘（n 为 0 到 15）。

--virtiofs[n] [dirid=]<mapping-id> [,cache=<enum>] [,direct-io=<1|0>] [,expose-acl=<1|0>] [,expose-xattr=<1|0>]

使用 Virtio-fs 在主机和客户机之间共享目录的配置。

--vmgenid <UUID> (*default* = 1 (autogenerated))

设置 VM Generation ID。使用 1 可在创建或更新时自动生成，传入 0 可显式禁用。

--vmstatestorage <storage ID>

VM 状态卷/文件的默认存储。

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

创建虚拟硬件 watchdog 设备。

qm importdisk

`qm disk import` 的别名。

qm importovf <vmid> <manifest> <storage> [OPTIONS]

使用从 OVF manifest 读取的参数创建新虚拟机。

<vmid>: <integer> (100 - 999999999)

虚拟机的（唯一）ID。

<manifest>: <string>
ovf 文件路径。

<storage>: <storage ID>
目标存储 ID。

--dryrun <boolean>
打印提取出的 OVF 参数的解析表示，但不创建虚拟机。

--format <qcow2 | raw | vmdk>
目标格式。

qm list [OPTIONS]

虚拟机索引（按节点）。

--full <boolean>
确定活动虚拟机的完整状态。

qm listsnapshot <vmid>

列出所有快照。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm migrate <vmid> <target> [OPTIONS]

迁移虚拟机。创建新的迁移任务。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<target>: <string>
目标节点。

--bwlimit <integer> (0 - N) (*default* = migrate limit from datacenter or storage config)
覆盖 I/O 带宽限制（单位 KiB/s）。

--force <boolean>
允许迁移使用本地设备的虚拟机。只有 root 可使用此选项。

--migration_network <string>
用于迁移的（子）网络 CIDR。

--migration_type <insecure | secure>
迁移流量默认使用 SSH tunnel 加密。在安全且完全私有的网络上，可以禁用此项以提升性能。

--online <boolean>
如果虚拟机正在运行，则使用在线/实时迁移。如果虚拟机已停止，则忽略。

--targetstorage <string>
从源存储到目标存储的映射。仅提供单个存储 ID 会将所有源存储映射到该存储。提供特殊值 1 会将每个源存储映射到自身。

--with-local-disks <boolean>
为本地磁盘启用实时存储迁移。

qm monitor <vmid>

进入 QEMU Monitor 界面。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm move-disk

qm disk move 的别名。

qm move_disk

qm disk move 的别名。

qm mtunnel

由 qmigrate 使用，请勿手动使用。

qm nbdstop <vmid>

停止嵌入式 nbd 服务器。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm pending <vmid>

获取包含当前值和待应用值的虚拟机配置。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm reboot <vmid> [OPTIONS]

通过关闭虚拟机再重新启动来重启虚拟机。应用待处理更改。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--timeout <integer> (0 - N)
等待关机的最大超时时间（秒）。

qm remote-migrate <vmid> [<target-vmid>] <target-endpoint> --target-bridge <string> --target-storage <string> [OPTIONS]

将虚拟机迁移到远程集群。创建新的迁移任务。实验性功能！

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<target-vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<target-endpoint>: apitoken=<PVEAPIToken=user@realm!token=SECRET> ,host=<ADDRESS> [,fingerprint=<FINGERPRINT>] [,port=<PORT>]
远程目标端点。

--bwlimit <integer> (0 - N) (*default* = migrate limit from datacenter or storage config)
覆盖 I/O 带宽限制（单位 KiB/s）。

--delete <boolean> (*default* = 0)
迁移成功后删除原始虚拟机及相关数据。默认情况下，原始虚拟机会以停止状态保留在源集群上。

--online <boolean>
如果虚拟机正在运行，则使用在线/实时迁移。如果虚拟机已停止，则忽略。

--target-bridge <string>
从源 bridge 到目标 bridge 的映射。仅提供单个 bridge ID 会将所有源 bridge 映射到该 bridge。提供特殊值 1 会将每个源 bridge 映射到自身。

--target-storage <string>
从源存储到目标存储的映射。仅提供单个存储 ID 会将所有源存储映射到该存储。提供特殊值 1 会将每个源存储映射到自身。

qm rescan

`qm disk rescan` 的别名。

qm reset <vmid> [OPTIONS]

重置虚拟机。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

qm resize

`qm disk resize` 的别名。

qm resume <vmid> [OPTIONS]

恢复虚拟机。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--nocheck <boolean>
无可用描述。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

qm rollback <vmid> <snapname> [OPTIONS]
将虚拟机状态回滚到指定快照。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<snapname>: <string>
快照名称。

--start <boolean> (*default* = 0)
回滚成功后是否应启动虚拟机。（注意：如果快照包含 RAM，虚拟机会自动启动。）

qm sendkey <vmid> <key> [OPTIONS]
向虚拟机发送按键事件。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<key>: <string>
按键（qemu monitor 编码）。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

qm set <vmid> [OPTIONS]
设置虚拟机选项（同步 API）。对于涉及热插拔或存储分配的任何操作，应考虑改用 POST 方法。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

gicversion: <2 | 3 | 4 | max | host > (*default* = host)
设置 ARM 虚拟机的 gicversion。

virtualization: <boolean> (*default* = 0)
为 arm64 虚拟机启用嵌套虚拟化。

snapshot=<boolean>
控制 qemu 的快照模式功能。启用后，对磁盘所做的更改是临时的，并会在虚拟机关机时丢

pxvditemplate <boolean> (*default* = 0)
启用/禁用 PXVDI Template。

--acpi <boolean> (*default* = 1)
启用/禁用 ACPI。

--affinity <string>
用于执行客户机进程的主机核心列表，例如：0,5,8-11

--agent [enabled=]<1|0> [,freeze-fs-on-backup=<1|0>] [,fstrim_cloned_disks=<1|0>] [,type=<virtio|isa>]
启用/禁用与 QEMU Guest Agent 及其属性的通信。

--amd-sev [type=]<sev-type> [,allow-smt=<1|0>] [,kernel-hashes=<1|0>] [,no-debug=<1|0>] [,no-key-sharing=<1|0>]
AMD CPU 提供的 Secure Encrypted Virtualization（SEV）功能。

--arch <aarch64 | x86_64 | riscv64 |loongarch64>
虚拟处理器架构。默认为主机架构。

--args <string>
传递给 kvm 的任意参数。

--audio0 device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>]
配置音频设备，与 QXL/Spice 结合使用时很有用。

--autostart <boolean> (*default* = 0)
崩溃后自动重启（当前会被忽略）。

--balloon <integer> (0 - N)
虚拟机目标 RAM 容量，单位 MiB。使用 0 会禁用 balloon 驱动。

--bios <ovmf | seabios> (*default* = seabios)
选择 BIOS 实现。

--boot [[legacy=]<[acdn]{1,4}>] [,order=<device[;device...]>]
指定客户机启动顺序。请使用 `order=` 子属性；不带键或使用 `legacy=` 的用法已弃用。

--bootdisk (ide|sata|scsi|virtio)\d+
启用从指定磁盘启动。已弃用：请改用 `boot: order=foo;bar`。

--cdrom <volume>
这是选项 -ide2 的别名。

--cicustom [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>]
cloud-init：指定自定义文件，以替换启动时自动生成的文件。

--cipassword <password>
cloud-init：要分配给用户的密码。通常不建议使用此项，请改用 ssh 密钥。另请注意，较旧版本
cloud-init 不支持哈希密码。

--citype <configdrive2 | nocloud | opennebula>
指定 cloud-init 配置格式。默认值取决于配置的操作系统类型（`ostype`）。Linux 使用 `noclou`
格式，windows 使用 `configdrive2`。

--ciupgrade <boolean> (*default* = 1)
cloud-init：首次启动后自动执行软件包升级。

--ciuser <string>
cloud-init：要更改 ssh 密钥和密码的用户名，用于替代镜像中配置的默认用户。

--cores <integer> (1 - N) (*default* = 1)
每个 socket 的核心数。

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

--cpulimit <number> (0 - 128) (*default* = 0)
CPU 使用限制。

--cpuunits <integer> (1 - 262144) (*default* = cgroup v1: 1024, cgroup v2: 100)
虚拟机的 CPU 权重，在 cgroup v2 中会限制到 [1, 10000] 范围内。

--delete <string>
要删除的设置列表。

--description <string>
虚拟机描述。显示在 Web 界面的虚拟机摘要中，并作为注释保存到配置文件内。

--digest <string>
如果当前配置文件具有不同的 SHA1 digest，则阻止更改。这可用于防止并发修改。

--efidisk0 [file=]<volume> [,efitype=<2m|4m>] [,format=<enum>] [,import-from=<source volume>] [,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。注意，此处
SIZE_IN_GiB，并会将默认 EFI 变量复制到该卷中。使用 STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

--force <boolean>
强制物理移除。未设置时，仅从配置文件中移除磁盘，并创建名为 unused[n] 的附加配置条目，
ID。取消链接 unused[n] 始终会导致物理移除。

> **Note**
> 需要选项：`delete`

--freeze <boolean>
启动时冻结 CPU（使用 c monitor 命令开始执行）。

--hookscript <string>
将在虚拟机生命周期的各个步骤中执行的脚本。

--hostpci[n] [[host=]<HOSTPCIID[;HOSTPCIID2...]>] [,device-id=<hex id>] [,legacy-igd=<1|0>] [,mapping=<mapping-id>] [,mdev=<string>] [,pcie=<1|0>] [,rombar=<1|0>] [,romfile=<string>] [,sub-device-id=<hex id>] [,sub-vendor-id=<hex id>] [,vendor-id=<hex id>] [,x-vga=<1|0>] [,ramfb=<1|0>]
将主机 PCI 设备映射到客户机。

--hotplug <string> (*default* = network,disk,usb)

选择性启用热插拔功能。这是一个逗号分隔的热插拔功能列表：network、disk、cpu、memory 和 cloudinit。使用 0 可完全禁用热插拔。使用 1 作为值是默认值 network,disk,usb 的别名。对于 machine version >= 7.1 且 ostype 为 l26 或 windows > 7 的客户机，可以使用 USB 热插拔。

--hugepages <1024 | 2 | any>

启用/禁用 hugepages 内存。

--ide[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>] [,import-from=<source volume>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>]
[,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>]
[,mbps_wr_max=<mbps>] [,media=<cdrom|disk>] [,model=<model>]
[,replicate=<1|0>] [,rerror=<ignore|report|stop>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>]

将卷用作 IDE 硬盘或 CD-ROM（n 为 0 到 3）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。使用 STORAGE_ID:0 和 import-from 参数可从现有卷导入。

--ipconfig[n] [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>]
[,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>]

cloud-init：指定对应接口的 IP 地址和网关。

IP 地址使用 CIDR 表示法；网关是可选项，但需要指定同类型 IP。

IP 地址可以使用特殊字符串 dhcp 以使用 DHCP，此时不应显式提供网关。 对于 IPv6，可以使用 auto 以使用无状态自动配置。这需要 cloud-init 19.4 或更新版本。

如果启用了 cloud-init 且既未指定 IPv4 地址也未指定 IPv6 地址，则默认在 IPv4 上使用 dhcp。

--ivshmem size=<integer> [,name=<string>]

虚拟机间共享内存。适用于虚拟机之间或虚拟机与主机之间的直接通信。

--keephugepages <boolean> (*default* = 0)

与 hugepages 一起使用。启用后，虚拟机关机后 hugepages 不会被删除，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>

VNC 服务器的键盘布局。通常不需要此选项，且往往更适合在客户机 OS 内处理。

--kvm <boolean> (*default* = 1)

启用/禁用 KVM 硬件虚拟化。

--localtime <boolean>
将实时时钟（RTC）设置为本地时间。如果 ostype 表示 Microsoft Windows OS，则默认启用。

--lock <backup | clone | create | migrate | rollback | snapshot | snapshot-delete | suspended | suspending>
锁定/解锁虚拟机。

--machine [[type=]<machine type>] [,enable-s3=<1|0>] [,enable-s4=<1|0>] [,viommu=<intel|virtio>]
指定 QEMU machine。

--memory [current=]<integer>
内存属性。

--migrate_downtime <number> (0 - N) (*default* = 0.1)
设置迁移可容忍的最大停机时间（秒）。如果迁移在最后阶段因需要传输太多新脏 RAM 而无法收

--migrate_speed <integer> (0 - N) (*default* = 0)
设置迁移最大速度（MB/s）。值为 0 表示无限制。

--name <string>
设置虚拟机名称。仅用于配置 Web 界面。

--nameserver <string>
cloud-init：设置容器的 DNS 服务器 IP 地址。如果既未设置 searchdomain 也未设置 nameserver，创建时会自动使用主机上的设置。

--net[n] [model=]<enum> [,bridge=<bridge>] [,firewall=<1|0>] [,link_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX:XX>] [,mtu=<integer>] [,queues=<integer>] [,rate=<number>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,<model>=<macaddr>]
指定网络设备。

--numa <boolean> (*default* = 0)
启用/禁用 NUMA。

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

--onboot <boolean> (*default* = 0)
指定虚拟机是否在系统启动期间启动。

--ostype <l24 | l26 | other | solaris | w2k | w2k3 | w2k8 | win10 | win11 | win7 | win8 | wvista | wxp>
指定客户机操作系统。

--parallel[n] /dev/parport\d+|/dev/usb/lp\d+
映射主机并口设备（n 为 0 到 2）。

--protection <boolean> (*default* = 0)
设置虚拟机的保护标志。这会禁用删除虚拟机和删除磁盘操作。

--reboot <boolean> (*default* = 1)
允许重启。如果设置为 0，虚拟机会在重启时退出。

--revert <string>
还原待处理更改。

--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。

--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>] [,import-from=<source volume>] [,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>]
将卷用作 SATA 硬盘或 CD-ROM（n 为 0 到 5）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。使用 STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

--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>] [,import-from=<source volume>] [,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>]
[,product=<product>] [,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>] [,vendor=<vendor>] [,werror=<enum>]
[,wwn=<wwn>]
将卷用作 SCSI 硬盘或 CD-ROM（n 为 0 到 30）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。使用 STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

--scsihw <lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci | virtio-scsi-single> (*default* = lsi)
SCSI 控制器型号。

--searchdomain <string>
cloud-init：设置容器的 DNS 搜索域。如果既未设置 searchdomain 也未设置 nameserver，创

--serial[n] (/dev/.+|socket)
在虚拟机内部创建串口设备（n 为 0 到 3）。

--shares <integer> (0 - 50000) (*default* = 1000)
自动 ballooning 的内存 share 数量。数值越大，该虚拟机获得的内存越多。该数值相对于所有其
0 会禁用自动 ballooning。自动 ballooning 由 pvestatd 执行。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

--snapshot <boolean>
Qemu snapshot mode。虚拟机关机时 Qemu 会删除临时数据，虚拟机磁盘可始终保持干净。h
project.gitlab.io/qemu/system/images.html#disk-005fimages-005fsnapshot-005fmod

--smbios1 [base64=<1|0>] [,family=<Base64 encoded string>] [,manufacturer=<Base64 encoded string>] [,product=<Base64 encoded string>] [,serial=<Base64 encoded string>] [,sku=<Base64 encoded string>] [,uuid=<UUID>] [,version=<Base64 encoded string>]
指定 SMBIOS type 1 字段。

--smp <integer> (1 - N) (*default* = 1)
CPU 数量。请改用选项 -sockets。

--sockets <integer> (1 - N) (*default* = 1)
CPU socket 数量。

--spice_enhancements [foldersharing=<1|0>] [,videostreaming=<off|all|filter>]
配置 SPICE 的其他增强功能。

--sshkeys <filepath>
cloud-init：设置公共 SSH 密钥（每行一个密钥，OpenSSH 格式）。

--startdate (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (*default* = now)
设置实时时钟的初始日期。有效日期格式为：now、2006-06-17T16:01:21 或 2006-06-17。

--startup `[[order=]\d+] [,up=\d+] [,down=\d+]`
启动和关机行为。Order 是定义通用启动顺序的非负数。关机按相反顺序完成。此外，可以设置
up 或 down 延迟（秒），用于指定启动或停止下一台虚拟机之前等待的延迟。

--tablet <boolean> (*default* = 1)
启用/禁用 USB tablet 设备。

--tags <string>
虚拟机标签。该信息仅为元信息。

--tdf <boolean> (*default* = 0)
启用/禁用时间漂移修复。

--template <boolean> (*default* = 0)
启用/禁用 Template。

**--tpmstate0 [file=]<volume> [,import-from=<source volume>]
[,size=<DiskSize>] [,version=<v1.2|v2.0>]**
配置用于存储 TPM 状态的磁盘。格式固定为 raw。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。注意，此处会忽略 SIZE_IN_GiB，并改用 4 MiB。使用 STORAGE_ID:0 和 `import-f` 参数可从现有卷导入。

--unused[n] [file=]<volume>
对未使用卷的引用。该项供内部使用，不应手动修改。

**--usb[n] [[host=]<HOSTUSBDEVICE|spice>] [,mapping=<mapping-id>]
[,usb3=<1|0>]**
配置 USB 设备（n 为 0 到 4；对于 machine version >= 7.1 且 ostype 为 l26 或 windows > 7 的客户机，n 可达 14）。

--vcpus <integer> (1 - N) (*default* = 0)
热插拔 vcpu 数量。

--vga [[type=]<enum>] [,clipboard=<vnc>] [,memory=<integer>]
配置 VGA 硬件。

**--virtio[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>] [,import-from=<source volume>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>]
[,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>] [,media=<cdrom|disk>]
[,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,ro=<1|0>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,trans=<none|lba|auto>]
[,werror=<enum>]**
将卷用作 VIRTIO 硬盘（n 为 0 到 15）。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 分配新卷。使 STORAGE_ID:0 和 `import-from` 参数可从现有卷导入。

**--virtiofs[n] [dirid=]<mapping-id> [,cache=<enum>]
[,direct-io=<1|0>] [,expose-acl=<1|0>] [,expose-xattr=<1|0>]**
使用 Virtio-fs 在主机和客户机之间共享目录的配置。

--vmgenid <UUID> (*default* = 1 (autogenerated))
设置 VM Generation ID。使用 1 可在创建或更新时自动生成，传入 0 可显式禁用。

--vmstatestorage <storage ID>
VM 状态卷/文件的默认存储。

--watchdog [[model=]<i6300esb|ib700>] [,action=<enum>]
创建虚拟硬件 watchdog 设备。

qm showcmd <vmid> [OPTIONS]

显示用于启动虚拟机的命令行（调试信息）。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--pretty <boolean> (*default* = 0)
将每个选项放在新行上，以提升人工可读性。

--snapshot <string>
从给定快照获取配置值。

qm shutdown <vmid> [OPTIONS]

关闭虚拟机。这类似于按下物理机器上的电源按钮。它会向客户机 OS 发送 ACPI 事件，随后客户机 OS 应执行正常关机。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--forceStop <boolean> (*default* = 0)
确保虚拟机停止。

--keepActive <boolean> (*default* = 0)
不要停用存储卷。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

--timeout <integer> (0 - N)
等待最大超时时间（秒）。

qm snapshot <vmid> <snapname> [OPTIONS]

创建虚拟机快照。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

<snapname>: <string>
快照名称。

--description <string>
文本描述或注释。

--vmstate <boolean>
保存 vmstate。

qm start <vmid> [OPTIONS]

启动虚拟机。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--force-cpu <string>
使用给定字符串覆盖 QEMU 的 -cpu 参数。

--machine [[type=]<machine type>] [,enable-s3=<1|0>] [,enable-s4=<1|0>] [,viommu=<intel|virtio>]
指定 QEMU machine。

--migratedfrom <string>
集群节点名称。

--migration_network <string>
用于迁移的（子）网络 CIDR。

--migration_type <insecure | secure>
迁移流量默认使用 SSH tunnel 加密。在安全且完全私有的网络上，可以禁用此项以提升性能。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

--stateuri <string>
某些命令会从此位置保存/还原状态。

--targetstorage <string>
从源存储到目标存储的映射。仅提供单个存储 ID 会将所有源存储映射到该存储。提供特殊值 1 会将每个源存储映射到自身。

--timeout <integer> (0 - N) (*default* = max(30, vm memory in GiB))
等待最大超时时间（秒）。

qm status <vmid> [OPTIONS]

显示虚拟机状态。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--verbose <boolean>
详细输出格式。

qm stop <vmid> [OPTIONS]

停止虚拟机。qemu 进程会立即退出。这类似于拔掉运行中计算机的电源，可能损坏虚拟机数据。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--keepActive <boolean> (*default* = 0)
不要停用存储卷。

--migratedfrom <string>
集群节点名称。

--overrule-shutdown <boolean> (*default* = 0)
停止前尝试中止活动的 qmshutdown 任务。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

--timeout <integer> (0 - N)
等待最大超时时间（秒）。

qm suspend <vmid> [OPTIONS]

挂起虚拟机。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--skiplock <boolean>
忽略锁，只有 root 允许使用此选项。

--statestorage <storage ID>
虚拟机状态的存储。

> **Note**
> 需要选项：todisk

--todisk <boolean> (*default* = 0)
如果设置，则将虚拟机挂起到磁盘。下次虚拟机启动时会恢复。

qm template <vmid> [OPTIONS]

创建 Template。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--disk <efidisk0 | ide0 | ide1 | ide2 | ide3 | nvme0 |nvme1 |nvme2 |nvme3 |nvme4 |nvme5 |sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | scsi16 | scsi17 | scsi18 | scsi19 | scsi2 | scsi20 | scsi21 | scsi22 | scsi23 | scsi24 | scsi25 | scsi26 | scsi27 | scsi28 | scsi29 | scsi3 | scsi30 | scsi4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | virtio0 | virtio1 | virtio10 | virtio11 | virtio12 | virtio13 | virtio14 | virtio15 | virtio2 | virtio3 | virtio4 | virtio5 | virtio6 | virtio7 | virtio8 | virtio9>
如果只想将 1 块磁盘转换为基础镜像。

qm terminal <vmid> [OPTIONS]

使用串口设备打开终端（虚拟机需要已配置串口设备，例如 serial0: socket）。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--escape <string> (*default* = ^O)
转义字符。

--iface <serial0 | serial1 | serial2 | serial3>
选择串口设备。默认会直接使用第一个合适的设备。

qm unlink

qm disk unlink 的别名。

qm unlock <vmid>

解锁虚拟机。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm vncproxy <vmid>

将虚拟机 VNC 流量代理到 stdin/stdout。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

qm wait <vmid> [OPTIONS]

等待直到虚拟机停止。

<vmid>: <integer> (100 - 999999999)
虚拟机的（唯一）ID。

--timeout <integer> (1 - N)
超时时间（秒）。默认永久等待。

常用命令示例：

```
# qm list
# qm status 100
# qm config 100
```

A.11 qmrestore - 还原 QemuServer vzdump 备份

qmrestore help

qmrestore <archive> <vmid> [OPTIONS]

还原 QemuServer vzdump 备份。

<archive>: <string>
备份文件。可以传入 - 以从标准输入读取。

<vmid>: <integer> (100 - 999999999)
虚拟机的唯一 ID。

--bwlimit <number> (0 - N)
覆盖 I/O 带宽限制，单位为 KiB/s。

--force <boolean>
允许覆盖已有虚拟机。

--live-restore <boolean>
立即从备份启动虚拟机，并在后台继续还原。仅适用于 PBS。

--pool <string>
将虚拟机加入指定资源池。

--storage <storage ID>
默认存储。

--unique <boolean>
分配唯一的随机以太网地址。

常用命令示例

```
qmrestore /var/lib/vz/dump/vzdump-qemu-100.vma.zst 101 --storage local-lvm
qmrestore <archive> <vmid> --force
```

A.12 pct - Proxmox 容器工具包

pct <COMMAND> [ARGS] [OPTIONS]

pct clone <vmid> <newid> [OPTIONS]

创建容器克隆或副本

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<newid>: <integer> (100 - 999999999)
克隆的 VMID。

--bwlimit <number> (0 - N) (*default* = clone limit from datacenter or storage config)
覆盖 I/O 带宽限制（KiB/s）。

--description <string>
新 CT 的说明。

--full <boolean>
创建所有磁盘的完整副本。克隆普通 CT 时始终会这样做。对于 CT 模板，默认会尝试创建链接克

--hostname <string>
设置新 CT 的主机名。

--pool <string>
将新 CT 添加到指定池。

--snapname <string>
快照名称。

--storage <storage ID>
完整克隆的目标存储。

--target <string>
目标节点。仅当原始 VM 位于共享存储上时才允许。

pct config <vmid> [OPTIONS]

获取容器配置。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--current <boolean> (*default* = 0)
获取当前值（而不是待应用值）。

--snapshot <string>
从给定快照获取配置值。

pct console <vmid> [OPTIONS]

为指定容器启动控制台。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--escape \^?[a-z] (*default* = ^a)
转义序列前缀。例如，要使用 <Ctrl+b q> 作为转义序列，请传入 ^*b*。

pct cpusets

打印已分配 CPU 集合列表。

pct create <vmid> <ostemplate> [OPTIONS]

创建或还原容器。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<ostemplate>: <string>
OS 模板或备份文件。

--arch <amd64 | arm64 | armhf | i386 | riscv32 | riscv64> (*default* = amd64)
OS 架构类型。

--bwlimit <number> (0 - N) (*default* = restore limit from datacenter or storage config)
覆盖 I/O 带宽限制（KiB/s）。

--cmode <console | shell | tty> (*default* = tty)
控制台模式。默认情况下，console 命令会尝试连接到一个可用的 tty 设备。将 cmode 设置为 *console* 时，它会改为尝试附加到 /dev/console。将 cmode 设置为 *shell* 时，它只会在容器内调用 shell（不登录）。

--console <boolean> (*default* = 1)
将控制台设备（/dev/console）附加到容器。

--cores <integer> (1 - 8192)
分配给容器的核心数量。默认情况下，容器可以使用所有可用核心。

--cpulimit <number> (0 - 8192) (*default* = 0)
CPU 使用限制。

> **Note**
> 如果计算机有 2 个 CPU，则总共有 *2* 份 CPU 时间。值 *0* 表示不限制 CPU。

--cpuunits <integer> (0 - 500000) (*default* = cgroup v1: 1024, cgroup v2: 100)
容器的 CPU 权重，在 cgroup v2 中会限制到 [1, 10000]。

--debug <boolean> (*default* = 0)
尝试输出更详细的信息。目前这只会在启动时启用 debug 日志级别。

--description <string>
容器说明。显示在 Web 界面的 CT 摘要中，并作为注释保存在配置文件内。

--dev[n] [[path=]<Path>] [,deny-write=<1|0>] [,gid=<integer>] [,mode=<Octal access mode>] [,uid=<integer>]
要直通给容器的设备

--features [force_rw_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>] [,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>]
允许容器访问高级功能。

--force <boolean>
允许覆盖现有容器。

--hookscript <string>
在容器生命周期的各个步骤中执行的脚本。

--hostname <string>
设置容器主机名。

--ignore-unpack-errors <boolean>
提取模板时忽略错误。

--lock <backup | create | destroyed | disk | fstrim | migrate | mounted | rollback | snapshot | snapshot-delete>
锁定或解锁容器。

--memory <integer> (16 - N) (*default* = 512)
容器 RAM 容量，单位为 MB。

--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>]
将卷用作容器挂载点。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 可分配新卷。

--nameserver <string>
设置容器的 DNS 服务器 IP 地址。如果既未设置 searchdomain 也未设置 nameserver，创建时

--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)>] [,link_down=<1|0>] [,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
指定容器的网络接口。

--onboot <boolean> (*default* = 0)
指定容器是否在系统启动期间启动。

--ostype <alpine | archlinux | centos | debian | devuan | fedora | gentoo | nixos | opensuse | ubuntu | unmanaged>
OS 类型。用于设置容器内部配置，并对应 /usr/share/lxc/config/<ostype>.common.conf 中的 lxc 设置脚本。值 *unmanaged* 可用于跳过 OS 特定设置。

--password <password>
设置容器内 root 密码。

--pool <string>
将 VM 添加到指定池。

--protection <boolean> (*default* = 0)
设置容器的保护标志。这会阻止 CT 或 CT 磁盘的删除/更新操作。

--restore <boolean>
将其标记为还原任务。

--rootfs [volume=]<volume> [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskSize>]
将卷用作容器根目录。

--searchdomain <string>
设置容器的 DNS 搜索域。如果既未设置 searchdomain 也未设置 nameserver，创建时会自动使

--ssh-public-keys <filepath>
设置 SSH 公钥（每行一个密钥，OpenSSH 格式）。

--start <boolean> (*default* = 0)
创建成功完成后启动 CT。

--startup `[[order=]\d+] [,up=\d+] [,down=\d+]`
启动和关机行为。Order 是定义总体启动顺序的非负数。关机会按相反顺序执行。此外，可以设置 *up* 或 *down* 延迟，用于指定启动或停止下一台 VM 前等待的时间。

--storage <storage ID> (*default* = local)
默认存储。

--swap <integer> (0 - N) (*default* = 512)
容器 SWAP 容量，单位为 MB。

--tags <string>
容器标签。该信息仅为元信息。

--template <boolean> (*default* = 0)
启用或禁用模板。

--timezone <string>
容器中使用的时区。如果未设置该选项，则不执行任何操作。可设置为 *host* 以匹配宿主机时区，或 /usr/share/zoneinfo/zone.tab 中的任意时区选项。

--tty <integer> (0 - 6) (*default* = 2)
指定容器可用的 tty 数量

--unique <boolean>
分配唯一的随机以太网地址。

> **Note**
> 需要选项：restore

--unprivileged <boolean> (*default* = 0)
使容器以非特权用户身份运行。（不应手动修改。）

--unused[n] [volume=]<volume>
未使用卷的引用。该项供内部使用，不应手动修改。

pct delsnapshot <vmid> <snapname> [OPTIONS]

删除 LXC 快照。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<snapname>: <string>
快照名称。

--force <boolean>
即使移除磁盘快照失败，也从配置文件中移除。

pct destroy <vmid> [OPTIONS]

销毁容器（同时删除所有使用的文件）。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--destroy-unreferenced-disks <boolean>
如果设置，则额外销毁所有已启用存储中带有该 VMID 且未在配置中引用的磁盘。

--force <boolean> (*default* = 0)
即使正在运行也强制销毁。

--purge <boolean> (*default* = 0)
从所有相关配置中移除容器，例如备份作业、复制作业或 HA。相关 ACL 和防火墙条目将 始终被移除。

pct df <vmid>

获取容器当前磁盘使用量。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

pct enter <vmid> [OPTIONS]

为指定容器启动 shell。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--keep-env <boolean> (*default* = 1)
保留当前环境。该选项将在 PVE 9 中默认禁用。如果依赖保留环境，请使用此选项以适应未来版本

pct exec <vmid> [<extra-args>] [OPTIONS]

在指定容器内启动命令。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<extra-args>: <array>
作为数组传入的额外参数

--keep-env <boolean> (*default* = 1)
保留当前环境。该选项将在 PVE 9 中默认禁用。如果依赖保留环境，请使用此选项以适应未来版本

pct fsck <vmid> [OPTIONS]
在容器卷上运行文件系统检查（fsck）。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--device <mp0 | mp1 | mp10 | mp100 | mp101 | mp102 | mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | mp110 | mp111 | mp112 | mp113 | mp114 | mp115 | mp116 | mp117 | mp118 | mp119 | mp12 | mp120 | mp121 | mp122 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp133 | mp134 | mp135 | mp136 | mp137 | mp138 | mp139 | mp14 | mp140 | mp141 | mp142 | mp143 | mp144 | mp145 | mp146 | mp147 | mp148 | mp149 | mp15 | mp150 | mp151 | mp152 | mp153 | mp154 | mp155 | mp156 | mp157 | mp158 | mp159 | mp16 | mp160 | mp161 | mp162 | mp163 | mp164 | mp165 | mp166 | mp167 | mp168 | mp169 | mp17 | mp170 | mp171 | mp172 | mp173 | mp174 | mp175 | mp176 | mp177 | mp178 | mp179 | mp18 | mp180 | mp181 | mp182 | mp183 | mp184 | mp185 | mp186 | mp187 | mp188 | mp189 | mp19 | mp190 | mp191 | mp192 | mp193 | mp194 | mp195 | mp196 | mp197 | mp198 | mp199 | mp2 | mp20 | mp200 | mp201 | mp202 | mp203 | mp204 | mp205 | mp206 | mp207 | mp208 | mp209 | mp21 | mp210 | mp211 | mp212 | mp213 | mp214 | mp215 | mp216 | mp217 | mp218 | mp219 | mp22 | mp220 | mp221 | mp222 | mp223 | mp224 | mp225 | mp226 | mp227 | mp228 | mp229 | mp23 | mp230 | mp231 | mp232 | mp233 | mp234 | mp235 | mp236 | mp237 | mp238 | mp239 | mp24 | mp240 | mp241 | mp242 | mp243 | mp244 | mp245 | mp246 | mp247 | mp248 | mp249 | mp25 | mp250 | mp251 | mp252 | mp253 | mp254 | mp255 | mp26 | mp27 | mp28 | mp29 | mp3 | mp30 | mp31 | mp32 | mp33 | mp34 | mp35 | mp36 | mp37 | mp38 | mp39 | mp4 | mp40 | mp41 | mp42 | mp43 | mp44 | mp45 | mp46 | mp47 | mp48 | mp49 | mp5 | mp50 | mp51 | mp52 | mp53 | mp54 | mp55 | mp56 | mp57 | mp58 | mp59 | mp6 | mp60 | mp61 | mp62 | mp63 | mp64 | mp65 | mp66 | mp67 | mp68 | mp69 | mp7 | mp70 | mp71 | mp72 | mp73 | mp74 | mp75 | mp76 | mp77 | mp78 | mp79 | mp8 | mp80 | mp81 | mp82 | mp83 | mp84 | mp85 | mp86 | mp87 | mp88 | mp89 | mp9 | mp90 | mp91 | mp92 | mp93 | mp94 | mp95 | mp96 | mp97 | mp98 | mp99 | rootfs>
要运行文件系统检查的卷

--force <boolean> (*default* = 0)
即使文件系统看起来干净也强制检查

pct fstrim <vmid> [OPTIONS]
在选定 CT 及其挂载点上运行 fstrim，但不包括 bind 或只读挂载点。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--ignore-mountpoints <boolean>
跳过所有挂载点，仅在容器根目录上执行 fstrim。

pct help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助

--verbose <boolean>
详细输出格式。

pct list

LXC 容器索引（按节点）。

pct listsnapshot <vmid>

列出所有快照。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

pct migrate <vmid> <target> [OPTIONS]

将容器迁移到另一节点。创建新的迁移任务。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<target>: <string>
目标节点。

--bwlimit <number> (0 - N) (*default* = migrate limit from datacenter or storage config)
覆盖 I/O 带宽限制（KiB/s）。

--online <boolean>
使用在线/实时迁移。

--restart <boolean>
使用重启迁移

--target-storage <string>
从源存储到目标存储的映射。只提供单个存储 ID 时，会将所有源存储映射到该存储。提供特殊值 *1* 时，会将每个源存储映射到其自身。

--timeout <integer> (*default* = 180)
重启迁移时等待关机的超时时间，单位为秒

pct mount <vmid>

在宿主机上挂载容器文件系统。这会持有容器锁，仅用于紧急维护，因为它会阻止除启动和停止之外的

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

pct move-volume <vmid> <volume> [<storage>] [<target-vmid>] [<target-volume
[OPTIONS]

将 rootfs-/mp-volume 移动到不同存储或不同容器。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<volume>: <mp0 | mp1 | mp10 | mp100 | mp101 | mp102 | mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | mp110 | mp111 | mp112 | mp113 | mp114 | mp115 | mp116 | mp117 | mp118 | mp119 | mp12 | mp120 | mp121 | mp122 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp133 | mp134 | mp135 | mp136 | mp137 | mp138 | mp139 | mp14 | mp140 | mp141 | mp142 | mp143 | mp144 | mp145 | mp146 | mp147 | mp148 | mp149 | mp15 | mp150 | mp151 | mp152 | mp153 | mp154 | mp155 | mp156 | mp157 | mp158 | mp159 | mp16 | mp160 | mp161 | mp162 | mp163 | mp164 | mp165 | mp166 | mp167 | mp168 | mp169 | mp17 | mp170 | mp171 | mp172 | mp173 | mp174 | mp175 | mp176 | mp177 | mp178 | mp179 | mp18 | mp180 | mp181 | mp182 | mp183 | mp184 | mp185 | mp186 | mp187 | mp188 | mp189 | mp19 | mp190 | mp191 | mp192 | mp193 | mp194 | mp195 | mp196 | mp197 | mp198 | mp199 | mp2 | mp20 | mp200 | mp201 | mp202 | mp203 | mp204 | mp205 | mp206 | mp207 | mp208 | mp209 | mp21 | mp210 | mp211 | mp212 | mp213 | mp214 | mp215 | mp216 | mp217 | mp218 | mp219 | mp22 | mp220 | mp221 | mp222 | mp223 | mp224 | mp225 | mp226 | mp227 | mp228 | mp229 | mp23 | mp230 | mp231 | mp232 | mp233 | mp234 | mp235 | mp236 | mp237 | mp238 | mp239 | mp24 | mp240 | mp241 | mp242 | mp243 | mp244 | mp245 | mp246 | mp247 | mp248 | mp249 | mp25 | mp250 | mp251 | mp252 | mp253 | mp254 | mp255 | mp26 | mp27 | mp28 | mp29 | mp3 | mp30 | mp31 | mp32 | mp33 | mp34 | mp35 | mp36 | mp37 | mp38 | mp39 | mp4 | mp40 | mp41 | mp42 | mp43 | mp44 | mp45 | mp46 | mp47 | mp48 | mp49 | mp5 | mp50 | mp51 | mp52 | mp53 | mp54 | mp55 | mp56 | mp57 | mp58 | mp59 | mp6 | mp60 | mp61 | mp62 | mp63 | mp64 | mp65 | mp66 | mp67 | mp68 | mp69 | mp7 | mp70 | mp71 | mp72 | mp73 | mp74 | mp75 | mp76 | mp77 | mp78 | mp79 | mp8 | mp80 | mp81 | mp82 | mp83 | mp84 | mp85 | mp86 | mp87 | mp88 | mp89 | mp9 | mp90 | mp91 | mp92 | mp93 | mp94 | mp95 | mp96 | mp97 | mp98 | mp99 | rootfs | unused0 | unused1 | unused10 | unused100 | unused101 | unused102 | unused103 | unused104 | unused105 | unused106 | unused107 | unused108 | unused109 | unused11 | unused110 | unused111 | unused112 | unused113 | unused114 | unused115 | unused116 | unused117 | unused118 | unused119 | unused12 | unused120 | unused121 | unused122 | unused123 | unused124 | unused125 | unused126 | unused127 | unused128 | unused129 | unused13 | unused130 | unused131 | unused132 | unused133 | unused134 | unused135 | unused136 | unused137 | unused138 | unused139 | unused14 | unused140 | unused141 | unused142 | unused143 | unused144 | unused145 | unused146 | unused147 | unused148 | unused149 | unused15 | unused150 | unused151 | unused152 | unused153 | unused154 | unused155 | unused156 | unused157 | unused158 | unused159 | unused16 | unused160 | unused161 | unused162 | unused163 | unused164 | unused165 | unused166 | unused167 | unused168 | unused169 | unused17 | unused170 | unused171 | unused172 | unused173 | unused174 | unused175 | unused176 | unused177 | unused178 | unused179 | unused18 | unused180 | unused181 | unused182 | unused183 | unused184 | unused185 | unused186 | unused187 | unused188 | unused189 | unused19 | unused190 | unused191 | unused192 | unused193 | unused194 | unused195 | unused196 | unused197 | unused198 | unused199 | unused2 | unused20 | unused200

将被移动的卷。

<storage>: <storage ID>
目标存储。

<target-vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<target-volume>: <mp0 | mp1 | mp10 | mp100 | mp101 | mp102 | mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | mp110 | mp111 | mp112 | mp113 | mp114 | mp115 | mp116 | mp117 | mp118 | mp119 | mp12 | mp120 | mp121 | mp122 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp133 | mp134 | mp135 | mp136 | mp137 | mp138 | mp139 | mp14 | mp140 | mp141 | mp142 | mp143 | mp144 | mp145 | mp146 | mp147 | mp148 | mp149 | mp15 | mp150 | mp151 | mp152 | mp153 | mp154 | mp155 | mp156 | mp157 | mp158 | mp159 | mp16 | mp160 | mp161 | mp162 | mp163 | mp164 | mp165 | mp166 | mp167 | mp168 | mp169 | mp17 | mp170 | mp171 | mp172 | mp173 | mp174 | mp175 | mp176 | mp177 | mp178 | mp179 | mp18 | mp180 | mp181 | mp182 | mp183 | mp184 | mp185 | mp186 | mp187 | mp188 | mp189 | mp19 | mp190 | mp191 | mp192 | mp193 | mp194 | mp195 | mp196 | mp197 | mp198 | mp199 | mp2 | mp20 | mp200 | mp201 | mp202 | mp203 | mp204 | mp205 | mp206 | mp207 | mp208 | mp209 | mp21 | mp210 | mp211 | mp212 | mp213 | mp214 | mp215 | mp216 | mp217 | mp218 | mp219 | mp22 | mp220 | mp221 | mp222 | mp223 | mp224 | mp225 | mp226 | mp227 | mp228 | mp229 | mp23 | mp230 | mp231 | mp232 | mp233 | mp234 | mp235 | mp236 | mp237 | mp238 | mp239 | mp24 | mp240 | mp241 | mp242 | mp243 | mp244 | mp245 | mp246 | mp247 | mp248 | mp249 | mp25 | mp250 | mp251 | mp252 | mp253 | mp254 | mp255 | mp26 | mp27 | mp28 | mp29 | mp3 | mp30 | mp31 | mp32 | mp33 | mp34 | mp35 | mp36 | mp37 | mp38 | mp39 | mp4 | mp40 | mp41 | mp42 | mp43 | mp44 | mp45 | mp46 | mp47 | mp48 | mp49 | mp5 | mp50 | mp51 | mp52 | mp53 | mp54 | mp55 | mp56 | mp57 | mp58 | mp59 | mp6 | mp60 | mp61 | mp62 | mp63 | mp64 | mp65 | mp66 | mp67 | mp68 | mp69 | mp7 | mp70 | mp71 | mp72 | mp73 | mp74 | mp75 | mp76 | mp77 | mp78 | mp79 | mp8 | mp80 | mp81 | mp82 | mp83 | mp84 | mp85 | mp86 | mp87 | mp88 | mp89 | mp9 | mp90 | mp91 | mp92 | mp93 | mp94 | mp95 | mp96 | mp97 | mp98 | mp99 | rootfs | unused0 | unused1 | unused10 | unused100 | unused101 | unused102 | unused103 | unused104 | unused105 | unused106 | unused107 | unused108 | unused109 | unused11 | unused110 | unused111 | unused112 | unused113 | unused114 | unused115 | unused116 | unused117 | unused118 | unused119 | unused12 | unused120 | unused121 | unused122 | unused123 | unused124 | unused125 | unused126 | unused127 | unused128 | unused129 | unused13 | unused130 | unused131 | unused132 | unused133 | unused134 | unused135 | unused136 | unused137 | unused138 | unused139 | unused14 | unused140 | unused141 | unused142 | unused143 | unused144 | unused145 | unused146 | unused147 | unused148 | unused149 | unused15 | unused150 | unused151 | unused152 | unused153 | unused154 | unused155 | unused156 | unused157 | unused158 | unused159 | unused16 | unused160 | unused161 | unused162 | unused163 | unused164 | unused165 | unused166 | unused167 | unused168 | unused169 | unused17 | unused170 | unused171 | unused172 | unused173 | unused174 | unused175 | unused176 | unused177 | unused178 | unused179 | unused18 | unused180 | unused181 | unused182 | unused183 | unused184 | unused185 | unused186 | unused187 | unused188 | unused189 | unused19 | unused190 | unused191 | unused192 | unused193 | unused194 | unused195 | unused196 | unused197 | unused198 | unused199 | unused2 | unused20

卷将移动到的配置键。默认为源卷键。

--bwlimit <number> (0 - N) (*default* = clone limit from datacenter or storage config)
覆盖 I/O 带宽限制（KiB/s）。

--delete <boolean> (*default* = 0)
成功复制后删除原始卷。默认情况下，原始卷会保留为未使用卷条目。

--digest <string>
如果当前配置文件具有不同的 SHA1 digest，则阻止更改。这可用于防止并发修改。

--target-digest <string>
如果目标容器的当前配置文件具有不同的 SHA1 digest，则阻止更改。这可用于防止并发修改。

pct move_volume

`pct move-volume` 的别名。

pct pending <vmid>

获取容器配置，包括待应用更改。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

pct pull <vmid> <path> <destination> [OPTIONS]

将文件从容器复制到本地系统。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<path>: <string>
要从容器内拉取的文件路径。

<destination>: <string>
目标

--group <string>
所有者组名或 ID。

--perms <string>
要使用的文件权限（默认为八进制，使用 *0x* 前缀表示十六进制）。

--user <string>
所有者用户名或 ID。

pct push <vmid> <file> <destination> [OPTIONS]

将本地文件复制到容器。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<file>: <string>
本地文件路径。

<destination>: <string>
容器内要写入的目标位置。

--group <string>
所有者组名或 ID。使用名称时，该名称必须存在于容器内。

--perms <string>
要使用的文件权限（默认为八进制，使用 *0x* 前缀表示十六进制）。

--user <string>
所有者用户名或 ID。使用名称时，该名称必须存在于容器内。

pct reboot <vmid> [OPTIONS]

通过关闭并再次启动来重启容器。会应用待处理更改。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--timeout <integer> (0 - N)
等待关机的最大超时时间，单位为秒。

pct remote-migrate <vmid> [<target-vmid>] <target-endpoint> --target-bridge <string> --target-storage <string> [OPTIONS]

将容器迁移到远程集群。创建新的迁移任务。 实验性功能！

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<target-vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<target-endpoint>: apitoken=<PVEAPIToken=user@realm!token=SECRET> ,host=<ADDRESS> [,fingerprint=<FINGERPRINT>] [,port=<PORT>]
远程目标端点

--bwlimit <integer> (0 - N) (*default* = migrate limit from datacenter or storage config)
覆盖 I/O 带宽限制（KiB/s）。

--delete <boolean> (*default* = 0)
成功迁移后删除原始 CT 及相关数据。默认情况下，原始 CT 会以停止状态保留在源集群上。

--online <boolean>
使用在线/实时迁移。

--restart <boolean>
使用重启迁移

--target-bridge <string>
从源网桥到目标网桥的映射。只提供单个网桥 ID 时，会将所有源网桥映射到该网桥。提供特殊值 *1* 时，会将每个源网桥映射到其自身。

--target-storage <string>
从源存储到目标存储的映射。只提供单个存储 ID 时，会将所有源存储映射到该存储。提供特殊值 *1* 时，会将每个源存储映射到其自身。

--timeout <integer> (*default* = 180)
重启迁移时等待关机的超时时间，单位为秒

pct rescan [OPTIONS]

重新扫描所有存储，并更新磁盘大小和未使用磁盘镜像。

--dryrun <boolean> (*default* = 0)
不要将更改实际写入配置。

--vmid <integer> (100 - 999999999)
VM 的唯一 ID。

pct resize <vmid> <disk> <size> [OPTIONS]

调整容器挂载点大小。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<disk>: <mp0 | mp1 | mp10 | mp100 | mp101 | mp102 | mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | mp110 | mp111 | mp112 | mp113 | mp114 | mp115 | mp116 | mp117 | mp118 | mp119 | mp12 | mp120 | mp121 | mp122 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp133 | mp134 | mp135 | mp136 | mp137 | mp138 | mp139 | mp14 | mp140 | mp141 | mp142 | mp143 | mp144 | mp145 | mp146 | mp147 | mp148 | mp149 | mp15 | mp150 | mp151 | mp152 | mp153 | mp154 | mp155 | mp156 | mp157 | mp158 | mp159 | mp16 | mp160 | mp161 | mp162 | mp163 | mp164 | mp165 | mp166 | mp167 | mp168 | mp169 | mp17 | mp170 | mp171 | mp172 | mp173 | mp174 | mp175 | mp176 | mp177 | mp178 | mp179 | mp18 | mp180 | mp181 | mp182 | mp183 | mp184 | mp185 | mp186 | mp187 | mp188 | mp189 | mp19 | mp190 | mp191 | mp192 | mp193 | mp194 | mp195 | mp196 | mp197 | mp198 | mp199 | mp2 | mp20 | mp200 | mp201 | mp202 | mp203 | mp204 | mp205 | mp206 | mp207 | mp208 | mp209 | mp21 | mp210 | mp211 | mp212 | mp213 | mp214 | mp215 | mp216 | mp217 | mp218 | mp219 | mp22 | mp220 | mp221 | mp222 | mp223 | mp224 | mp225 | mp226 | mp227 | mp228 | mp229 | mp23 | mp230 | mp231 | mp232 | mp233 | mp234 | mp235 | mp236 | mp237 | mp238 | mp239 | mp24 | mp240 | mp241 | mp242 | mp243 | mp244 | mp245 | mp246 | mp247 | mp248 | mp249 | mp25 | mp250 | mp251 | mp252 | mp253 | mp254 | mp255 | mp26 | mp27 | mp28 | mp29 | mp3 | mp30 | mp31 | mp32 | mp33 | mp34 | mp35 | mp36 | mp37 | mp38 | mp39 | mp4 | mp40 | mp41 | mp42 | mp43 | mp44 | mp45 | mp46 | mp47 | mp48 | mp49 | mp5 | mp50 | mp51 | mp52 | mp53 | mp54 | mp55 | mp56 | mp57 | mp58 | mp59 | mp6 | mp60 | mp61 | mp62 | mp63 | mp64 | mp65 | mp66 | mp67 | mp68 | mp69 | mp7 | mp70 | mp71 | mp72 | mp73 | mp74 | mp75 | mp76 | mp77 | mp78 | mp79 | mp8 | mp80 | mp81 | mp82 | mp83 | mp84 | mp85 | mp86 | mp87 | mp88 | mp89 | mp9 | mp90 | mp91 | mp92 | mp93 | mp94 | mp95 | mp96 | mp97 | mp98 | mp99 | rootfs>

要调整大小的磁盘。

<size>: \+?\d+(\.\d+)?[KMGT]?

新的大小。带有 + 号时，该值会加到卷的实际大小上；不带时，该值被视为绝对大小。不支持缩小

--digest <string>

如果当前配置文件具有不同的 SHA1 digest，则阻止更改。这可用于防止并发修改。

pct restore <vmid> <ostemplate> [OPTIONS]

创建或还原容器。

<vmid>: <integer> (100 - 999999999)

VM 的唯一 ID。

<ostemplate>: <string>

OS 模板或备份文件。

--arch <amd64 | arm64 | armhf | i386 | riscv32 | riscv64> (*default* = amd64)

OS 架构类型。

--bwlimit <number> (0 - N) (*default* = restore limit from datacenter or storage config)
覆盖 I/O 带宽限制（KiB/s）。

--cmode <console | shell | tty> (*default* = tty)
控制台模式。默认情况下，console 命令会尝试连接到一个可用的 tty 设备。将 cmode 设置为 *console* 时，它会改为尝试附加到 /dev/console。将 cmode 设置为 *shell* 时，它只会在容器内调用 shell（不登录）。

--console <boolean> (*default* = 1)
将控制台设备（/dev/console）附加到容器。

--cores <integer> (1 - 8192)
分配给容器的核心数量。默认情况下，容器可以使用所有可用核心。

--cpulimit <number> (0 - 8192) (*default* = 0)
CPU 使用限制。

> **Note**
> 如果计算机有 2 个 CPU，则总共有 *2* 份 CPU 时间。值 *0* 表示不限制 CPU。

--cpuunits <integer> (0 - 500000) (*default* = cgroup v1: 1024, cgroup v2: 100)
容器的 CPU 权重，在 cgroup v2 中会限制到 [1, 10000]。

--debug <boolean> (*default* = 0)
尝试输出更详细的信息。目前这只会在启动时启用 debug 日志级别。

--description <string>
容器说明。显示在 Web 界面的 CT 摘要中，并作为注释保存在配置文件内。

--dev[n] [[path=]<Path>] [,deny-write=<1|0>] [,gid=<integer>] [,mode=<Octal access mode>] [,uid=<integer>]
要直通给容器的设备

--features [force_rw_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>] [,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>]
允许容器访问高级功能。

--force <boolean>
允许覆盖现有容器。

--hookscript <string>
在容器生命周期的各个步骤中执行的脚本。

--hostname <string>
设置容器主机名。

--ignore-unpack-errors <boolean>
提取模板时忽略错误。

--lock <backup | create | destroyed | disk | fstrim | migrate | mounted | rollback | snapshot | snapshot-delete>
锁定或解锁容器。

--memory <integer> (16 - N) (*default* = 512)
容器 RAM 容量，单位为 MB。

--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>]
将卷用作容器挂载点。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 可分配新卷。

--nameserver <string>
设置容器的 DNS 服务器 IP 地址。如果既未设置 searchdomain 也未设置 nameserver，创建时

--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)>] [,link_down=<1|0>] [,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
指定容器的网络接口。

--onboot <boolean> (*default* = 0)
指定容器是否在系统启动期间启动。

--ostype <alpine | archlinux | centos | debian | devuan | fedora | gentoo | nixos | opensuse | ubuntu | unmanaged>
OS 类型。用于设置容器内部配置，并对应 /usr/share/lxc/config/<ostype>.common.conf 中的 lxc 设置脚本。值 *unmanaged* 可用于跳过 OS 特定设置。

--password <password>
设置容器内 root 密码。

--pool <string>
将 VM 添加到指定池。

--protection <boolean> (*default* = 0)
设置容器的保护标志。这会阻止 CT 或 CT 磁盘的删除/更新操作。

--rootfs [volume=]<volume> [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskSize>]
将卷用作容器根目录。

--searchdomain <string>
设置容器的 DNS 搜索域。如果既未设置 searchdomain 也未设置 nameserver，创建时会自动

--ssh-public-keys <filepath>
设置 SSH 公钥（每行一个密钥，OpenSSH 格式）。

--start <boolean> (*default* = 0)
创建成功完成后启动 CT。

--startup `[[order=]\d+] [,up=\d+] [,down=\d+]`
启动和关机行为。Order 是定义总体启动顺序的非负数。关机会按相反顺序执行。此外，可以设置 *up* 或 *down* 延迟，用于指定启动或停止下一台 VM 前等待的时间。

--storage <storage ID> (*default* = local)
默认存储。

--swap <integer> (0 - N) (*default* = 512)
容器 SWAP 容量，单位为 MB。

--tags <string>
容器标签。该信息仅为元信息。

--template <boolean> (*default* = 0)
启用或禁用模板。

--timezone <string>
容器中使用的时区。如果未设置该选项，则不执行任何操作。可设置为 *host* 以匹配宿主机时区，/usr/share/zoneinfo/zone.tab 中的任意时区选项。

--tty <integer> (0 - 6) (*default* = 2)
指定容器可用的 tty 数量

--unique <boolean>
分配唯一的随机以太网地址。

> **Note**
> 需要选项：restore

--unprivileged <boolean> (*default* = 0)
使容器以非特权用户身份运行。（不应手动修改。）

--unused[n] [volume=]<volume>
未使用卷的引用。该项供内部使用，不应手动修改。

pct resume <vmid>

恢复容器。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

pct rollback <vmid> <snapname> [OPTIONS]

将 LXC 状态回滚到指定快照。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<snapname>: <string>
快照名称。

--start <boolean> (*default* = 0)
回滚成功后是否应启动容器

pct set <vmid> [OPTIONS]

设置容器选项。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--arch <amd64 | arm64 | armhf | i386 | riscv32 | riscv64> (*default* = amd64)
OS 架构类型。

--cmode <console | shell | tty> (*default* = tty)
控制台模式。默认情况下，console 命令会尝试连接到一个可用的 tty 设备。将 cmode 设置为 *console* 时，它会改为尝试附加到 /dev/console。将 cmode 设置为 *shell* 时，它只会在容器内调用 shell（不登录）。

--console <boolean> (*default* = 1)
将控制台设备（/dev/console）附加到容器。

--cores <integer> (1 - 8192)
分配给容器的核心数量。默认情况下，容器可以使用所有可用核心。

--cpulimit <number> (0 - 8192) (*default* = 0)
CPU 使用限制。

> **Note**
> 如果计算机有 2 个 CPU，则总共有 *2* 份 CPU 时间。值 *0* 表示不限制 CPU。

--cpuunits <integer> (0 - 500000) (*default* = cgroup v1: 1024, cgroup v2: 100)
容器的 CPU 权重，在 cgroup v2 中会限制到 [1, 10000]。

--debug <boolean> (*default* = 0)
尝试输出更详细的信息。目前这只会在启动时启用 debug 日志级别。

--delete <string>
要删除的设置列表。

--description <string>
容器说明。显示在 Web 界面的 CT 摘要中，并作为注释保存在配置文件内。

--dev[n] [[path=]<Path>] [,deny-write=<1|0>] [,gid=<integer>] [,mode=<Octal access mode>] [,uid=<integer>]
要直通给容器的设备

--digest <string>
如果当前配置文件具有不同的 SHA1 digest，则阻止更改。这可用于防止并发修改。

--features [force_rw_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>] [,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>]
允许容器访问高级功能。

--hookscript <string>
在容器生命周期的各个步骤中执行的脚本。

--hostname <string>
设置容器主机名。

--lock <backup | create | destroyed | disk | fstrim | migrate | mounted | rollback | snapshot | snapshot-delete>
锁定或解锁容器。

--memory <integer> (16 - N) (*default* = 512)
容器 RAM 容量，单位为 MB。

--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>]
将卷用作容器挂载点。使用特殊语法 STORAGE_ID:SIZE_IN_GiB 可分配新卷。

--nameserver <string>
设置容器的 DNS 服务器 IP 地址。如果既未设置 searchdomain 也未设置 nameserver，创建时

--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)>] [,link_down=<1|0>] [,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
指定容器的网络接口。

--onboot <boolean> (*default* = 0)
指定容器是否在系统启动期间启动。

--ostype <alpine | archlinux | centos | debian | devuan | fedora | gentoo | nixos | opensuse | ubuntu | unmanaged>
OS 类型。用于设置容器内部配置，并对应 /usr/share/lxc/config/<ostype>.common.conf 中的 lxc 设置脚本。值 *unmanaged* 可用于跳过 OS 特定设置。

--protection <boolean> (*default* = 0)
设置容器的保护标志。这会阻止 CT 或 CT 磁盘的删除/更新操作。

--revert <string>
还原待应用更改。

--rootfs [volume=]<volume> [,acl=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskSize>]
将卷用作容器根目录。

--searchdomain <string>
设置容器的 DNS 搜索域。如果既未设置 searchdomain 也未设置 nameserver，创建时会自动

--startup`[[order=]\d+] [,up=\d+] [,down=\d+]`
启动和关机行为。Order 是定义总体启动顺序的非负数。关机会按相反顺序执行。此外，可以设置 *up* 或 *down* 延迟，用于指定启动或停止下一台 VM 前等待的时间。

--swap <integer> (0 - N) (*default* = 512)
容器 SWAP 容量，单位为 MB。

--tags <string>
容器标签。该信息仅为元信息。

--template <boolean> (*default* = 0)
启用或禁用模板。

--timezone <string>
容器中使用的时区。如果未设置该选项，则不执行任何操作。可设置为 *host* 以匹配宿主机时区，/usr/share/zoneinfo/zone.tab 中的任意时区选项。

--tty <integer> (0 - 6) (*default* = 2)
指定容器可用的 tty 数量

--unprivileged <boolean> (*default* = 0)
使容器以非特权用户身份运行。（不应手动修改。）

--unused[n] [volume=]<volume>
未使用卷的引用。该项供内部使用，不应手动修改。

pct shutdown <vmid> [OPTIONS]

关闭容器。这会触发容器的正常关机，详情请参见 lxc-stop(1)。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--forceStop <boolean> (*default* = 0)
确保容器停止。

--timeout <integer> (0 - N) (*default* = 60)
等待最大超时时间，单位为秒。

pct snapshot <vmid> <snapname> [OPTIONS]

创建容器快照。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

<snapname>: <string>
快照名称。

--description <string>
文本说明或注释。

pct start <vmid> [OPTIONS]

启动容器。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--debug <boolean> (*default* = 0)
如果设置，则在启动时启用非常详细的 debug 日志级别。

--skiplock <boolean>
忽略锁 - 只有 root 允许使用该选项。

pct status <vmid> [OPTIONS]

显示 CT 状态。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--verbose <boolean>
详细输出格式

pct stop <vmid> [OPTIONS]

停止容器。这会突然停止容器内运行的所有进程。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

--overrule-shutdown <boolean> (*default* = 0)
停止前尝试中止活动的 *vzshutdown* 任务。

--skiplock <boolean>
忽略锁 - 只有 root 允许使用该选项。

pct suspend <vmid>

挂起容器。这是实验性功能。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

pct template <vmid>

创建模板。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

pct unlock <vmid>

解锁 VM。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

pct unmount <vmid>

卸载容器文件系统。

<vmid>: <integer> (100 - 999999999)
VM 的唯一 ID。

常用命令示例：

```
pct list
pct status <vmid>
pct start <vmid>
pct shutdown <vmid>
```

A.13 pveam - Proxmox VE Appliance 管理器

pveam <COMMAND> [ARGS] [OPTIONS]

pveam available [OPTIONS]

列出可用模板。

--section <mail | system | turnkeylinux>
将列表限制为指定分类。

pveam download <storage> <template>

下载应用模板。

<storage>: <storage ID>
用于保存模板的存储

<template>: <string>
要下载的模板

pveam help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助。

--verbose <boolean>
详细输出格式。

pveam list <storage>
获取存储上的所有模板列表

<storage>: <storage ID>
仅列出指定存储上的模板

pveam remove <template_path>
移除模板。

<template_path>: <string>
要移除的模板。

pveam update
更新容器模板数据库。

常用命令示例

```
pveam update
pveam available --section system
pveam list local
```

A.14 pvecm - Proxmox VE 集群管理器

pvecm <COMMAND> [ARGS] [OPTIONS]

pvecm add <hostname> [OPTIONS]
将当前节点添加到现有集群。

<hostname>: <string>
现有集群成员的主机名（或 IP）。

--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}
证书 SHA 256 指纹。

--force <boolean>
如果节点已存在，不抛出错误。

--link[n] [address=]<IP> [,priority=<integer>]
单条 corosync 链路的地址和优先级信息。（最多支持 8 条链路：link0..link7）

--nodeid <integer> (1 - N)
该节点的节点 ID。

--use_ssh <boolean>
始终使用 SSH 加入，即使对端可能通过 API 完成加入。

--votes <integer> (0 - N)
该节点的票数。

pvecm addnode <node> [OPTIONS]

向集群配置中添加节点。此调用供内部使用。

<node>: <string>
集群节点名称。

--apiversion <integer>
新节点的 JOIN_API_VERSION。

--force <boolean>
如果节点已存在，不抛出错误。

--link[n] [address=]<IP> [,priority=<integer>]
单条 corosync 链路的地址和优先级信息。（最多支持 8 条链路：link0..link7）

--new_node_ip <string>
要添加节点的 IP 地址。如果未给出链路，则使用该地址。

--nodeid <integer> (1 - N)
该节点的节点 ID。

--votes <integer> (0 - N)
该节点的票数。

pvecm apiver

返回该节点上可用的集群加入 API 版本。

pvecm create <clustername> [OPTIONS]

生成新的集群配置。如果未给出链路，则默认使用本地 IP 地址作为 link0。

<clustername>: <string>
集群名称。

--link[n] [address=]<IP> [,priority=<integer>]
单条 corosync 链路的地址和优先级信息。（最多支持 8 条链路：link0..link7）

--nodeid <integer> (1 - N)
该节点的节点 ID。

--votes <integer> (1 - N)
该节点的票数。

pvecm delnode <node>

从集群配置中移除节点。

<node>: <string>
集群节点名称。

pvecm expected <expected>
告知 corosync 新的 expected votes 值。

<expected>: <integer> (1 - N)
预期票数。

pvecm help [OPTIONS]
获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助。

--verbose <boolean>
详细输出格式。

pvecm keygen <filename>
为 corosync 生成新的加密密钥。

<filename>: <string>
输出文件名。

pvecm mtunnel [<extra-args>] [OPTIONS]
供 VM/CT 迁移使用 - 请勿手动使用。

<extra-args>: <array>
以数组形式传入的额外参数。

--get_migration_ip <boolean> (*default* = 0)
如果已配置，则返回迁移 IP。

--migration_network <string>
用于检测本地迁移 IP 的迁移网络。

--run-command <boolean>
使用 tcp socket 作为标准输入来运行命令。该命令会先通过标准输出打印 IP 地址和端口，每项单

pvecm nodes
显示本地视角下的集群节点。

pvecm qdevice remove
移除已配置的 QDevice。

pvecm qdevice setup <address> [OPTIONS]
设置 QDevice 的使用。

<address>: <string>
指定外部 corosync QDevice 的网络地址。

--force <boolean>
对可能存在危险的操作不抛出错误。

--network <string>
用于连接外部 qdevice 的网络。

pvecm status

显示本地视角下的集群状态。

pvecm updatecerts [OPTIONS]

更新节点证书（并生成所有需要的文件/目录）。

--force <boolean>
强制生成新的 SSL 证书。

--silent <boolean>
忽略错误（例如集群没有仲裁时）。

--unmerge-known-hosts <boolean> (*default* = 0)
取消合并旧版 SSH known hosts。

A.15 pvesr - Proxmox VE 存储复制

pvesr <COMMAND> [ARGS] [OPTIONS]

常用命令示例：

```
pvesr list
pvesr status
```

pvesr create-local-job <id> <target> [OPTIONS]

创建新的复制任务。

<id>: [1-9][0-9]{2,8}-\d{1,9}
复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

<target>: <string>
目标节点。

--comment <string>
说明。

--disable <boolean>
用于禁用/停用该条目的标志。

--rate <number> (1 - N)
速率限制，单位为 mbps（兆字节每秒），使用浮点数表示。

--remove_job <full | local>
将复制任务标记为待删除。该任务会移除所有本地复制快照。设置为 *full* 时，还会尝试移除目标节

--schedule <string> (*default* = */15)
存储复制调度计划。其格式是 systemd 日历事件的一个子集。

--source <string>
供内部使用，用于检测客户机是否已被接管。

pvesr delete <id> [OPTIONS]

将复制任务标记为待删除。

<id>: [1-9][0-9]{2,8}-\d{1,9}
复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

--force <boolean> (*default* = 0)
会移除任务配置条目，但不会执行清理。

--keep <boolean> (*default* = 0)
保留目标端的已复制数据（不移除）。

pvesr disable <id>

禁用复制任务。

<id>: [1-9][0-9]{2,8}-\d{1,9}
复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

pvesr enable <id>

启用复制任务。

<id>: [1-9][0-9]{2,8}-\d{1,9}
复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

pvesr finalize-local-job <id> [<extra-args>] [OPTIONS]

完成复制任务。此操作会移除时间戳不同于 <last_sync> 的所有复制快照。

<id>: [1-9][0-9]{2,8}-\d{1,9}
复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

<extra-args>: <array>
需要纳入处理的卷 ID 列表。

--last_sync <integer> (0 - N)
上次成功同步的时间（UNIX epoch）。如果未指定，则会移除所有复制快照。

pvesr help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助。

--verbose <boolean>
详细输出格式。

pvesr list

列出复制任务。

pvesr prepare-local-job <id> [<extra-args>] [OPTIONS]

为启动复制任务做准备。此命令会在复制开始前于目标节点上调用。该调用供内部使用，并在 std-
out 上返回一个 JSON 对象。该方法首先测试 VM <vmid> 是否位于本地节点；如果是，则立即停止。
ID 的快照，并移除时间戳不同于 <last_sync> 的所有复制快照。它还会移除所有未使用的卷。返回值

<id>: [1-9][0-9]{2,8}-\d{1,9}
复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

<extra-args>: <array>
需要纳入处理的卷 ID 列表。

--force <boolean> (*default* = 0)
允许移除所有现有卷（空卷列表）。

--last_sync <integer> (0 - N)
上次成功同步的时间（UNIX epoch）。如果未指定，则会移除所有复制快照。

--parent_snapname <string>
快照名称。

--scan <string>
要扫描陈旧卷的存储 ID 列表。

pvesr read <id>

读取复制任务配置。

<id>: [1-9][0-9]{2,8}-\d{1,9}
复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

pvesr run [OPTIONS]

该方法由 systemd-timer 调用，用于执行所有同步任务或指定的同步任务。

--id [1-9][0-9]{2,8}-\d{1,9}
复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

--mail <boolean> (*default* = 0)
发生失败时发送电子邮件通知。

--verbose <boolean> (*default* = 0)
向 stdout 打印更详细的日志。

pvesr schedule-now <id>

调度复制任务，使其尽快启动。

<id>: [1-9][0-9]{2,8}-\d{1,9}

复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

pvesr set-state <vmid> <state>

在迁移时设置任务复制状态。该调用供内部使用，接受以 JSON 对象表示的任务状态。

<vmid>: <integer> (100 - 999999999)

虚拟机的（唯一）ID。

<state>: <string>

以 JSON 解码字符串表示的任务状态。

pvesr status [OPTIONS]

列出此节点上所有复制任务的状态。

--guest <integer> (100 - 999999999)

仅列出此客户机的复制任务。

pvesr update <id> [OPTIONS]

更新复制任务配置。

<id>: [1-9][0-9]{2,8}-\d{1,9}

复制任务 ID。该 ID 由客户机 ID 和任务编号组成，中间用连字符分隔，即 *<GUEST>-<JOBNUM*

--comment <string>

说明。

--delete <string>

要删除的设置列表。

--digest <string>

如果当前配置文件的 digest 不同，则阻止更改。此项可用于防止并发修改。

--disable <boolean>

用于禁用/停用该条目的标志。

--rate <number> (1 - N)

速率限制，单位为 mbps（兆字节每秒），使用浮点数表示。

--remove_job <full | local>

将复制任务标记为待删除。该任务会移除所有本地复制快照。设置为 *full* 时，还会尝试移除目标节

--schedule <string> (*default* = */15)

存储复制调度计划。其格式是 systemd 日历事件的一个子集。

--source <string>

供内部使用，用于检测客户机是否已被接管。

A.16 pveum - Proxmox VE 用户管理器

pveum <COMMAND> [ARGS] [OPTIONS]

pveum acl delete <path> --roles <string> [OPTIONS]

更新访问控制列表（添加或移除权限）。

<path>: <string>
访问控制路径

--groups <string>
组列表。

--propagate <boolean> (*default* = 1)
允许传播（继承）权限。

--roles <string>
角色列表。

--tokens <string>
API token 列表。

--users <string>
用户列表。

pveum acl list [FORMAT_OPTIONS]

获取访问控制列表（ACL）。

pveum acl modify <path> --roles <string> [OPTIONS]

更新访问控制列表（添加或移除权限）。

<path>: <string>
访问控制路径

--groups <string>
组列表。

--propagate <boolean> (*default* = 1)
允许传播（继承）权限。

--roles <string>
角色列表。

--tokens <string>
API token 列表。

--users <string>
用户列表。

pveum acldel

pveum acl delete 的别名。

pveum aclmod

pveum acl modify 的别名。

pveum group add <groupid> [OPTIONS]

创建新组。

<groupid>: <string>
无可用描述

--comment <string>
无可用描述

pveum group delete <groupid>

删除组。

<groupid>: <string>
无可用描述

pveum group list [FORMAT_OPTIONS]

组索引。

pveum group modify <groupid> [OPTIONS]

更新组数据。

<groupid>: <string>
无可用描述

--comment <string>
无可用描述

pveum groupadd

pveum group add 的别名。

pveum groupdel

pveum group delete 的别名。

pveum groupmod

pveum group modify 的别名。

pveum help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助

--verbose <boolean>
详细输出格式。

pveum passwd <userid> [OPTIONS]

更改用户密码。

<userid>: <string>
完整用户 ID，格式为 name@realm。

--confirmation-password <string>
执行此次更改的用户的当前密码。

pveum pool add <poolid> [OPTIONS]

创建新资源池。

<poolid>: <string>
无可用描述

--comment <string>
无可用描述

pveum pool delete <poolid>

删除资源池。

<poolid>: <string>
无可用描述

pveum pool list [OPTIONS] [FORMAT_OPTIONS]

列出资源池或获取资源池配置。

--poolid <string>
无可用描述

--type <lxc | qemu | storage>
无可用描述

> **Note**
> 需要选项：poolid

pveum pool modify <poolid> [OPTIONS]

更新资源池。

<poolid>: <string>
无可用描述

--allow-move <boolean> (*default* = 0)
允许添加已在其他资源池中的客户机。该客户机会从当前资源池中移除，并添加到此资源池。

--comment <string>
无可用描述

--delete <boolean> (*default* = 0)
移除传入的 VMID 和/或存储 ID，而不是添加它们。

--storage <string>
要从此资源池添加或移除的存储 ID 列表。

--vms <string>
要从此资源池添加或移除的客户机 VMID 列表。

pveum realm add <realm> --type <string> [OPTIONS]

添加认证服务器。

<realm>: <string>
认证域 ID

--acr-values ^[^\x00-\x1F\x7F <>#”]*$
指定请求 Authorization Server 在 Auth Request 中使用的 Authentication Context Class Reference 值。

--autocreate <boolean> (*default* = 0)
如果用户不存在，则自动创建用户。

--base_dn <string>
LDAP 基础域名

--bind_dn <string>
LDAP 绑定域名

--capath <string> (*default* = /etc/ssl/certs)
CA 证书存储路径

--case-sensitive <boolean> (*default* = 1)
用户名区分大小写

--cert <string>
客户端证书路径

--certkey <string>
客户端证书密钥路径

--check-connection <boolean> (*default* = 0)
检查到服务器的 bind 连接。

--client-id <string>
OpenID Client ID

--client-key <string>
OpenID Client Key

--comment <string>
描述。

--default <boolean>
将其用作默认 realm

--domain \S+
AD 域名

--filter <string>
用于用户同步的 LDAP 过滤器。

--group_classes <string> (*default* = groupOfNames, group, univentionGroup, ipausergroup)
组的 objectclasses。

--group_dn <string>
用于组同步的 LDAP 基础域名。如果未设置，将使用 base_dn。

--group_filter <string>
用于组同步的 LDAP 过滤器。

--group_name_attr <string>
表示组名的 LDAP 属性。如果未设置或未找到，将使用 DN 的第一个值作为名称。

--groups-autocreate <boolean> (*default* = 0)
如果组不存在，则自动创建组。

--groups-claim (?^:[A-Za-z0-9\.\-_]+)
用于获取组的 OpenID claim。

--groups-overwrite <boolean> (*default* = 0)
登录时会覆盖该用户的所有组。

--issuer-url <string>
OpenID Issuer URL

--mode <ldap | ldap+starttls | ldaps> (*default* = ldap)
LDAP 协议模式。

--password <string>
LDAP bind 密码。将存储在 */etc/pve/priv/realm/<REALM>.pw*。

--port <integer> (1 - 65535)
服务器端口。

--prompt (?:none|login|consent|select_account|\S+)
指定 Authorization Server 是否提示 End-User 重新认证并同意授权。

--query-userinfo <boolean> (*default* = 1)
启用查询 userinfo 端点以获取 claim 值。

--scopes <string> (*default* = email profile)
指定应授权并返回的 scopes（用户详情），例如 *email* 或 *profile*。

--secure <boolean>
使用安全的 LDAPS 协议。已弃用：请改用 *mode*。

--server1 <string>
服务器 IP 地址（或 DNS 名称）

--server2 <string>
备用服务器 IP 地址（或 DNS 名称）

--sslversion <tlsv1 | tlsv1_1 | tlsv1_2 | tlsv1_3>
LDAPS TLS/SSL 版本。不建议使用低于 1.2 的版本！

**--sync-defaults-options [enable-new=<1|0>] [,full=<1|0>]
[,purge=<1|0>]
[,remove-vanished=([acl];[properties];[entry])|none]
[,scope=<users|groups|both>]**
同步行为的默认选项。

--sync_attributes \w+=[^,]+(,\s*\w+=[^,]+)*
逗号分隔的 key=value 对列表，用于指定哪些 LDAP 属性映射到哪些 PVE 用户字段。例如，要将 LDAP 属性 *mail* 映射到 PVE 的 *email*，请写作 *email=mail*。默认情况下，每个 PVE 用户字段都由同名 LDAP 属性表示。

**--tfa type=<TFATYPE> [,digits=<COUNT>] [,id=<ID>] [,key=<KEY>]
[,step=<SECONDS>] [,url=<URL>]**
使用双因素认证。

--type <ad | ldap | openid | pam | pve>
Realm 类型。

--user_attr \S{2,}
LDAP 用户属性名

--user_classes <string> (*default* = inetorgperson, posixaccount, person, user)
用户的 objectclasses。

--username-claim <string>
用于生成唯一用户名的 OpenID claim。

--verify <boolean> (*default* = 0)
验证服务器的 SSL 证书

pveum realm delete <realm>

删除认证服务器。

<realm>: <string>
认证域 ID

pveum realm list [FORMAT_OPTIONS]

认证域索引。

pveum realm modify <realm> [OPTIONS]

更新认证服务器设置。

<realm>: <string>
认证域 ID

--acr-values ^[^\x00-\x1F\x7F <>#”]*$
指定请求 Authorization Server 在 Auth Request 中使用的 Authentication Context Class Reference 值。

--autocreate <boolean> (*default* = 0)
如果用户不存在，则自动创建用户。

--base_dn <string>
LDAP 基础域名

--bind_dn <string>
LDAP 绑定域名

--capath <string> (*default* = /etc/ssl/certs)
CA 证书存储路径

--case-sensitive <boolean> (*default* = 1)
用户名区分大小写

--cert <string>
客户端证书路径

--certkey <string>
客户端证书密钥路径

--check-connection <boolean> (*default* = 0)
检查到服务器的 bind 连接。

--client-id <string>
OpenID Client ID

--client-key <string>
OpenID Client Key

--comment <string>
描述。

--default <boolean>
将其用作默认 realm

--delete <string>
要删除的设置列表。

--digest <string>
如果当前配置文件具有不同 digest，则阻止更改。可用于防止并发修改。

--domain \S+
AD 域名

--filter <string>
用于用户同步的 LDAP 过滤器。

--group_classes <string> (*default* = groupOfNames, group, univentionGroup, ipausergroup)
组的 objectclasses。

--group_dn <string>
用于组同步的 LDAP 基础域名。如果未设置，将使用 base_dn。

--group_filter <string>
用于组同步的 LDAP 过滤器。

--group_name_attr <string>
表示组名的 LDAP 属性。如果未设置或未找到，将使用 DN 的第一个值作为名称。

--groups-autocreate <boolean> (*default* = 0)
如果组不存在，则自动创建组。

--groups-claim (?^:[A-Za-z0-9\.\-_]+)
用于获取组的 OpenID claim。

--groups-overwrite <boolean> (*default* = 0)
登录时会覆盖该用户的所有组。

--issuer-url <string>
OpenID Issuer URL

--mode <ldap | ldap+starttls | ldaps> (*default* = ldap)
LDAP 协议模式。

--password <string>
LDAP bind 密码。将存储在 */etc/pve/priv/realm/<REALM>.pw*。

--port <integer> (1 - 65535)
服务器端口。

--prompt (?:none|login|consent|select_account|\S+)
指定 Authorization Server 是否提示 End-User 重新认证并同意授权。

--query-userinfo <boolean> (*default* = 1)
启用查询 userinfo 端点以获取 claim 值。

--scopes <string> (*default* = email profile)
指定应授权并返回的 scopes（用户详情），例如 *email* 或 *profile*。

--secure <boolean>
使用安全的 LDAPS 协议。已弃用：请改用 *mode*。

--server1 <string>
服务器 IP 地址（或 DNS 名称）

--server2 <string>
备用服务器 IP 地址（或 DNS 名称）

--sslversion <tlsv1 | tlsv1_1 | tlsv1_2 | tlsv1_3>
LDAPS TLS/SSL 版本。不建议使用低于 1.2 的版本！

**--sync-defaults-options [enable-new=<1|0>] [,full=<1|0>]
[,purge=<1|0>]
[,remove-vanished=([acl];[properties];[entry])|none]
[,scope=<users|groups|both>]**
同步行为的默认选项。

--sync_attributes \w+=[^,]+(,\s*\w+=[^,]+)*
逗号分隔的 key=value 对列表，用于指定哪些 LDAP 属性映射到哪些 PVE 用户字段。例如，要将 LDAP 属性 *mail* 映射到 PVE 的 *email*，请写作 *email=mail*。默认情况下，每个 PVE 用户字段都由同名 LDAP 属性表示。

**--tfa type=<TFATYPE> [,digits=<COUNT>] [,id=<ID>] [,key=<KEY>]
[,step=<SECONDS>] [,url=<URL>]**
使用双因素认证。

--user_attr \S{2,}
LDAP 用户属性名

--user_classes <string> (*default* = inetorgperson, posixaccount, person, user)
用户的 objectclasses。

--verify <boolean> (*default* = 0)
验证服务器的 SSL 证书

pveum realm sync <realm> [OPTIONS]

将配置的 LDAP 中的用户和/或组同步到 user.cfg。NOTE: 同步后的组名为 *name-$realm*，请确保这些

<realm>: <string>
认证域 ID

--dry-run <boolean> (*default* = 0)
如果设置，则不写入任何内容。

--enable-new <boolean> (*default* = 1)
立即启用新同步的用户。

--full <boolean>
已弃用：请改用 *remove-vanished*。如果设置，则使用 LDAP Directory 作为事实来源，删除同

--purge <boolean>
已弃用：请改用 *remove-vanished*。移除同步过程中从配置中删除的用户或组的 ACL。

--remove-vanished ([acl];[properties];[entry])|none (*default* = none)
分号分隔的列表，指定同步期间对应对象或用户消失时要移除的内容。可用值如下：*entry* 表示当
表示移除现有用户/组中未出现在源端的已设置属性（包括自定义属性）。*acl* 表示当同步结果未返
acl。也可以使用 *none*（默认值）代替列表。

--scope <both | groups | users>
选择要同步的内容。

pveum role add <roleid> [OPTIONS]

创建新角色。

<roleid>: <string>
无可用描述

--privs <string>
无可用描述

pveum role delete <roleid>

删除角色。

<roleid>: <string>
无可用描述

pveum role list [FORMAT_OPTIONS]

角色索引。

pveum role modify <roleid> [OPTIONS]

更新现有角色。

<roleid>: <string>
无可用描述

--append <boolean>
无可用描述

> **Note**
> 需要选项：privs

--privs <string>
无可用描述

pveum roleadd

pveum role add 的别名。

pveum roledel

pveum role delete 的别名。

pveum rolemod

pveum role modify 的别名。

pveum ticket <username> [OPTIONS]

创建或验证认证 ticket。

<username>: <string>
用户名

--new-format <boolean> (*default* = 1)
该参数现在会被忽略，并假定为 1。

--otp <string>
用于双因素认证的一次性密码。

--path <string>
验证 ticket，并检查用户是否在 *path* 上拥有 *privs* 访问权限

> **Note**
> 需要选项：privs

--privs <string>
验证 ticket，并检查用户是否在 *path* 上拥有 *privs* 访问权限

> **Note**
> 需要选项：path

--realm <string>
可以选择通过该参数传入 realm。通常 realm 会直接添加到用户名中，格式为 <username>@<

--tfa-challenge <string>
用户要响应的已签名 TFA challenge 字符串。

pveum user add <userid> [OPTIONS]

创建新用户。

<userid>: <string>
完整用户 ID，格式为 name@realm。

--comment <string>
无可用描述

--email <string>
无可用描述

--enable <boolean> (*default* = 1)
启用账户（默认）。可以将其设置为 *0* 以禁用账户

--expire <integer> (0 - N)
账户过期日期（自 epoch 起的秒数）。*0* 表示无过期日期。

--firstname <string>
无可用描述

--groups <string>
无可用描述

--keys [0-9a-zA-Z!=]{0,4096}
双因素认证密钥（yubico）。

--lastname <string>
无可用描述

--password <string>
初始密码。

pveum user delete <userid>

删除用户。

<userid>: <string>
完整用户 ID，格式为 name@realm。

pveum user list [OPTIONS] [FORMAT_OPTIONS]

用户索引。

--enabled <boolean>
按 enable 属性进行可选过滤。

--full <boolean> (*default* = 0)
包含组和 token 信息。

pveum user modify <userid> [OPTIONS]

更新用户配置。

<userid>: <string>
完整用户 ID，格式为 name@realm。

--append <boolean>
无可用描述

> **Note**
> 需要选项：groups

--comment <string>
无可用描述

--email <string>
无可用描述

--enable <boolean> (*default* = 1)
启用账户（默认）。可以将其设置为 *0* 以禁用账户

--expire <integer> (0 - N)
账户过期日期（自 epoch 起的秒数）。*0* 表示无过期日期。

--firstname <string>
无可用描述

--groups <string>
无可用描述

--keys [0-9a-zA-Z!=]{0,4096}
双因素认证密钥（yubico）。

--lastname <string>
无可用描述

pveum user permissions [<userid>] [OPTIONS] [FORMAT_OPTIONS]

获取指定用户/token 的有效权限。

<userid>: (?^:^(?^:[^\s:/]+)\@(?^:[A-Za-z][A-Za-z0-9\.\-_]+)(?:!(?^:[A-Za-z][A-Za-z0-

用户 ID 或完整 API token ID

--path <string>
只输出该特定路径，而不是整棵树。

pveum user tfa delete <userid> [OPTIONS]

删除用户的 TFA 条目。

<userid>: <string>
完整用户 ID，格式为 name@realm。

--id <string>
TFA ID；如果未提供，则删除所有 TFA 条目。

pveum user tfa list [<userid>]

列出 TFA 条目。

<userid>: <string>
完整用户 ID，格式为 name@realm。

pveum user tfa unlock <userid>

解锁用户的 TFA 认证。

<userid>: <string>
完整用户 ID，格式为 name@realm。

pveum user token add <userid> <tokenid> [OPTIONS] [FORMAT_OPTIONS]

为特定用户生成新的 API token。NOTE: 返回 API token 值；需要保存该值，因为之后无法再次取回！

<userid>: <string>
完整用户 ID，格式为 name@realm。

<tokenid>: (?^:[A-Za-z][A-Za-z0-9\.\-_]+)
用户专属 token 标识符。

--comment <string>
无可用描述

--expire <integer> (0 - N) (*default* = same as user)
API token 过期日期（自 epoch 起的秒数）。*0* 表示无过期日期。

--privsep <boolean> (*default* = 1)
使用独立 ACL 限制 API token 权限（默认），或授予对应用户的完整权限。

pveum user token delete <userid> <tokenid> [FORMAT_OPTIONS]

移除特定用户的 API token。

<userid>: <string>
完整用户 ID，格式为 name@realm。

<tokenid>: (?^:[A-Za-z][A-Za-z0-9\.\-_]+)
用户专属 token 标识符。

pveum user token list <userid> [FORMAT_OPTIONS]

获取用户 API token。

<userid>: <string>
完整用户 ID，格式为 name@realm。

pveum user token modify <userid> <tokenid> [OPTIONS] [FORMAT_OPTIONS]

更新特定用户的 API token。

<userid>: <string>
完整用户 ID，格式为 name@realm。

<tokenid>: (?^:[A-Za-z][A-Za-z0-9\.\-_]+)
用户专属 token 标识符。

--comment <string>
无可用描述

--expire <integer> (0 - N) (*default* = same as user)
API token 过期日期（自 epoch 起的秒数）。*0* 表示无过期日期。

--privsep <boolean> (*default* = 1)
使用独立 ACL 限制 API token 权限（默认），或授予对应用户的完整权限。

pveum user token permissions <userid> <tokenid> [OPTIONS] [FORMAT_OPTIONS]

获取指定 token 的有效权限。

<userid>: <string>
完整用户 ID，格式为 name@realm。

<tokenid>: (?^:[A-Za-z][A-Za-z0-9\.\-_]+)
用户专属 token 标识符。

--path <string>
只输出该特定路径，而不是整棵树。

pveum user token remove

pveum user token delete 的别名。

pveum useradd

pveum user add 的别名。

pveum userdel

pveum user delete 的别名。

pveum usermod

pveum user modify 的别名。

A.17 vzdump - 虚拟机和容器备份工具

vzdump help

vzdump {<vmid>} [OPTIONS]

创建备份。

<vmid>: <string>
要备份的客户系统 ID。

--all <boolean> (*default* = 0)
备份此主机上所有已知的客户系统。

--bwlimit <integer> (0 - N) (*default* = 0)
限制 I/O 带宽（单位为 KiB/s）。

--compress <0 | 1 | gzip | lzo | zstd> (*default* = 0)
压缩转储文件。

--dumpdir <string>
将生成的文件存储到指定目录。

--exclude <string>
排除指定的客户系统（假定使用 --all）

--exclude-path <array>
排除特定文件/目录（shell glob）。以 / 开头的路径锚定到容器根目录，其他路径相对于每个子目

--fleecing [[enabled=]<1|0>] [,storage=<storage ID>]
backup fleecing 的选项（仅 VM）。

--ionice <integer> (0 - 8) (*default* = 7)
使用 BFQ 调度器时设置 IO 优先级。对于 VM 的 snapshot 和 suspend 模式备份，这只影响压缩
8 表示使用 idle 优先级，否则使用带指定数值的 best-effort 优先级。

--job-id \S+
备份作业的 ID。如果设置，备份通知的 *backup-job* 元数据字段将设置为该值。只有 root@pam
可以设置此参数。

--lockwait <integer> (0 - N) (*default* = 180)
等待全局锁的最长时间（分钟）。

--mailnotification <always | failure> (*default* = always)
已弃用：请改用通知目标/匹配器。指定何时发送通知邮件

--mailto <string>
已弃用：请改用通知目标/匹配器。应接收电子邮件通知的电子邮件地址或用户的逗号分隔列表。

--maxfiles <integer> (1 - N)
已弃用：请改用 *prune-backups*。每个客户系统的最大备份文件数。

--mode <snapshot | stop | suspend> (*default* = snapshot)
备份模式。

--node <string>
仅在此节点上执行时运行。

--notes-template <string>
用于为备份生成备注的模板字符串。它可以包含变量，这些变量会被相应值替换。目前支持 {\{\c
ter}}、{\{\guestname}}、{\{\node}} 和 {\{\vmid}}，未来可能会加入更多变量。必须为
\n 和 \\。

Note
需要选项：storage

--notification-mode <auto | legacy-sendmail | notification-system> (*default* = auto)
确定使用哪种通知系统。如果设置为 *legacy-sendmail*，vzdump 会考虑 mailto/mailnotification 参数，并通过 *sendmail* 命令向指定地址发送电子邮件。如果设置为 *notification-system*，将通过 PVE 的通知系统发送通知，并忽略 mailto 和 mailnotification。如果设置为 *auto*（默认设置），则在设置 mailto 时发送电子邮件，否则使用通知系统。

--notification-policy <always | failure | never> (*default* = always)
已弃用：不要使用

--notification-target <string>
已弃用：不要使用

--pbs-change-detection-mode <data | legacy | metadata>
用于检测文件变化并切换容器备份编码格式的 PBS 模式。

--performance [max-workers=<integer>] [,pbs-entries-max=<integer>]
其他与性能相关的设置。

--pigz <integer> (*default* = 0)
当 N>0 时使用 pigz 而不是 gzip。N=1 使用一半核心，N>1 使用 N 作为线程数。

--pool <string>
备份指定池中包含的所有已知客户系统。

--protected <boolean>
如果为 true，将备份标记为受保护。

> **Note**
> 需要选项：storage

--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。

--quiet <boolean> (*default* = 0)
减少输出。

--remove <boolean> (*default* = 1)
根据 *prune-backups* 清理较旧的备份。

--script <string>
使用指定的 hook 脚本。

--stdexcludes <boolean> (*default* = 1)
排除临时文件和日志。

--stdout <boolean>
将 tar 写入 stdout，而不是写入文件。

--stop <boolean> (*default* = 0)
停止此主机上正在运行的备份作业。

--stopwait <integer> (0 - N) (*default* = 10)
等待客户系统停止的最长时间（分钟）。

--storage <storage ID>
将生成的文件存储到此存储。

--tmpdir <string>
将临时文件存储到指定目录。

--zstd <integer> (*default* = 1)
Zstd 线程数。N=0 使用一半可用核心；如果 N 设置为大于 0 的值，则将 N 用作线程数。

A.17.1 常用命令示例

备份单个客户系统到默认备份位置：

```
# vzdump 100
```

查看 vzdump 帮助：

```
# vzdump help
```

A.18 ha-manager - Proxmox VE HA 管理器

ha-manager <COMMAND> [ARGS] [OPTIONS]

ha-manager add <sid> [OPTIONS]

创建新的 HA 资源。

<sid>: <type>:<name>
HA 资源 ID。它由资源类型和资源专用名称组成，两者用冒号分隔（例如：vm:100 / ct:100）。VM 或 CT ID 作为快捷写法（例如：100）。

--comment <string>
描述。

--group <string>
HA 组标识符。

--max_relocate <integer> (0 - N) (*default* = 1)
服务启动失败时，尝试重定位服务的最大次数。

--max_restart <integer> (0 - N) (*default* = 1)
服务在某个节点上启动失败后，在该节点上尝试重启服务的最大次数。

--state <disabled | enabled | ignored | started | stopped> (*default = started*)
请求的资源状态。

--type <ct | vm>
资源类型。

ha-manager config [OPTIONS]

列出 HA 资源。

--type <ct | vm>
仅列出指定类型的资源

ha-manager crm-command migrate <sid> <node>

请求将资源迁移（在线）到另一个节点。

<sid>: <type>:<name>
HA 资源 ID。它由资源类型和资源专用名称组成，两者用冒号分隔（例如：vm:100 / ct:100）。VM 或 CT ID 作为快捷写法（例如：100）。

<node>: <string>
目标节点。

ha-manager crm-command node-maintenance disable <node>

更改节点维护请求状态。

<node>: <string>
集群节点名称。

ha-manager crm-command node-maintenance enable <node>

更改节点维护请求状态。

<node>: <string>
集群节点名称。

ha-manager crm-command relocate <sid> <node>

请求将资源重定位到另一个节点。这会在原节点上停止该服务，并在目标节点上重新启动。

<sid>: <type>:<name>
HA 资源 ID。它由资源类型和资源专用名称组成，两者用冒号分隔（例如：vm:100 / ct:100）。VM 或 CT ID 作为快捷写法（例如：100）。

<node>: <string>
目标节点。

ha-manager crm-command stop <sid> <timeout>

请求停止该服务。

<sid>: <type>:<name>
HA 资源 ID。它由资源类型和资源专用名称组成，两者用冒号分隔（例如：vm:100 / ct:100）。VM 或 CT ID 作为快捷写法（例如：100）。

<timeout>: <integer> (0 - N)
超时时间，单位为秒。如果设置为 0，将执行强制停止。

ha-manager groupadd <group> --nodes <string> [OPTIONS]

创建新的 HA 组。

<group>: <string>
HA 组标识符。

--comment <string>
描述。

--nodes <node>[:<pri>]{,<node>[:<pri>]}*
集群节点名称列表，可选择附带优先级。

--nofailback <boolean> (*default* = 0)
CRM 会尝试在优先级最高的节点上运行服务。如果优先级更高的节点上线，CRM 会将服务迁移至
nofailback 可阻止这种行为。

--restricted <boolean> (*default* = 0)
绑定到受限组的资源只能在该组定义的节点上运行。

--type <group>
组类型。

ha-manager groupconfig

获取 HA 组。

ha-manager groupremove <group>

删除 HA 组配置。

<group>: <string>
HA 组标识符。

ha-manager groupset <group> [OPTIONS]

更新 HA 组配置。

<group>: <string>
HA 组标识符。

--comment <string>
描述。

--delete <string>
要删除的设置列表。

--digest <string>
如果当前配置文件具有不同的摘要，则阻止更改。这可用于防止并发修改。

--nodes <node>[:<pri>]{,<node>[:<pri>]}*
集群节点名称列表，可选择附带优先级。

--nofailback <boolean> (*default* = 0)
CRM 会尝试在优先级最高的节点上运行服务。如果优先级更高的节点上线，CRM 会将服务迁移至
nofailback 可阻止这种行为。

--restricted <boolean> (*default* = 0)
绑定到受限组的资源只能在该组定义的节点上运行。

ha-manager help [OPTIONS]

获取指定命令的帮助信息。

--extra-args <array>
显示特定命令的帮助信息

--verbose <boolean>
详细输出格式。

ha-manager migrate

ha-manager crm-command migrate 的别名。

ha-manager relocate

ha-manager crm-command relocate 的别名。

ha-manager remove <sid>

删除资源配置。

<sid>: <type>:<name>
HA 资源 ID。它由资源类型和资源专用名称组成，两者用冒号分隔（例如：vm:100 / ct:100）。
VM 或 CT ID 作为快捷写法（例如：100）。

ha-manager set <sid> [OPTIONS]

更新资源配置。

<sid>: <type>:<name>
HA 资源 ID。它由资源类型和资源专用名称组成，两者用冒号分隔（例如：vm:100 / ct:100）。
VM 或 CT ID 作为快捷写法（例如：100）。

--comment <string>
描述。

--delete <string>
要删除的设置列表。

--digest <string>
如果当前配置文件具有不同的摘要，则阻止更改。这可用于防止并发修改。

--group <string>
HA 组标识符。

--max_relocate <integer> (0 - N) (*default* = 1)
服务启动失败时，尝试重定位服务的最大次数。

--max_restart <integer> (0 - N) (*default* = 1)
服务在某个节点上启动失败后，在该节点上尝试重启服务的最大次数。

--state <disabled | enabled | ignored | started | stopped> (*default* = started)
请求的资源状态。

ha-manager status [OPTIONS]

显示 HA 管理器状态。

--verbose <boolean> (*default* = 0)
详细输出。包含完整的 CRM 和 LRM 状态（JSON）。

A.18.1 常用命令示例

查看 HA 资源状态：

```
# ha-manager status
```

列出已配置的 HA 资源：

```
# ha-manager config
```

Appendix B

服务守护进程

B.1 pve-firewall - Proxmox VE 防火墙守护进程

pve-firewall <COMMAND> [ARGS] [OPTIONS]

pve-firewall compile

编译并打印防火墙规则。这对测试很有用。

pve-firewall help [OPTIONS]

获取指定命令的帮助信息。

--extra-args <array>
显示特定命令的帮助信息

--verbose <boolean>
详细输出格式。

pve-firewall localnet

打印本地网络信息。

pve-firewall restart

重启 Proxmox VE 防火墙服务。

pve-firewall simulate [OPTIONS]

模拟防火墙规则。此操作不会模拟内核的 *routing* 表，而只是简单假定从源区域到目标区域的路由可达

--dest <string>
目标 IP 地址。

--dport <integer>
目标端口。

--from (host|outside|vm\d+|ct\d+|([a-zA-Z][a-zA-Z0-9]{0,9})/(\S+)) (*default* = outside)
源区域。

--protocol (tcp|udp) (*default* = tcp)
协议。

--source <string>
源 IP 地址。

--sport <integer>
源端口。

--to (host|outside|vm\d+|ct\d+|([a-zA-Z][a-zA-Z0-9]{0,9})/(\S+)) (*default* = host)
目标区域。

--verbose <boolean> (*default* = 0)
详细输出。

pve-firewall start [OPTIONS]

启动 Proxmox VE 防火墙服务。

--debug <boolean> (*default* = 0)
调试模式 - 保持在前台运行

pve-firewall status

获取防火墙状态。

pve-firewall stop

停止 Proxmox VE 防火墙服务。注意，停止操作会主动移除所有与 Proxmox VE 相关的 iptable 规则，可能导致主机失去保护。

B.1.1 常用命令示例

查看防火墙状态：

```
# pve-firewall status
```

编译并打印当前防火墙规则用于检查：

```
# pve-firewall compile
```

B.2 pvedaemon - Proxmox VE API 守护进程

pvedaemon <COMMAND> [ARGS] [OPTIONS]

pvedaemon help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助。

--verbose <boolean>
详细输出格式。

pvedaemon restart

重启守护进程；如果尚未运行，则启动它。

pvedaemon start [OPTIONS]

启动守护进程。

--debug <boolean> (*default* = 0)
调试模式，保持在前台运行

pvedaemon status

获取守护进程状态。

pvedaemon stop

停止守护进程。

常用命令示例

```
pvedaemon status
systemctl status pvedaemon.service
```

B.3 pveproxy - Proxmox VE API 代理守护进程

pveproxy <COMMAND> [ARGS] [OPTIONS]

pveproxy help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助

--verbose <boolean>
详细输出格式。

pveproxy restart

重启守护进程（如果未运行则启动）。

pveproxy start [OPTIONS]

启动守护进程。

--debug <boolean> (*default* = 0)
调试模式 - 保持在前台运行

pveproxy status

获取守护进程状态。

pveproxy stop

停止守护进程。

常用命令示例：

```
pveproxy status
pveproxy restart
```

B.4 pvestatd - Proxmox VE 状态守护进程

pvestatd <COMMAND> [ARGS] [OPTIONS]

pvestatd help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助。

--verbose <boolean>
详细输出格式。

pvestatd restart

重启守护进程（如果未运行则启动）。

pvestatd start [OPTIONS]

启动守护进程。

--debug <boolean> (*default* = 0)
调试模式 - 保持在前台运行。

pvestatd status

获取守护进程状态。

pvestatd stop

停止守护进程。

B.5 spiceproxy - SPICE 代理服务

spiceproxy <COMMAND> [ARGS] [OPTIONS]

spiceproxy help [OPTIONS]

获取指定命令的帮助信息。

--extra-args <array>
显示特定命令的帮助信息

--verbose <boolean>
详细输出格式。

spiceproxy restart

重启守护进程（如果尚未运行则启动）。

spiceproxy start [OPTIONS]

启动守护进程。

--debug <boolean> (*default* = 0)
调试模式 - 保持在前台运行

spiceproxy status

获取守护进程状态。

spiceproxy stop

停止守护进程。

B.5.1 常用命令示例

查看 spiceproxy 状态：

```
# spiceproxy status
```

B.6 pmxcfs - Proxmox 集群文件系统

pmxcfs [OPTIONS]

帮助选项：

-h, --help
显示帮助选项

应用选项：

-d, --debug
打开调试消息

-f, --foreground
不将服务器守护进程化

-l, --local
强制本地模式（忽略 corosync.conf，强制 quorum）

该服务通常使用 systemd 工具集启动和管理。服务名为 *pve-cluster*。

常用命令示例

```
systemctl start pve-cluster

systemctl stop pve-cluster

systemctl status pve-cluster
```

B.7 pve-ha-crm - 集群资源管理器守护进程

pve-ha-crm <COMMAND> [ARGS] [OPTIONS]

pve-ha-crm help [OPTIONS]

获取指定命令的帮助信息。

--extra-args <array>
显示特定命令的帮助信息

--verbose <boolean>
详细输出格式。

pve-ha-crm start [OPTIONS]

启动守护进程。

--debug <boolean> (*default* = 0)
调试模式 - 保持在前台运行

pve-ha-crm status

获取守护进程状态。

pve-ha-crm stop

停止守护进程。

B.7.1 常用命令示例

查看 HA CRM 守护进程状态：

```
# pve-ha-crm status
```

B.8 pve-ha-lrm - 本地资源管理器守护进程

pve-ha-lrm <COMMAND> [ARGS] [OPTIONS]

pve-ha-lrm help [OPTIONS]

获取指定命令的帮助。

--extra-args <array>
显示特定命令的帮助。

--verbose <boolean>
详细输出格式。

pve-ha-lrm start [OPTIONS]

启动守护进程。

--debug <boolean> (*default* = 0)
调试模式 - 保持在前台运行。

pve-ha-lrm status

获取守护进程状态。

pve-ha-lrm stop

停止守护进程。

B.9 pvescheduler - Proxmox VE 调度器守护进程

pvescheduler <COMMAND> [ARGS] [OPTIONS]

pvescheduler help [OPTIONS]

获取指定命令的帮助信息。

--extra-args <array>
显示特定命令的帮助信息

--verbose <boolean>
详细输出格式。

pvescheduler restart

重启守护进程（如果尚未运行则启动）。

pvescheduler start [OPTIONS]

启动守护进程。

--debug <boolean> (*default* = 0)
调试模式 - 保持在前台运行

pvescheduler status

获取守护进程状态。

pvescheduler stop

停止守护进程。

B.9.1 常用命令示例

查看 pvescheduler 状态：

```
# pvescheduler status
```

Appendix C

配置文件

C.1 常规

Proxmox VE 中的大多数配置文件都位于挂载在 /etc/pve 的 共享集群文件系统上。也存在例外，例如
配置文件 /etc/vzdump.conf。

通常，配置文件中的属性派生自相关 API 端点同样使用的 JSON Schema。

C.1.1 属性名称的大小写形式

从历史上看，较长的属性（以及子属性）通常使用 snake_case，或者写成一个单词。 这很可能是因为
Proxmox VE 技术栈主要使用 Perl 编程语言开发，在 Perl 中访问使用 kebab-case 的属性需要额外的
采用了不同约定。

对于新属性，首选 kebab-case。计划为现有 snake_case 属性引入别名，并在长期 方向上，在保持
API、CLI 和正在使用的配置文件切换 为 kebab-case。

C.2 数据中心配置

文件 /etc/pve/datacenter.cfg 是 Proxmox VE 的配置文件。它包含由所有节点使用的 集群范围

C.2.1 文件格式

该文件使用简单的冒号分隔键/值格式。每一行采用以下格式：

```
OPTION: value
```

文件中的空行会被忽略，以 # 字符开头的行会被视为注释并同样被忽略。

C.2.2 选项

bwlimit: [clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>] [,move=<LIMIT>] [,restore=<LIMIT>]
为各种操作设置 I/O 带宽限制（单位为 KiB/s）。

clone=<LIMIT>
克隆磁盘时的带宽限制，单位为 KiB/s。

default=<LIMIT>
默认带宽限制，单位为 KiB/s。

migration=<LIMIT>
迁移客户机时的带宽限制，单位为 KiB/s（包括移动本地磁盘）。

move=<LIMIT>
移动磁盘时的带宽限制，单位为 KiB/s。

restore=<LIMIT>
从备份还原客户机时的带宽限制，单位为 KiB/s。

consent-text: <string>
登录前显示的同意文本。

console: <applet | html5 | vv | xtermjs>
选择默认控制台查看器。可以使用内置 Java applet（VNC；已弃用，并映射到 html5）、外部 virt-viewer 兼容应用程序（SPICE）、基于 HTML5 的 VNC 查看器（noVNC），或基于 HTML5 的控制台客户端（xtermjs）。如果所选查看器不可用（例如虚拟机未启用 SPICE），则 noVNC。

crs: [ha=<basic|static>] [,ha-rebalance-on-start=<1|0>]
集群资源调度设置。

ha=<basic | static> (*default* = basic)
配置 HA 管理器应如何选择用于启动或恢复服务的节点。使用 *basic* 时，只考虑服务数量；使 *static* 时，会考虑服务的静态 CPU 和内存配置。

ha-rebalance-on-start=<boolean> (*default* = 0)
设置后，当 HA 服务的请求状态从 stop 变为 start 时，使用 CRS 选择合适的节点。

description: <string>
数据中心说明。显示在 Web 界面的数据中心备注面板中，并作为注释保存在配置文件内。

email_from: <string>
指定发送通知时使用的发件人电子邮件地址（默认为 root@$hostname）。

fencing: <both | hardware | watchdog> (*default* = watchdog)
设置 HA 集群的 fencing 模式。hardware 模式需要在 /etc/pve/ha/fence.cfg 中配置有效的 fence 设备。使用 both 时会同时使用两种模式。

Warning
hardware 和 *both* 仍为实验性功能，且处于开发中。

ha: shutdown_policy=<enum>
集群范围的 HA 设置。

shutdown_policy=<conditional | failover | freeze | migrate> (*default* = conditional)
描述节点关机或重启时处理 HA 服务的策略。Freeze 会始终冻结关机时仍位于该节点上的服
HA 管理器恢复。Failover 不会将服务标记为冻结，因此如果关机节点未能很快重新上线（<
1 分钟），服务会恢复到其他节点。*conditional* 会根据关机类型自动选择；也就是说，重启
2 分钟后被恢复。Migrate 会在触发重启或关机时尝试将所有运行中的服务移动到其他节点。

http_proxy: http://.*
指定用于下载的外部 http 代理（示例：*http://username:password@host:port/*）。

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>
VNC 服务器的默认键盘布局。

language: <ar | ca | da | de | en | es | eu | fa | fr | he | hr | it | ja | ka | kr | nb | nl | nn | pl | pt_BR | ru | sl | sv | tr | ukr | zh_CN | zh_TW>
默认 GUI 语言。

mac_prefix: <string> (*default* = BC:24:11)
虚拟客户机自动生成 MAC 地址时使用的前缀。默认值 BC:24:11 是 IEEE 分配给 Proxmox Server Solutions GmbH 的 Organizationally Unique Identifier（OUI），用于 MAC Address Block Large（MA-L）。允许在本地网络中使用该前缀，即公众无法直接访问的网络（
LAN 或 NAT/Masquerading 环境）。

请注意，当运行多个集群，并且这些集群的虚拟客户机网络存在（部分）共享时，强烈建议扩展默认 MAC 前缀，或生成一个自定义的有效前缀，以降低 MAC 冲突概率。例如，可为每个集群在 Proxmox OUI 后追加一个不同的十六进制值，如第一个集群使用 BC:24:11:0，第二个集群使用 BC:24:1
也可以通过 VLAN 等方式在逻辑上隔离客户机网络。

+ 对于可公网访问的客户机，建议注册自己的 IEEE OUI，或与你方及托管服务提供商的网络管理员协调

max_workers: <integer> (1 - N)
定义在执行 *stopall VMs* 或 ha-manager 任务等操作时，每个节点最多启动多少个 worker。

migration: [type=]<secure|insecure> [,network=<CIDR>]
集群范围的迁移设置。

network=<CIDR>
用于迁移的（子）网络 CIDR。

type=<insecure | secure> (*default* = secure)
默认情况下，迁移流量使用 SSH 隧道加密。在安全且完全私有的网络上，可以禁用此加密以

migration_unsecure: <boolean>
默认情况下，迁移会通过 SSH 隧道保障安全。对于安全的私有网络，可以禁用该机制以加快迁移
migration 属性。

next-id: [lower=<integer>] [,upper=<integer>]
控制空闲 VMID 自动选择池的范围。

lower=<integer> (*default* = 100)
空闲 next-id API 范围的下边界，包含该值。

upper=<integer> (*default* = 1000000)
空闲 next-id API 范围的上边界，不包含该值。

notify: [fencing=<always|never>]
[,package-updates=<auto|always|never>]
[,replication=<always|never>] [,target-fencing=<TARGET>]
[,target-package-updates=<TARGET>] [,target-replication=<TARGET>]
集群范围的通知设置。

fencing=<always | never>
未使用 - 请改用数据中心通知设置。

package-updates=<always | auto | never> (*default* = auto)
已弃用：请改用数据中心通知设置。 控制每日更新任务发送通知的频率：

- *auto* 对具有有效订阅的系统每天发送一次，因为这些系统被认为已用于生产环境，因此应
- *always* 每次更新时，如果存在新的待处理更新，则发送通知。
- *never* 不对新的待处理更新发送通知。

replication=<always | never>
未使用 - 请改用数据中心通知设置。

target-fencing=<TARGET>
未使用 - 请改用数据中心通知设置。

target-package-updates=<TARGET>
未使用 - 请改用数据中心通知设置。

target-replication=<TARGET>
未使用 - 请改用数据中心通知设置。

registered-tags: <tag>[;<tag>...]
设置和删除时需要对 / 拥有 Sys.Modify 权限的标签列表。在此设置且同时出现在 *user-tag-access* 中的标签也需要 Sys.Modify。

tag-style: [case-sensitive=<1|0>]
[,color-map=<tag>:<hex-color>[:<hex-color-for-text>][;<tag>=...]]
[,ordering=<config|alphabetical>] [,shape=<enum>]
标签样式选项。

case-sensitive=<boolean> (*default* = 0)
控制在更新时过滤唯一标签是否区分大小写。

color-map=<tag>:<hex-color>[:<hex-color-for-text>][;<tag>=...]

标签的手动颜色映射（用分号分隔）。

ordering=<alphabetical | config> (*default* = alphabetical)
控制 Web 界面和 API 更新中的标签排序。

shape=<circle | dense | full | none> (*default* = circle)
Web UI 树中的标签形状。*full* 绘制完整标签。*circle* 只绘制带背景色的圆点。*dense* 只绘制一个小矩形（当每个客户机分配了大量标签时很有用）。*none* 禁用标签显示。

u2f: [appid=<APPID>] [,origin=<URL>]
u2f

appid=<APPID>
U2F AppId URL 覆盖值。默认为 origin。

origin=<URL>
U2F Origin 覆盖值。主要适用于使用单一 URL 的单节点环境。

user-tag-access: [user-allow=<enum>] [,user-allow-list=<tag>[;<tag>...]]
用户可设置标签的权限选项。

user-allow=<existing | free | list | none> (*default* = free)
控制用户可在其管理的资源（如客户机）上设置或删除哪些标签。对 / 拥有 Sys.Modify 权限的用户始终不受限制。

- *none* 不允许使用任何标签。
- *list* 允许使用 *user-allow-list* 中的标签。
- *existing* 与 list 类似，但资源上已存在的标签也可使用。
- *free* 不限制标签。

user-allow-list=<tag>[;<tag>...]
当 *user-allow* 的值为 *list* 或 *existing* 时，允许用户设置和删除的标签列表（用分号分隔）。

webauthn: [allow-subdomains=<1|0>] [,id=<DOMAINNAME>] [,origin=<URL>] [,rp=<RELYING_PARTY>]
webauthn 配置。

allow-subdomains=<boolean> (*default* = 1)
是否允许 origin 为子域名，而不是必须精确匹配 URL。

id=<DOMAINNAME>
Relying party ID。必须是不包含协议、端口或路径的域名。更改此项*会*破坏现有凭据。

origin=<URL>
站点 origin。必须是 https:// URL（或 http://localhost）。应包含用户在浏览器中
Web 界面的地址。更改此项*可能*破坏现有凭据。

rp=<RELYING_PARTY>
Relying party 名称。可使用任意文本标识符。更改此项*可能*破坏现有凭据。

Appendix D

日历事件

D.1 调度格式

Proxmox VE 提供非常灵活的调度配置。它基于 systemd 的时间日历事件格式。[1] 日历事件可用于在单
表示一个或多个时间点。

此类日历事件使用以下格式：

```
[WEEKDAY] [[YEARS-]MONTHS-DAYS] [HOURS:MINUTES[:SECONDS]]
```

此格式允许配置作业应在哪些日期运行。也可以设置一个或多个开始时间。它会告知 复制调度器作业应
晚上 10 点运行的作业：'mon,tue,wed,thu,fri 22'，也可缩写为：'mon..fri 22'。大多数合

> **Note**
> 小时使用 24 小时制格式。

为了便于使用更短的配置，可以为每个来宾设置一个或多个重复时间。它们表示复制 会在开始时间本身
8 点开始复制，并每 15 分钟重复一次直到上午 9 点，可使用：'8:00/15'

由此可见，如果未使用小时分隔符（:），该值会被解释为分钟。如果使用该分隔符， 左侧的值表示小时
* 匹配所有可能的值。

如需更多用法思路，请参见 下方的更多示例。

D.2 详细规范

weekdays

日期使用英文缩写指定：sun, mon, tue, wed, thu, fri and sat。 可以使用逗号分隔列
".." 分隔开始日和结束日来设置日期范围， 例如 mon..fri。这些格式可以混合使用。如果省略
'*'。

time-format

时间格式由小时和分钟的间隔列表组成。小时和分钟用 ':' 分隔。小时和分钟都可以是值列表或
先写小时，再写分钟。如果不需要，可以省略小时。在这种情况下，小时值假定为 '*'。有效取值
0-23，分钟 0-59。

[1] 更多信息请参见 man 7 systemd.time

D.2.1 示例：

有一些特殊值具有特定含义：

Table D.1: 特殊值

值	语法
`minutely`	`*-*-* *:*:00`
`hourly`	`*-*-* *:00:00`
`daily`	`*-*-* 00:00:00`
`weekly`	`mon *-*-* 00:00:00`
`monthly`	`*-*-01 00:00:00`
`yearly` 或 `annually`	`*-01-01 00:00:00`
`quarterly`	`*-01,04,07,10-01 00:00:00`
`semiannually` 或 `semi-annually`	`*-01,07-01 00:00:00`

Table D.2: 调度示例

调度字符串	替代写法	含义
mon,tue,wed,thu,fri	mon..fri	每个工作日 0:00
sat,sun	sat..sun	仅周末 0:00
mon,wed,fri	—	仅周一、周三和周五 0:00
12:05	12:05	每天 12:05
*/5	0/5	每五分钟
mon..wed 30/10	mon,tue,wed 30/10	周一、周二、周三每个整点后的 30、40 和 50 分钟
mon..fri 8..17,22:0/15	—	每个工作日 8:00 到 18:00 之间，以及 22:00 到 23:00 之间每 15 分钟
fri 12..13:5/20	fri 12,13:5/20	周五 12:05、12:25、12:45、13:05、1 和 13:45
12,14,16,18,20,22:5	12/2:5	每天从 12:05 到 22:05，每 2 小时
*	*/1	每分钟（最小间隔）
*-05	—	每月第 5 天
Sat *-1..7 15:00	—	每月第一个周六 15:00
2015-10-21	—	2015 年 10 月 21 日 00:00

Appendix E

QEMU vCPU 列表

E.1 简介

这是 QEMU 中定义的 AMD 和 Intel x86-64/amd64 CPU 类型列表，可追溯到 2007 年。

E.2 Intel CPU 类型

Intel processors

- *Nahelem*：第 1 代 Intel Core 处理器
- *Nahelem-IBRS (v2)*：添加 Spectre v1 防护（*+spec-ctrl*）
- *Westmere*：第 1 代 Intel Core 处理器（Xeon E7-）
- *Westmere-IBRS (v2)*：添加 Spectre v1 防护（*+spec-ctrl*）
- *SandyBridge*：第 2 代 Intel Core 处理器
- *SandyBridge-IBRS (v2)*：添加 Spectre v1 防护（*+spec-ctrl*）
- *IvyBridge*：第 3 代 Intel Core 处理器
- *IvyBridge-IBRS (v2)*：添加 Spectre v1 防护（*+spec-ctrl*）
- *Haswell*：第 4 代 Intel Core 处理器
- *Haswell-noTSX (v2)*：禁用 TSX（*-hle*, *-rtm*）
- *Haswell-IBRS (v3)*：重新添加 TSX，并添加 Spectre v1 防护（*+hle*, *+rtm*, *+spec-ctrl*）
- *Haswell-noTSX-IBRS (v4)*：禁用 TSX（*-hle*, *-rtm*）
- *Broadwell*：第 5 代 Intel Core 处理器
- *Skylake*：第 1 代 Xeon Scalable 服务器处理器

- *Skylake-IBRS (v2)*：添加 Spectre v1 防护，禁用 CLFLUSHOPT（*+spec-ctrl*, *-clflushopt*）
- *Skylake-noTSX-IBRS (v3)*：禁用 TSX（*-hle*, *-rtm*）
- *Skylake-v4*: 添加 EPT switching（*+vmx-eptp-switching*）
- *Cascadelake*: 第 2 代 Xeon Scalable 处理器
- *Cascadelake-v2*：添加 arch_capabilities msr（*+arch-capabilities*, *+rdctl-no*, *+ibrs-all*, *+skip-l1dfl-vmentry*, *+mds-no*）
- *Cascadelake-v3*：禁用 TSX（*-hle*, *-rtm*）
- *Cascadelake-v4*：添加 EPT switching（*+vmx-eptp-switching*）
- *Cascadelake-v5*：添加 XSAVES（*+xsaves*, *+vmx-xsaves*）
- *Cooperlake*：面向 4 路和 8 路服务器的第 3 代 Xeon Scalable 处理器
- *Cooperlake-v2*：添加 XSAVES（*+xsaves*, *+vmx-xsaves*）
- *Icelake*: 第 3 代 Xeon Scalable 服务器处理器
- *Icelake-v2*：禁用 TSX（*-hle*, *-rtm*）
- *Icelake-v3*：添加 arch_capabilities msr（*+arch-capabilities*, *+rdctl-no*, *+ibrs-all*, *+skip-l1dfl-vmentry*, *+mds-no*, *+pschange-mc-no*, *+taa-no*）
- *Icelake-v4*：添加缺失 flags（*+sha-ni*, *+avx512ifma*, *+rdpid*, *+fsrm*, *+vmx-rdseed-exit*, *+vmx-pml*, *+vmx-eptp-switching*）
- *Icelake-v5*：添加 XSAVES（*+xsaves*, *+vmx-xsaves*）
- *Icelake-v6*：添加 ”5-level EPT”（*+vmx-page-walk-5*）
- *SapphireRapids*：第 4 代 Xeon Scalable 服务器处理器

E.3 AMD CPU 类型

AMD processors

- *Opteron_G3*：K10
- *Opteron_G4*：Bulldozer
- *Opteron_G5*：Piledriver
- *EPYC*：第 1 代 Zen 处理器
- *EPYC-IBPB (v2)*：添加 Spectre v1 防护（*+ibpb*）
- *EPYC-v3*：添加缺失 flags（*+perfctr-core*, *+clzero*, *+xsaveerptr*, *+xsaves*）
- *EPYC-Rome*：第 2 代 Zen 处理器

- *EPYC-Rome-v2*：添加 Spectre v2、v4 防护（*+ibrs*, *+amd-ssbd*）
- *EPYC-Milan*：第 3 代 Zen 处理器
- *EPYC-Milan-v2*：添加缺失 flags（*+vaes*, *+vpclmulqdq*, *+stibp-always-on*, *+amd-psfd*, *+no-nested-data-bp*, *+lfence-always-serializing*, *+null-sel-clr-base*）

E.4 ARM64 CPU 类型

- *cortex-a35*：qemu arm64 tcg 支持
- *cortex-a53*：qemu arm64 tcg 支持
- *cortex-a57*：qemu arm64 tcg 支持
- *cortex-a72*：qemu arm64 tcg 支持
- *cortex-a76*：qemu arm64 tcg 支持
- *neoverse-n1*：qemu arm64 tcg 支持
- *neoverse-n2*：qemu arm64 tcg 支持
- *neoverse-v1*：qemu arm64 tcg 支持
- *Kunpeng-920*：qemu tcg 或 kvm 支持

E.5 riscv64 CPU 类型

- *rv64*：qemu rv64 tcg 支持

E.6 loongarch64 CPU 类型

- *la464*：qemu loongarch la464 tcg 或 kvm 支持

Appendix F

防火墙宏定义

Amanda Amanda 备份

Action	proto	dport	sport
PARAM	udp	10080	
PARAM	tcp	10080	

Auth Auth (identd) 流量

Action	proto	dport	sport
PARAM	tcp	113	

BGP Border Gateway Protocol 流量

Action	proto	dport	sport
PARAM	tcp	179	

BitTorrent BitTorrent 3.1 及更早版本的 BitTorrent 流量

Action	proto	dport	sport
PARAM	tcp	6881:6889	
PARAM	udp	6881	

BitTorrent32 BitTorrent 3.2 及更高版本的 BitTorrent 流量

Action	proto	dport	sport
PARAM	tcp	6881:6999	
PARAM	udp	6881	

CVS Concurrent Versions System pserver 流量

Action	proto	dport	sport
PARAM	tcp	2401	

Ceph Ceph 存储集群流量（Ceph Monitors、OSD 和 MDS 守护进程）

Action	proto	dport	sport
PARAM	tcp	6789	
PARAM	tcp	3300	
PARAM	tcp	6800:7300	

Citrix Citrix/ICA 流量（ICA、ICA Browser、CGP）

Action	proto	dport	sport
PARAM	tcp	1494	
PARAM	udp	1604	
PARAM	tcp	2598	

DAAP Digital Audio Access Protocol 流量（iTunes、Rythmbox 守护进程）

Action	proto	dport	sport
PARAM	tcp	3689	
PARAM	udp	3689	

DCC Distributed Checksum Clearinghouse 垃圾邮件过滤机制

Action	proto	dport	sport
PARAM	tcp	6277	

DHCPfwd 转发的 DHCP 流量

Action	proto	dport	sport
PARAM	udp	67:68	67:68

DHCPv6 DHCPv6 流量

Action	proto	dport	sport
PARAM	udp	546:547	546:547

DNS Domain Name System 流量 (udp 和 tcp)

Action	proto	dport	sport
PARAM	udp	53	
PARAM	tcp	53	

Distcc Distributed Compiler 服务

Action	proto	dport	sport
PARAM	tcp	3632	

FTP File Transfer Protocol

Action	proto	dport	sport
PARAM	tcp	21	

Finger Finger 协议 (RFC 742)

Action	proto	dport	sport
PARAM	tcp	79	

GNUnet GNUnet 安全点对点网络流量

Action	proto	dport	sport
PARAM	tcp	2086	
PARAM	udp	2086	
PARAM	tcp	1080	

Action	proto	dport	sport
PARAM	udp	1080	

GRE Generic Routing Encapsulation 隧道协议

Action	proto	dport	sport
PARAM	47		

Git Git 分布式版本控制流量

Action	proto	dport	sport
PARAM	tcp	9418	

HKP OpenPGP HTTP key server 协议流量

Action	proto	dport	sport
PARAM	tcp	11371	

HTTP Hypertext Transfer Protocol (WWW)

Action	proto	dport	sport
PARAM	tcp	80	

HTTPS 基于 SSL 的 Hypertext Transfer Protocol (WWW)

Action	proto	dport	sport
PARAM	tcp	443	

ICPV2 Internet Cache Protocol V2 (Squid) 流量

Action	proto	dport	sport
PARAM	udp	3130	

ICQ AOL Instant Messenger 流量

Action	proto	dport	sport
PARAM	tcp	5190	

IMAP Internet Message Access Protocol

Action	proto	dport	sport
PARAM	tcp	143	

IMAPS 基于 SSL 的 Internet Message Access Protocol

Action	proto	dport	sport
PARAM	tcp	993	

IPIP IPIP 封装流量

Action	proto	dport	sport
PARAM	94		

IPsec IPsec 流量

Action	proto	dport	sport
PARAM	udp	500	500
PARAM	50		

IPsecah IPsec 认证（AH）流量

Action	proto	dport	sport
PARAM	udp	500	500
PARAM	51		

IPsecnat IPsec 流量和 Nat-Traversal

Action	proto	dport	sport
PARAM	udp	500	
PARAM	udp	4500	
PARAM	50		

IRC Internet Relay Chat 流量

Action	proto	dport	sport
PARAM	tcp	6667	

Jetdirect HP Jetdirect 打印

Action	proto	dport	sport
PARAM	tcp	9100	

L2TP Layer 2 Tunneling Protocol 流量

Action	proto	dport	sport
PARAM	udp	1701	

LDAP Lightweight Directory Access Protocol 流量

Action	proto	dport	sport
PARAM	tcp	389	

LDAPS 安全 Lightweight Directory Access Protocol 流量

Action	proto	dport	sport
PARAM	tcp	636	

MDNS Multicast DNS

Action	proto	dport	sport
PARAM	udp	5353	

MSNP Microsoft Notification Protocol

Action	proto	dport	sport
PARAM	tcp	1863	

MSSQL Microsoft SQL Server

Action	proto	dport	sport
PARAM	tcp	1433	

Mail 邮件流量（SMTP、SMTPS、Submission）

Action	proto	dport	sport
PARAM	tcp	25	
PARAM	tcp	465	
PARAM	tcp	587	

Munin Munin 网络资源监控流量

Action	proto	dport	sport
PARAM	tcp	4949	

MySQL MySQL 服务器

Action	proto	dport	sport
PARAM	tcp	3306	

NNTP NNTP 流量（Usenet）。

Action	proto	dport	sport
PARAM	tcp	119	

NNTPS 加密 NNTP 流量（Usenet）

Action	proto	dport	sport
PARAM	tcp	563	

NTP Network Time Protocol（ntpd）

Action	proto	dport	sport
PARAM	udp	123	

NeighborDiscovery IPv6 邻居请求、邻居通告和路由器通告

Action	proto	dport	sport
PARAM	icmpv6	router-solicitation	
PARAM	icmpv6	router-advertisement	
PARAM	icmpv6	neighbor-solicitation	
PARAM	icmpv6	neighbor-advertisement	

OSPF OSPF 组播流量

Action	proto	dport	sport
PARAM	89		

OpenVPN OpenVPN 流量

Action	proto	dport	sport
PARAM	udp	1194	

PCA Symantec PCAnywere (tm)

Action	proto	dport	sport
PARAM	udp	5632	
PARAM	tcp	5631	

PMG Proxmox Mail Gateway Web 界面

Action	proto	dport	sport
PARAM	tcp	8006	

POP3 POP3 流量

Action	proto	dport	sport
PARAM	tcp	110	

POP3S 加密 POP3 流量

Action	proto	dport	sport
PARAM	tcp	995	

PPtP Point-to-Point Tunneling Protocol

Action	proto	dport	sport
PARAM	47		
PARAM	tcp	1723	

Ping ICMP echo request

Action	proto	dport	sport
PARAM	icmp	echo-request	

PostgreSQL PostgreSQL 服务器

Action	proto	dport	sport
PARAM	tcp	5432	

Printer Line Printer protocol 打印

Action	proto	dport	sport
PARAM	tcp	515	

RDP Microsoft Remote Desktop Protocol 流量

Action	proto	dport	sport
PARAM	tcp	3389	

RIP Routing Information Protocol (双向)

Action	proto	dport	sport
PARAM	udp	520	

RNDC BIND 远程管理协议

Action	proto	dport	sport
PARAM	tcp	953	

Razor Razor 反垃圾邮件系统

Action	proto	dport	sport
PARAM	tcp	2703	

Rdate 远程时间获取 (rdate)

Action	proto	dport	sport
PARAM	tcp	37	

Rsync Rsync 服务器

Action	proto	dport	sport
PARAM	tcp	873	

SANE SANE 网络扫描

Action	proto	dport	sport
PARAM	tcp	6566	

SMB Microsoft SMB 流量

Action	proto	dport	sport
PARAM	udp	135,445	
PARAM	udp	137:139	
PARAM	udp	1024:65535	137
PARAM	tcp	135,139,445	

SMBswat Samba Web 管理工具

Action	proto	dport	sport
PARAM	tcp	901	

SMTP Simple Mail Transfer Protocol

Action	proto	dport	sport
PARAM	tcp	25	

SMTPS 加密 Simple Mail Transfer Protocol

Action	proto	dport	sport
PARAM	tcp	465	

SNMP Simple Network Management Protocol

Action	proto	dport	sport
PARAM	udp	161:162	
PARAM	tcp	161	

SPAMD Spam Assassin SPAMD 流量

Action	proto	dport	sport
PARAM	tcp	783	

SPICEproxy Proxmox VE SPICE 显示代理流量

Action	proto	dport	sport
PARAM	tcp	3128	

SSH Secure shell 流量

Action	proto	dport	sport
PARAM	tcp	22	

SVN Subversion 服务器 (svnserve)

Action	proto	dport	sport
PARAM	tcp	3690	

SixXS SixXS IPv6 Deployment and Tunnel Broker

Action	proto	dport	sport
PARAM	tcp	3874	
PARAM	udp	3740	
PARAM	41		
PARAM	udp	5072,8374	

Squid Squid Web 代理流量

Action	proto	dport	sport
PARAM	tcp	3128	

Submission 邮件提交流量

Action	proto	dport	sport
PARAM	tcp	587	

Syslog Syslog 协议 (RFC 5424) 流量

Action	proto	dport	sport
PARAM	udp	514	
PARAM	tcp	514	

TFTP Trivial File Transfer Protocol 流量

Action	proto	dport	sport
PARAM	udp	69	

Telnet Telnet 流量

Action	proto	dport	sport
PARAM	tcp	23	

Telnets 基于 SSL 的 Telnet

Action	proto	dport	sport
PARAM	tcp	992	

Time RFC 868 Time 协议

Action	proto	dport	sport
PARAM	tcp	37	

Trcrt Traceroute 流量（最多 30 跳）

Action	proto	dport	sport
PARAM	udp	33434:33524	
PARAM	icmp	echo-request	

VNC VNC display 0 - 99 的 VNC 流量

Action	proto	dport	sport
PARAM	tcp	5900:5999	

VNCL listen 模式下从 VNC 服务器到 VNC 查看器的 VNC 流量

Action	proto	dport	sport
PARAM	tcp	5500	

Web WWW 流量（HTTP 和 HTTPS）

Action	proto	dport	sport
PARAM	tcp	80	
PARAM	tcp	443	

Webcache Web 缓存/代理流量（端口 8080）

Action	proto	dport	sport
PARAM	tcp	8080	

Webmin Webmin 流量

Action	proto	dport	sport
PARAM	tcp	10000	

Whois Whois（nicname，RFC 3912）流量

Action	proto	dport	sport
PARAM	tcp	43	

Appendix G

Markdown 入门

> Markdown is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML).
>
> — John Gruber *https://daringfireball.net/projects/markdown/*

Proxmox VE Web 界面支持使用 Markdown，在节点和虚拟客户机备注中渲染富文本格式。

Proxmox VE 支持 CommonMark，并支持 GFM（GitHub Flavoured Markdown）的大多数扩展，例如表格和任务列表。

G.1 Markdown 基础

请注意，此处仅介绍基础知识。如需更完整的资料，请在网上查找，例如 https://www.markdowngu

G.1.1 标题

```
# This is a Heading h1
## This is a Heading h2
##### This is a Heading h5
```

G.1.2 强调

使用 *text* 或 _text_ 表示强调。

使用 **text** 或 __text__ 表示加粗的重点文本。

也可以组合使用，例如：

```
_You **can** combine them_
```

G.1.3 链接

可以使用自动链接检测，例如 https://forum.proxmox.com/ 会转换为可点击链接。

也可以控制链接文本，例如：

```
Now, [the part in brackets will be the link text](https://forum.proxmox.com ↩
   /).
```

G.1.4 列表

无序列表

使用 * 或 - 创建无序列表，例如：

```
* Item 1
* Item 2
* Item 2a
* Item 2b
```

添加缩进可用于创建嵌套列表。

有序列表

```
1. Item 1
1. Item 2
1. Item 3
  1. Item 3a
  1. Item 3b
```

Note
有序列表中的数字不需要正确，最终会自动编号。

任务列表

任务列表使用空框 [ ] 表示未完成任务，使用带 X 的框表示已完成任务。

例如：

```
- [X] First task already done!
- [X] Second one too
- [ ] This one is still to-do
- [ ] So is this one
```

G.1.5 表格

表格使用管道符号 | 分隔列，并使用 - 分隔表头和表体；在该分隔行中还可以设置 文本对齐方式，使某

```
| Left columns  | Right columns |  Some  | More | Cols.| Centering Works  ↩
   Too
| -------------  ↩
   |--------------:|--------|------|------|:------------------:|
| left foo      | right foo     | First  | Row  | Here | >center<  ↩
             |
| left bar      | right bar     | Second | Row  | Here | 12345  ↩
                |
| left baz      | right baz     | Third  | Row  | Here | Test  ↩
                 |
| left zab      | right zab     | Fourth | Row  | Here | b''◢b''b''b''b' ↩
   ''◢b''b''b''b''◢b''b''b''              |
| left rab      | right rab     | And    | Last | Here | The End  ↩
              |
```

请注意，不必使用空白字符将各列严格对齐，但这样会让表格编辑更容易。

G.1.6 块引用

可以通过在行首添加 > 输入块引用，类似纯文本电子邮件中的写法。

```
> Markdown is a lightweight markup language with plain-text-formatting  ↩
   syntax,
> created in 2004 by John Gruber with Aaron Swartz.
>
>> Markdown is often used to format readme files, for writing messages in  ↩
   online discussion forums,
>> and to create rich text using a plain text editor.
```

G.1.7 代码和片段

可以使用反引号避免处理少量词语或段落。这有助于避免代码或配置片段被误解释为 Markdown。

行内代码

用单个反引号包围一行中的部分内容，可用于编写行内代码，例如：

```
This hosts IP address is `10.0.0.1`.
```

整块代码

对于跨多行的代码块，可以使用三个反引号开始和结束该代码块，例如：

常用命令示例

```
pveversion -v
pvesh get /nodes
```

```
# This is the network config I want to remember here
auto vmbr2
iface vmbr2 inet static
        address 10.0.0.1/24
        bridge-ports ens20
        bridge-stp off
        bridge-fd 0
        bridge-vlan-aware yes
        bridge-vids 2-4094

```

Appendix H

GNU 自由文档许可证

Version 1.3, 3 November 2008

Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, ↩
    Inc.
    <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document ”free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ”copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ”Document”, below, refers to any such manual or work. Any member of the public is a licensee,

and is addressed as ”you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A ”Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A ”Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ”Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ”Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ”Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ”Transparent” is called ”Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ”Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ”Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The ”publisher” means any person or entity that distributes copies of the Document to the public.

A section ”Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ”Acknowledgements”, ”Dedications”, ”Endorsements”, or ”History”.) To ”Preserve the Title” of such a section when you modify the Document means that it remains a section ”Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network

protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled ”History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled ”History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the ”History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled ”Acknowledgements” or ”Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled ”Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled ”Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled ”Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents,

unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ”History” in the various original documents, forming one section Entitled ”History”; likewise combine any sections Entitled ”Acknowledgements”, and any sections Entitled ”Dedications”. You must delete all sections Entitled ”Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ”aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of

these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled ”Acknowledgements”, ”Dedications”, or ”History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

”Massive Multiauthor Collaboration Site” (or ”MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A ”Massive Multiauthor Collaboration” (or ”MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

”CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

”Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is ”eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.