

第 9 章 Proxmox 容器管理工具

与基于全虚拟化的虚拟机技术相比，容器是一种轻量级的虚拟化技术。容器并不对操作系统进行虚拟化，而是直接调用其所处主机的操作系统。这意味着所有的容器都共享同一个主机操作系统内核，并直接访问主机的各类资源。

由于容器无需虚拟化操作系统内核，因此能有效节约 CPU 和内存资源，从而赋予容器技术相当的优势。一般情况下，容器运行时的资源消耗接近于 0，完全可以忽略不计。但是，容器技术也有不可忽视的弱点：

- 容器内只能运行基于 Linux 的操作系统，比如你无法在容器内运行 FreeBSD 或 MS Windows 系统。
- 出于安全性的考虑，对主机资源的访问需要被有效控制。通常通过 AppArmor，SecComp 过滤器或 Linux 内核的其他组件来实现。这意味着在容器内无法调用一些 Linux 内核调用。

Proxmox VE 使用 [LXC](#) 作为底层容器技术。我们把 LXC 当作底层库来调用，并利用了其众多功能特性。直接使用 LXC 相关工具的难度太高，我们将有关工具进行了包装，以“Proxmox 容器管理工具”的形式提供给用户使用，该工具的名字是 `pct`。

容器管理工具 `pct` 和 Proxmox VE 紧密集成在一起，不仅能够感知 Proxmox VE 集群环境，而且能够象虚拟机那样直接利用 Proxmox VE 的网络资源和存储资源。你甚至可以在容器上配置使用 Proxmox VE 防火墙和 HA 高可用性。

我们的主要目标是提供一个和虚拟机一样的容器运行环境，同时能避免不必要的代价。我们称之为“系统容器”。

➤ 注意

如果你想运行微容器（如 `docker`，`rkt` 等），最好是在虚拟机里面运行它。

9.1 技术概览

- LXC (<https://linuxcontainers.org>)
- 集成在 Proxmox VE 图形界面中 (GUI)
- 简单易用的命令行工具 pct
- 支持通过 Proxmox VE REST API 调用
- 通过 lxcfs 提供容器化的 /proc 文件系统
- 基于 AppArmor/Seccomp 的安全性增强
- 基于 CRIU 的在线迁移 (计划中)
- 基于最新的 Linux 内核 (4.4.X)
- 基于镜像的部署 (模板)
- 可直接使用 Proxmox VE 存储服务
- 基于主机的容器配置 (网络、DNS、存储等)

9.2 安全性分析

由于容器直接使用主机 Linux 内核，所以恶意用户可利用的攻击面非常宽泛。如果你计划向不可信的用户提供容器服务，必须认真考虑该问题。一般来说，基于全虚拟化的虚拟机能够达到更好的隔离效果。

好消息是，LXC 利用了 Linux 内核的众多安全特性，例如 AppArmor、CGroups 以及 PID 和用户 namespaces，这大大改善了容器的使用安全性。

9.3 用户操作系统配置

我们通常会尝试检测容器中的操作系统类型，然后修改容器中的一些文件，以确保容器正常工作。以下是我们在容器启动时的例行操作清单：

设置/etc/hostname

设置容器名称

修改/etc/hosts

允许查找容器主机名

网络配置

向容器传递完整的网络配置信息

配置 DNS

向容器传递 DNS 服务器配置信息

调整 init 系统初始化服务

例如，修改 getty 进程数量

设置 root 口令

创建新容器时，修改 root 口令

重新生成 ssh_host_keys

以确保每个容器的 key 都不重复

随机化 crontab

以确保各容器的 cron 调度任务不会同时启动

Proxmox VE 会用如下注释行将修改内容标识出来

```
# --- BEGIN PVE ---
```

```
<data>
```

```
# --- END PVE ---
```

以上标识符会插入相关文件的合适位置。如果配置文件中已经有标识符，Proxmox VE 会更新相关配置，并不再修改原标识符位置。

可以在配置文件相同路径下创建一个.pve-ignore 文件，避免 Proxmox VE 修改该配置文件。例如，只要/etc/.pve-ignore.hosts 文件存在，Proxmox VE 就不会修改 /etc/hosts 文件配置内容。用户用如下命令创建空文件即可：

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

由于大部分配置修改都和操作系统类型相关，因此配置内容随 Linux 发行版和版本号改变而不同。你可以将 ostype 设置为 unmanaged 彻底禁止 Proxmox VE 修改配置。

操作系统类型检测通过测试容器内特定文件内容而实现：

Ubuntu

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

Debian

检测/etc/debian_version

Fedora

检测/etc/fedora-release

RedHat or CentOS

检测/etc/redhat-release

ArchLinux

检测/etc/arch-release

Alpine

检测/etc/alpine-release

Gentoo

检测/etc/gentoo-release

9.4 容器镜像

容器镜像，也称为“模板”或“应用程序”，实际上一个一个 tar 文件，其中包含了运行容器所需要的所有文件。你可以将它理解为容器的一个备份。和当前大多数容器管理工具一样，pct 使用容器镜像创建容器，例如：

```
pct create 999 local:vztmpl/debian-8.0-standard_8.0-1_amd64.tar.gz
```

Proxmox VE 自带了一组最基本的容器镜像，涵盖了大部分常见操作系统，你也可以运行命令行工具 pveam (Proxmox VE Appliance Manager 的缩写) 下载这些镜像。你还可以用该工具 (或通过 WebGUI 界面) 下载 [TurnKey Linux](#) 的容器镜像。

我们的镜像源有一组可用镜像，Proxmox VE 默认设置有一个 cron 任务每天定时下载该镜像列表。你也可以手工触发升级：

```
pveam update
```

然后可以查看可用镜像列表：

```
pveam available
```

该列表包含了很多镜像，你可以指定查看感兴趣的小节，例如可以指定查看 system 类的镜像：

列出可用镜像：

```
# pveam available --section system
system archlinux-base_2015-24-29-1_x86_64.tar.gz
system centos-7-default_20160205_amd64.tar.xz
system debian-6.0-standard_6.0-7_amd64.tar.gz
system debian-7.0-standard_7.0-3_amd64.tar.gz
system debian-8.0-standard_8.0-1_amd64.tar.gz
system ubuntu-12.04-standard_12.04-1_amd64.tar.gz
system ubuntu-14.04-standard_14.04-1_amd64.tar.gz
system ubuntu-15.04-standard_15.04-1_amd64.tar.gz
system ubuntu-15.10-standard_15.10-1_amd64.tar.gz
```

你需要将镜像下载到 Proxmox VE 的存储中之后才可以使用它创建容器。最直接的方法就是下载到 local 存储服务中。在 Proxmox VE 集群环境中，建议将镜像保存在一个共享存储服务中，以便所有节点访问。

```
pveam download local debian-8.0-standard_8.0-1_amd64.tar.gz
```

现在就可以使用该镜像创建容器了。也可以用如下命令查看 local 上已下载的所有镜像：

```
# pveam list local
local:vztmpl/debian-8.0-standard_8.0-1_amd64.tar.gz 190.20MB
```

以上命令列出了保存镜像的 Proxmox VE 存储卷的完整标识符。其中包含有存储服务名称，其他 Proxmox VE 命令可以直接使用该存储服务名。例如可以用如下命令删除镜像：

```
pveam remove local:vztmpl/debian-8.0-standard_8.0-1_amd64.tar.gz
```

9.5 容器存储

传统容器的存储模型十分简陋，通常只支持一个挂载点，即根文件系统。此外，支持的文件系统类型也非常有限，例如 ext4 和 nfs，这大大限制了容器的使用。如果要挂载额外的存储，用户必须自行编写脚本文件，既复杂又容易出错。因此，我们在 Proxmox VE 中努力避免这些问题。

我们提供的基于 LXC 的容器模型在存储方面非常灵活。首先，你可以使用多个挂载点，从而可以根据应用类型的不同选择合适的存储。例如，你可以为根文件系统选用速度相对较慢（因此也较便宜）的存储，同时为数据库应用挂载第二个高速高性能分布式存储。进一步信息可查看[挂载点](#)一节。

第二个重大改进是，你可以将 Proxmox VE 支持的任意类型的存储服务用于容器。也就是说，你可以将容器保存在本地 lvmthin 或 zfs 上，共享 iSCSI 存储或 ceph 分布式存储服务上。进一步，还可以利用存储服务的高级特性，比如快照和克隆。vzdump 也可以利用快照特性实现一致的容器备份。

最后，但也很重要，你可以在容器中直接使用服务器本地设备，或通过绑定挂载使用服务器本地目录。这样就可以在容器中以零性能损失访问本地存储服务。而绑定挂载也提供了一种在多个容器间共享数据的简便方式。

9.5.1 FUSE 挂载

☒ 警告

鉴于当前 Linux 内核的冻结子系统的问题，而以挂起或快照模式备份时需要将容器冻结，所以强烈反对在容器中使用 FUSE 挂载。

如果实在不能用其他挂载机制或存储技术替代 FUSE 挂载，万不得已时，仍然可以在 Proxmox 服务器创建 FUSE 挂载，并通过绑定挂载提供给容器使用。

9.5.2 容器内设置存储配额

在容器内设置存储配额能够有效限制每个用户可以使用的硬盘空间大小。目前，该功能仅在基于 ext4 文件系统的容器上可以使用，并且不能用于非特权容器。

激活 quota 选项后，挂载点会增加以下挂载参数项：srjquota=aquota.user,grpjquota=aquota.group,jqfmt=vfsv0

这些参数让你能够像在其他系统上一样使用存储配额功能。你可以用如下命令初始化 /aquota.user 和 /aquota.group 文件：

```
quotacheck -cmug /
```

```
quotaon /
```

然后可以通过 edquota 命令编辑配额。具体配置可以参考容器所采用的 Linux 发行版镜像自带的技术文档。

➤ 注意

你需要对每个挂载点执行以上命令，并用实际挂载点路径替代根路径 /。

9.5.3 容器内设置访问控制列表

容器内可以配置使用标准 Posix Access Control Lists。通过 ACLs 能够详尽地控制文件属主，其细致程度远胜传统的 user/group/others 模型。

9.6 容器设置项

9.6.1 通用设置项

Create: LXC Container

General Template Root Disk CPU Memory Network DNS Confirm

Node: elsa Resource Pool:
VM ID: 100 Password:
Hostname: container1 Confirm password:
Unprivileged container: ☐ SSH public key:
[Load SSH Key File](#)

Back Next

容器通用设置项如下：

- Node：容器所处的物理服务器
- CT ID：Proxmox VE 用于标识容器的唯一序号。
- Hostname：容器的主机名。
- Resource Pool：逻辑上划分的一组容器和虚拟机。
- Password：容器的 root 口令。
- SSH Public Key：用于 SSH 连接登录 root 用户的公钥。
- Unprivileged container：该项目用于在容器创建阶段设置创建特权容器或非特权容器。

特权容器

增强容器安全性的常用手段有关闭不必要的功能，使用强制访问控制（AppArmor），SecComp 过滤器和命名空间 namespaces。LXC 工作组认为特权容器技术是不安全的，并且不打算为新发现的容器逃逸漏洞申报 CVE 编号和发布快速修复补丁。所以，特权容器仅限于在内部可信环境中使用，或在确保容器中没有不可信 root 权限进程时使用。

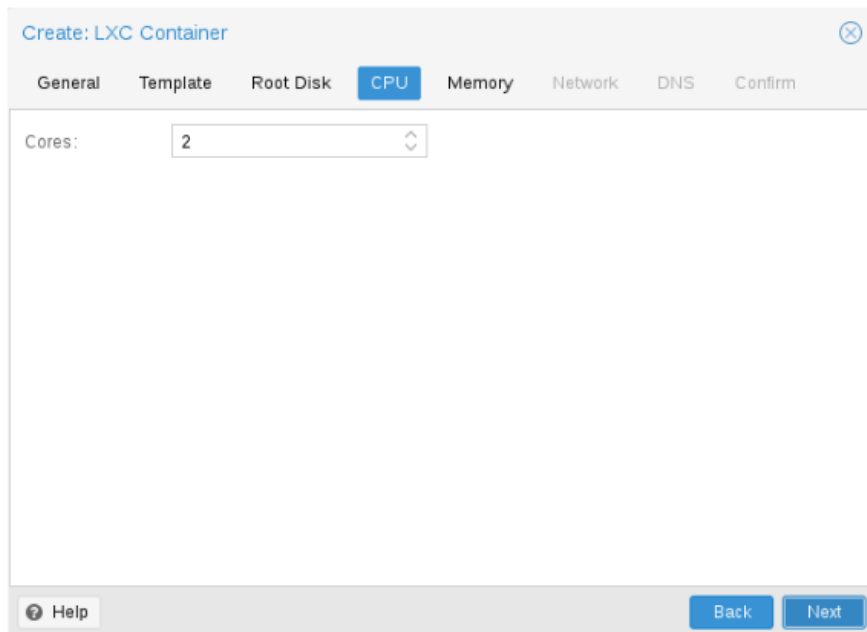
非特权容器

非特权容器采用了名为用户命名空间 user namespaces 的内核新特性。容器内 UID 为 0 的 root 用户被影射至外部的一个非特权用户。也就是说，在非特权容器中常见的安全问题（容器逃逸，资源滥用等）最终只能影响到外部的一个随机的非特权用户，并最终归结为一个一般的内核安全缺陷，而非 LXC 容器安全性问题。LXC 工作组认为非特权容器的设计是安全的。

➤ 注意

如果容器使用 systemd 作为系统初始化服务，请确保容器内的 systemd 版本高于或等于 220。

9.6.2 CPU



The screenshot shows a 'Create: LXC Container' window with a 'CPU' tab selected. The 'Cores' field is set to 2. The window has a title bar with a close button. Below the tabs, there is a large empty area. At the bottom, there are buttons for 'Help', 'Back', and 'Next'.

你可以通过 cores 参数项设置容器内可见的 CPU 数量。该参数项基于 Linux 的 cpuset cgroup（控制 group）实现。在 pvestatd 服务内有一个任务专门用于在 CPU 之间平衡容器工作负载。你可以用如下命令查看 CPU 分配情况：

```
# pct cpusets
```

```
-----
```

```
102: 6 7
```

```
105: 2 3 4 5
```

108: 0 1

容器能够直接调用主机内核，所以容器内的所有任务进程都由主机的 CPU 调度器直接管理。Proxmox VE 默认使用 Linux CFS（完全公平调度器），并提供以下参数项可以进一步控制 CPU 分配。

cpulimit： 你可以设置该参数项进一步控制分配的 CPU 时间片。需要注意，该参数项类型为浮点数，因此你可以完美地实现这样的配置效果，即给容器分配 2 个 CPU 核心，同时限制容器总的 CPU 占用为 0.5 个 CPU 核心。具体如下：

cores: 2

cpulimit: 0.5

cpuunits： 该参数项是传递给内核调度器的一个相对权重值。参数值越大，容器得到的 CPU 时间越多。具体得到的 CPU 时间片由当前容器权重占全部容器权重总和的比重决定。该参数默认值为 1024，可以调大该参数以提高容器的优先权。

9.6.3 内存



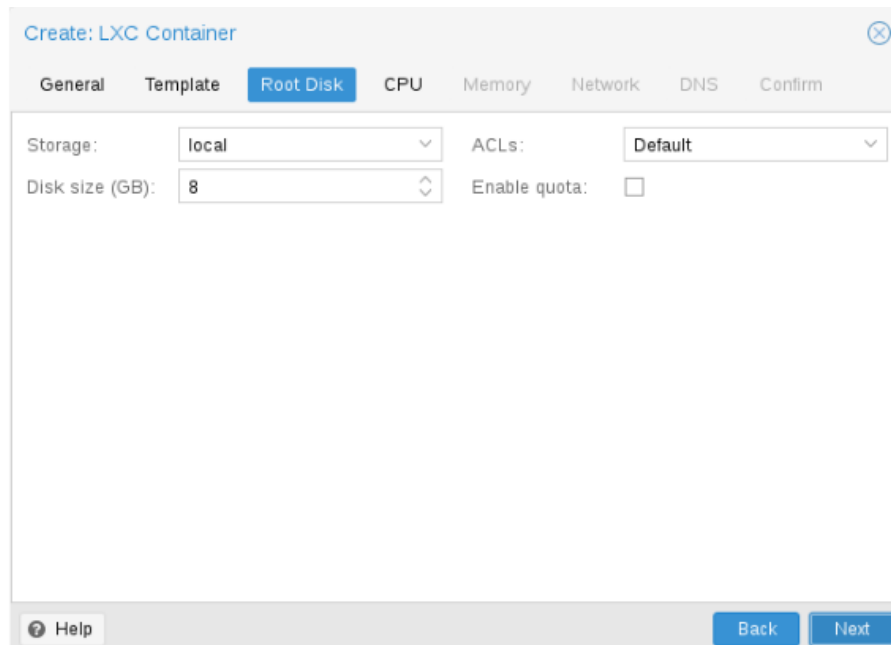
The screenshot shows the 'Create: LXC Container' window in Proxmox VE. The 'Memory' tab is selected, showing two input fields: 'Memory (MB)' and 'Swap (MB)', both set to 512. The window has a title bar with a close button. Below the tabs, there is a 'Help' button and 'Back' and 'Next' buttons at the bottom right.

容器内存由 cgroup 内存控制器管理。

memory： 容器的内存总占用量上限。对应于 cgroup 的 `memory.limit_in_bytes` 参数项。

swap : 用于设置允许容器使用主机 swap 空间的大小。对应于 cgroup 的 memory.memsw.limit_in_bytes 参数项，cgroup 的参数实际上是内存和交换分区容量之和（memory+swap）。

9.6.4 挂载点



容器的根挂载点通过 rootfs 属性配置。除此之外，你还可以再配置 10 个挂载点，分别对应于参数 mp0 到 mp9。具体设置项目如下：

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

配置容器根文件系统存储卷。全部配置参数见后续内容。

- mp[n]: [volume=<volume> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>] [,quota=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskSize>]

配置容器附加挂载点存储卷。

acl=<boolean>

启用/禁用 acl。

backup=<boolean>

用于配置在备份容器时是否将挂载点纳入备份范围。（仅限于附加挂载点）

mp=<Path>

存储卷在容器内部的挂载点路径。

➤ 注意

出于安全性考虑，禁止含有文件链接。

quota=<boolean>

在容器内启用用户空间配额（对基于 zfs 子卷的存储卷无效）。

ro=<boolean>

用于标识只读挂载点。

shared=<boolean> (default = 0)

用于标识当前存储卷挂载点对所有节点可见。

☒ 警告

设置该参数不等于自动共享挂载点，而仅仅表示当前挂载点被假定已经共享。

size=<DiskSize>

存储卷容量（参数值只读）。

volume=<volume>

存储卷命令，即挂载到容器的设备或文件系统路径。

目前主要有 3 类不同的挂载：基于存储服务的挂载，绑定挂载，设备挂载。

容器 rootfs 典型配置示例

rootfs: thin1:base-100-disk-1,size=8G

基于存储服务的挂载

基于存储服务的挂载由 Proxmox VE 的存储子系统管理，一共有 3 种不同方式：

- 硬盘镜像：也就是内建了 ext4 文件系统的硬盘镜像。

- ZFS 存储卷：技术上类似于绑定挂载，但通过 Proxmox VE 存储子系统管理，并且支持容量扩充和快照功能。
- 目录：可以设置 size=0 禁止创建硬盘镜像，直接创建目录存储。

绑定挂载

绑定挂载可以将 Proxmox VE 主机上的任意目录挂载到容器使用。可行的使用方法有：

- 在容器中访问主机目录
- 在容器中访问主机挂载的 USB 设备
- 在容器中访问主机挂载的 NFS 存储

绑定挂载并不由 Proxmox VE 存储子系统管理，因此你不能创建快照或在容器内启用配额管理。在非特权容器内，你可能会因为用户映射关系和不能配置 ACL 而遇到权限问题。

➤ 注意

vzdump 将不会备份绑定挂载设备上的数据。

☒ 警告

出于安全性考虑，最好为绑定挂载创建专门的源目录路径，例如在/mnt/bindmounts下创建的目录。永远不要将/，/var 或/etc 等系统目录直接绑定挂载给容器使用，否则将可能带来极大的安全风险。

➤ 注意

绑定挂载的源路径必须没有任何链接文件。

例如，要将主机目录/mnt/bindmounts/shared 挂载到 ID 为 100 的容器中的/shared 下，可在配置文件/etc/pve/lxc/100.conf 中增加一行配置信息 p0:/mnt/bindmounts/shared,mp=/shared。或者运行命令 pct set 100 -mp0 /mnt/bindmounts/shared,mp=/shared 也可以达到同样效果。

设备挂载

设备挂载可以将 Proxmox VE 上的块存储设备直接挂载到容器中使用。和绑定挂载类似，设备挂载也不由 Proxmox VE 存储子系统管理，但用户仍然可以配置使用 quota 和 acl 等功能。

➤ 注意

设备挂载仅在非常特殊的场景下才值得使用，大部分情况下，基于存储服务的挂载能提供和设备挂载几乎一样的功能和性能，同时还提供更多的功能特性。

➤ 注意

vzdump 将不会备份设备挂载上的数据。

9.6.5 网络

The screenshot shows the 'Create: LXC Container' window with the 'Network' tab selected. The configuration for the network interface 'eth0' is as follows:

| Field | Value |
|--------------------|--|
| Name (i.e. eth0): | eth0 |
| MAC address: | auto |
| Bridge: | vmbr0 |
| VLAN Tag: | no VLAN |
| Rate limit (MB/s): | unlimited |
| Firewall: | <input type="checkbox"/> |
| IPv4: | <input checked="" type="radio"/> Static <input type="radio"/> DHCP |
| IPv4/CIDR: | |
| Gateway (IPv4): | |
| IPv6: | <input checked="" type="radio"/> Static <input type="radio"/> DHCP <input type="radio"/> SLAAC |
| IPv6/CIDR: | |
| Gateway (IPv6): | |

单个容器最多支持配置 10 个虚拟网卡设备，其名称分别为 net0 到 net9，并支持以下配置参数项：

- net[n]: name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:XX>]

[,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>] [,mtu=<integer>]

[,rate=<mbps>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>]

[,type=<veth>]

为容器配置虚拟网卡设备。

bridge=<bridge>

虚拟网卡设备连接的虚拟交换机。

firewall=<boolean>

设置是否在虚拟网卡上启用防火墙策略。

gw=<GatewayIPv4>

IPv4 通信协议的默认网关。

gw6=<GatewayIPv6>

IPv6 通信协议的默认网关。

hwaddr=<XX:XX:XX:XX:XX:XX>

虚拟网卡的 MAC 地址。默认由 Proxmox VE 动态生成，但也可以手工指定，例如设置为当前 IPv6 地址的链路本地地址。（lxc.network.hwaddr）

ip=<IPv4Format/CIDR>

IPv4 地址，以 CIDR 格式表示。

ip6=<IPv6Format/CIDR>

IPv6 地址，以 CIDR 格式表示。

mtu=<integer> (64 -N)

虚拟网卡的传输单元。（lxc.network.mtu）

name=<string>

容器内可见的虚拟网卡名称。（lxc.network.name）

rate=<mbps>

虚拟网卡的传输速度。

tag=<integer> (1 -4094)

虚拟网卡的 VLAN 标签。

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

虚拟网卡允许通过的 VLAN 号。

type=<veth>

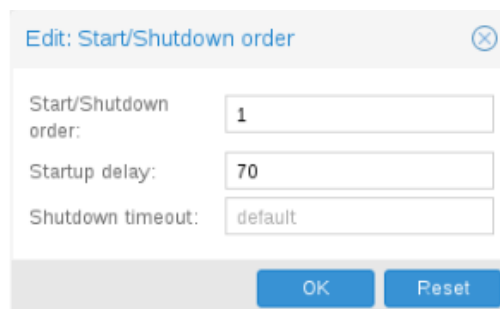
虚拟网卡类型。

9.6.6 容器的自启动和自关闭

创建容器后，你也许会希望容器能够随主机自行启动。为此，你可以在 Web 界面的容器 Options 选项卡上选择 Start at boot，或用如下命令设置：

```
pct set <ctid> -onboot 1
```

启动和关闭顺序



The screenshot shows a web interface dialog titled "Edit: Start/Shutdown order". It contains three input fields: "Start/Shutdown order:" with the value "1", "Startup delay:" with the value "70", and "Shutdown timeout:" with the value "default". At the bottom of the dialog are two buttons: "OK" and "Reset".

如果要精细调整容器的启动顺序，可以使用以下参数：

- **Start/Shutdown order**：用于设置启动优先级。例如，设为 1 表示你希望容器第 1 个被启动。（我们采用了和启动顺序相反的关闭顺序，所以 Start order 设置为 1 的容器将最后被关闭）
- **Startup delay**：用于设置当前容器启动和继续启动下一个容器之间的时间间隔。例如，设置为 240 表示你希望当前容器启动 240 秒后再继续启动下一个容器。
- **Shutdown timeout**：用于设置发出关闭命令后 Proxmox VE 等待容器执行关闭操作的时间，单位为秒。该参数默认值为 60，也就是说 Proxmox VE 在发出关闭容器命令后，会等待 60 秒钟，如果容器不能在 60 秒内完成关机离线操作，Proxmox VE 将通知用户容器关闭操作失败。

请注意，未设置 Start/Shutdown order 参数的容器将始终在设置了这些参数的容器之后启动。并且这些参数仅能影响同一 Proxmox VE 主机上的容器启动顺序，其作用范围局限在单一服务器内部，而不是整个集群。

9.7 备份和恢复

9.7.1 容器备份

可以使用 `vzdump` 命令备份容器。详细信息请参考 `vzdump` 的 man 手册。

9.7.2 容器备份恢复

可以用 `pct restore` 命令将 `vzdump` 生成的容器备份恢复出来。默认情况下，`pct restore` 将尝试尽可能按照备份文件中的配置信息恢复容器。但也可以在恢复命令中手工指定容器配置参数，以覆盖备份文件中的配置备份（详情可查看 `pct` 命令的 man 手册）。

➤ 注意

可运行命令 `pvesm extractconfig` 查看 `vzdump` 备份文件中的配置备份信息。

根据对挂载点处理方式的不同，一共有两种恢复模式：

“简单”恢复模式

如果在恢复命令中既没有指定 `rootfs` 参数也没有指定任何 `mpX` 参数，则按以下步骤从备份配置文件恢复挂载点配置信息：

- 1.从备份文件提取挂载点及相关配置项。
- 2.对于基于存储服务的挂载点，创建相应存储卷（在 `storage` 参数指定的存储服务上创建，如未设置则默认在 `local` 存储服务上创建）。
- 3.从备份文件中提取备份数据。
- 4.增加绑定挂载点和设备挂载点，并进一步恢复配置（仅限于 `root` 用户）。

基于 Web 界面的恢复操作采用的就是简单模式。

➤ 注意

鉴于绑定挂载点和设备挂载点中的数据永远不会被备份，因此最后一步中不会有任何实际数据被恢复，而仅仅是恢复挂载点配置信息。这种处理方法基于一个前提假设，即这两类挂载点中的数据也不会被其他机制备份（例如，同时绑定挂载到多个容器的 NFS 存储空间），或根本不需要备份。

“高级”恢复模式

如果指定 `rootfs` 参数（或者，指定任意 `mpX` 参数组合），恢复命令 `pct restore` 将自动进入高级恢复模式。高级恢复模式将完全忽略备份文件中保存的 `rootfs` 和 `mpX` 配置信息，转而采用命令行中指定的配置信息。

高级模式允许在恢复操作时灵活配置挂载点信息，例如：

- 为每个挂载点分别设置目标存储，存储卷容量及其他配置参数。
- 按照新指定的挂载点调整备份文件数据存储分布情况。
- 恢复到设备挂载点和（或）绑定挂载点（仅限于 root 用户）。

9.8 使用 `pct` 迁移容器

Proxmox VE 使用 `pct` 命令管理容器。你可以用 `pct` 命令创建或销毁容器，也可以控制容器的运行（启动，关闭，迁移等）。你还可以用 `pct` 命令设置相关配置文件中的参数，例如网络配置或内存上限。

9.8.1 命令行示例

使用 Debian 模板创建一个容器（假定你已经通过 Web 界面下载了模板）

```
pct create 100 /var/lib/vz/template/cache/debian-8.0-standard_8.0-1_amd64.tar.gz
```

启动 100 号容器

```
pct start 100
```

通过 `getty` 启动登录控制台

```
pct console 100
```

进入 LXC 命名空间并使用 root 用户启动一个 shell

```
pct enter 100
```

显示容器配置

```
pct config 100
```

在容器运行的状态下增加名为 eth0 的虚拟网卡，同时设置桥接虚拟交换机 vmb0，IP 地址和网关。

```
pct set 100 -net0  
name=eth0,bridge=vmb0,ip=192.168.15.147/24,gw=192.168.15.1
```

调整容器内存减少到 512MB

```
pct set 100 -memory 512
```

9.8.2 获取调试日志

在 pct start 无法启动某个容器时，运行 lxc-start 命令搜集调试日志输出有可能帮助排查故障原因（用容器 ID 替换如下命令中的 ID）：

```
lxc-start -n ID -F -l DEBUG -o /tmp/lxc-ID.log
```

该命令将尝试用前台模式启动容器，可以在另外一个控制台运行 pct shutdown ID 或 pct stop ID 停止容器。

收集到的日志信息保存在/tmp/lxc-ID.log 中。

➤ 注意

如果你在最近一次运行 pct start 命令尝试启动容器后修改了容器配置，在执行 lxc-start 命令前，你应该至少再运行一次 pct start 命令，以更新容器 lxc-start 命令可用的配置。

9.9 迁移

在集群环境中，你可以用如下命令迁移容器

```
pct migrate <vmid> <target>
```

该命令只对关机离线的容器有效。如果容器使用了本地存储和挂载点，而迁移目标服务器配置了同名的存储服务和资源，迁移命令将自动通过网络把相关数据复制到目标服务器。

如果你想迁移在线状态的容器，唯一的方法是使用重启迁移。具体是在迁移命令中使用-restart 标识和可选参数-timeout。

重启迁移命令将关闭容器，并在指定时间后杀死容器（默认时间为 180 秒），然后按照离线迁移的步骤迁移容器，并在迁移完成后在目标节点重新启动容器。

9.10 容器配置文件

容器配置信息全部保存在/etc/pve/lxc/<CTID>.conf 文件中，其中<CTID>是容器的数字 ID。和/etc/pve 目录下的所有文件一样，容器配置文件也会被自动复制到集群的所有其他节点。

➤ 注意

小于 100 的 CTID 都被 Proxmox VE 内部保留使用。同一集群内不能有重复的 CTID。

容器配置文件示例

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

容器配置文件采用了简单的文本格式，可以用常见的编辑器（vi，nano 等）编辑修改。这也是日常进行少量配置调整的常用方法，但务必注意必须重启容器才能让新的配置生效。

因此，更好的方法是使用 `pct` 命令修改配置，或通过 WebGUI 进行。Proxmox VE 能让大部分配置变更对运行中的容器即时生效。该功能称为“热插拔”，并且无须重启容器。

9.10.1 配置文件格式

容器配置文件采用了简单的冒号分隔的键/值格式。每一行的格式如下：

```
# this is a comment
```

```
OPTION: value
```

空行将被自动忽略，以 `#` 字符开头的行将被当作注释处理，也会被自动忽略。

可以在配置文件中直接添加底层 LXC 风格的配置，例如：

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

或

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

这些配置将被直接传递给底层 LXC 管理工具。

9.10.2 快照

当你创建一个快照后，`pct` 将在原配置文件中创建一个独立小节保存快照创建时的配置。例如，创建名为“testsnapshot”的快照后，你的配置文件会类似于下面的例子：

创建快照后的配置文件示例

```
memory: 512
```

```
swap: 512
```

```
parent: testsnaphot
```

```
...
```

```
[testsnaphot]
```

```
memory: 512
```

```
swap: 512
```

```
snaptime: 1457170803
```

```
...
```

其中 parent 和 snaptime 是和快照相关的配置属性。属性 parent 用于保存快照之间的父/子关系。属性 snaptime 用于标识快照创建时间 (Unix epoch) 。

9.10.3 参数项

- arch: <amd64 | i386> (default = amd64)

操作系统架构类型。

- cmode: <console | shell | tty> (default = tty)

控制台模式。默认情况下，控制台命令尝试打开到 tty 设备的连接。设置 cmode 为 console 后，将尝试连接到/dev/console 设备。设置 cmode 为 shell 后，将直接调用容器内的 shell (no login) 。

- console: <boolean> (default = 1)

挂接到容器的控制台设备 (/dev/console) 。

- cores: <integer> (1 -128)

分配给容器的 CPU 核心数量。默认容器可以使用全部的核心。

- cpulimit: <number> (0 -128) (default = 0)

CPU 分配限额。

➤ 注意

如计算机有 2 个 CPU，一共可分配的 CPU 时间为 2。设置为 0 表示无限制。

- cpuunits: <integer> (0 -500000) (default = 1024)

分配给容器的 CPU 权重。该参数用于内核的公平调度器。参数值越大，容器能获得的 CPU 时间片越多。获得的 CPU 时间片具体由当前容器权重和所有其他容器权重总和的比值决定。

➤ 注意

可将该参数设为 0 以禁用公平调度器。

- description: <string>

容器描述信息。仅供 Web 界面显示使用。

- hostname: <string>

容器的主机名。

- lock: <backup | migrate | rollback | snapshot>

设置锁定/解锁容器。

- memory: <integer> (16 -N) (default = 512)

分配给容器的内存容量。

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

[,quota=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskSize>]

给容器设置附加挂载点。

acl=<boolean>

设置启用或禁用 ACL。

backup=<boolean>

设置在备份容器时是否将挂载点纳入备份范围（仅对卷挂载点有效）。

mp=<Path>

容器内的挂载点路径。

➤ 注意

出于安全性考虑，禁止包含任何文件链接。

quota=<boolean>

启用容器内的用户配额功能（对基于 ZFS 子卷的挂载点无效）。

ro=<boolean>

设置挂载点为只读。

shared=<boolean> (default = 0)

设置非卷挂载点为所有节点可共享。

☒ 警告

设置该参数不等于自动共享挂载点，而仅仅表示当前挂载点被假定已经共享。

size=<DiskSize>

挂载点存储卷容量（参数值只读）。

volume=<volume>

挂载到容器的卷、设备或目录。

- nameserver: <string>

设置容器所使用的 DNS 服务器 IP 地址。如未指定 nameserver 和 searchdomain，将在创建容器时直接使用主机的相关配置。

- net[n]: name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:XX>] [,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>] [,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]

为容器配置虚拟网卡设备。

bridge=<bridge>

虚拟网卡设备连接的虚拟交换机。

firewall=<boolean>

设置是否在虚拟网卡上启用防火墙策略。

gw=<GatewayIPv4>

IPv4 通信协议的默认网关。

gw6=<GatewayIPv6>

IPv6 通信协议的默认网关。

hwaddr=<XX:XX:XX:XX:XX:XX>

虚拟网卡的 MAC 地址。默认由 Proxmox VE 动态生成，但也可以手工指定，例如设置为当前 IPv6 地址的链路本地地址。（lxc.network.hwaddr）

ip=<IPv4Format/CIDR>

IPv4 地址，以 CIDR 格式表示。

ip6=<IPv6Format/CIDR>

IPv6 地址，以 CIDR 格式表示。

mtu=<integer> (64 -N)

虚拟网卡的最大传输单元。（ lxc.network.mtu ）

name=<string>

容器内可见的虚拟网卡名称。（ lxc.network.name ）

rate=<mbps>

虚拟网卡的最大传输速度。

tag=<integer> (1 -4094)

虚拟网卡的 VLAN 标签。

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

虚拟网卡允许通过的 VLAN 号。

type=<veth>

虚拟网卡类型。

- onboot: <boolean> (default = 0)

设置容器是否随主机自动启动。

- ostype: <alpine | archlinux | centos | debian | fedora | gentoo | opensuse | ubuntu | unmanaged>

设置操作系统类型。供容器内部配置使用，并和 /usr/share/lxc/config/<ostype>.common.conf 中的 lxc 启动脚本对应。设置为 unmanaged 表示跳过操作系统相关配置。

- protection: <boolean> (default = 0)

设置容器的保护标志。设置后将禁止删除/变更容器及容器硬盘配置。

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

为容器配置根文件系统卷。

acl=<boolean>

设置启用或禁用 ACL。

quota=<boolean>

在容器内启用用户空间配额（对基于 zfs 子卷的存储卷无效）。

ro=<boolean>

用于标识只读挂载点。

shared=<boolean> (default = 0)

设置非卷挂载点为所有节点可共享。

☒ 警告

设置该参数不等于自动共享挂载点，而仅仅表示当前挂载点被假定已经共享。

size=<DiskSize>

挂载点存储卷容量（参数值只读）。

volume=<volume>

挂载到容器的卷、设备或目录。

- searchdomain: <string>

设置容器的 DNS 搜索域。如未指定 nameserver 和 searchdomain，将在创建容器时直接使用主机的相关配置。

- startup: `[order=]\d+`[,up=\d+][,down=\d+]`

启动和关闭行为设置。参数 order 为非负整数，用于定义启动顺序。关闭顺序和启动顺序相反。此外还可以设置启动延时秒数，以指定下一个虚拟机启动或关闭之前的时间间隔。

- swap: <integer> (0 -N) (default = 512)

分配给容器的 SWAP 容量，单位为 MB。

- template: <boolean> (default = 0)

启用/禁用模板。

- tty: <integer> (0 -6) (default = 2)

指定容器可用的 tty 数量。

- `unprivileged`: <boolean> (default = 0)

设置容器以非特权用户权限运行。（不要手工修改该配置）

- `unused[n]`: <string>

标识未使用的存储卷。仅供 Proxmox VE 内部使用，不要手工修改该配置。

9.11 锁

容器迁移、快照创建和备份（`vzdump`）操作会设置容器锁，以避免不恰当的并发操作。某些情况下，你需要手工移除容器锁（例如，意外断电）。

```
pct unlock <CTID>
```

⚠ 警告

执行该操作前，务必确保设置锁的操作已经停止运行。
