ZFS: Switch Legacy-Boot to Proxmox Boot Tool

Contents

Introduction

This HOWTO is meant for legacy-booted systems, with root on ZFS, installed using a Proxmox VE ISO between 5.4 and 6.3, and which are booted using grub.

You will *not* need this if any of the following points are true:

- System installed using Proxmox VE ISO 6.4 or later
- System uses UEFI to boot and was installed in UEFI mode
- System is not using ZFS as the root filesystem

Problem Description

On systems booting from GRUB legacy BIOS boot with root filesystem on ZFS, doing a `zpool upgrade` on the 'rpool' will **break boot**. For more details see #Background

Solution Overview

The system will be adapted to carry out the first boot steps from a small, separate partition with a simple FAT based file-system instead of the more complex ZFS directly.

The `EFI System Partition` (ESP), which often already exists, will be set up and used to hold that initial RAM disk (`initrd`). In the end, both legacy BIOS or UEFI can use this setup for booting.

Switching to proxmox-boot-tool from a Running Proxmox VE System

Checks

The following checks will help you to determine if you boot from ZFS directly through GRUB and thus would benefit from this how-to.

1. Check if root is on ZFS

Run the following command as root: `findmnt /`

The system has its root on ZFS, if the output says that FSTYPE is zfs. For example, if it looks like:

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

2. Check which bootloader is used

See the reference documentation section (https://pve.proxmox.com/pve-docs/chapter-sysadmin.html#sysboot_determine_bootloader_used) about how to find out which boot-loader is being used in your system.

If you use ZFS on root and the command `ls /sys/firmware/efi` outputs "No such file or directory", the chances are high that you boot from GRUB and thus would benefit from switching to proxmox-boot-tool, using the steps in this how-to.

3. Finding potential ESPs

Any partition or block device with a size of 512M or more can be used by `proxmox-boot-tool` as a target.

Systems installed using a Proxmox VE ISO newer than 5.4 already set up a second VFAT partition (for example /dev/sda2) with size 512 M. You can check the partitions with `lsblk`. For instance, here is a system with root on a RAID-Z1, installed with Proxmox VE 5.4:

```
# lsblk -o +FSTYPE
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT FSTYPE
sda      8:0    0    12G  0 disk            zfs_member
├─sda1   8:1    0  1007K  0 part            zfs_member
├─sda2   8:2    0   512M  0 part
└─sda3   8:3    0  11.5G  0 part            zfs_member
sdb      8:16   0    12G  0 disk            zfs_member
├─sdb1   8:17   0  1007K  0 part            zfs_member
├─sdb2   8:18   0   512M  0 part
└─sdb3   8:19   0  11.5G  0 part            zfs_member
sdc      8:32   0    12G  0 disk            zfs_member
├─sdc1   8:33   0  1007K  0 part            zfs_member
```

```
├─sdc2   8:34   0   512M  0 part
└─sdc3   8:35   0  11.5G  0 part            zfs_member
```

All three disks (sda, sdb and sdc) have a second partition with 512M size and no FS type (sda2, sdb2 and sdc2).

This means that in the above example, the three partitions sda2, sdb2 and sdc2 would be used with `proxmox-boot-tool` in the next steps.

Switching to proxmox-boot-tool

0. Upgrade to Proxmox VE 6.4

The support for booting a ZFS legacy-GRUB setup through the proxmox-boot-tool is only available since Proxmox VE 6.4

```
# pveversion
pve-manager/6.4-5/6c7bf5de (running kernel: 5.4.106-1-pve)
```

See the 11. How can I upgrade Proxmox VE to the next release? (https://pve.proxmox.com/pve-docs/chapter-pve-faq.html#:~:text=How%20can%20I%20upgrade%20Proxmox%20VE%20to%20the%20next%20release?) point in the documentation's FAQ, if you're unsure about how to do that.

You can check if your Proxmox VE version is recent enough for using proxmox-boot-tool by simply executing

```
# proxmox-boot-tool help
```

That should print a usage help.

1. Format the new intermediate boot devices

Hint: You could skip this step if the partition already has a `vfat` partition set up. If you never plan to boot via EFI you may still want to re-format by adding the `--force` flag to below's format command to get a clean setup without the EFI bootloader configured and taking up space.

For each 512M sized block device you found when following the section Finding potential ESPs, you will now set up a fresh, new VFAT partition using `proxmox-boot-tool format /dev/sdXY`

For the example used in the section Finding potential ESPs, you would execute:

```
# proxmox-boot-tool format /dev/sda2
# proxmox-boot-tool format /dev/sdb2
# proxmox-boot-tool format /dev/sdc2
```

NOTE: Be sure that you're passing the correct partitions; **the format command will destroy any data on the passed partition**!

2. Initialize & Add the new intermediate boot devices

Now we add the newly formatted VFAT partitions to the proxmox-boot-tool configuration using `proxmox-boot-tool init /dev/sdXY`

For the example used in the section Finding potential ESPs, you would execute:

```
# proxmox-boot-tool init /dev/sda2
# proxmox-boot-tool init /dev/sdb2
# proxmox-boot-tool init /dev/sdc2
```

`proxmox-boot-tool` may print a warning about a non-exisiting UUID like: `WARN: /dev/disk/by-uuid/E8A5-779A does not exist - clean '/etc/kernel/proxmox-boot-uuids'! - skipping`.

You can run the `clean` command in that case:

```
# proxmox-boot-tool clean
```

Which will simply remove any non-existent parition from the Proxmox boot tool configuration.

3. Verify the status

To verify that everything has been set up correctly, you can run the status command:

```
# proxmox-boot-tool status
Re-executing '/usr/sbin/proxmox-boot-tool' in new private mount namespace..
373A-957C is configured with: grub
3961-474D is configured with: grub
3C07-40DC is configured with: grub
```

In the above example, we see all three partitions from the three ZFS disks setup for "grub" only. It is totally fine if some or all disks display `uefi,grub` instead.

Following this, it should be possible to boot the systems from any of the 3 disks.

Repairing a System Stuck in the GRUB Rescue Shell

If you end up with a system stuck in the `grub rescue>` shell, the following steps should make it bootable again:

1. Boot using a Proxmox VE version 6.4 or newer ISO
2. Select `Install Proxmox VE (Debug Mode)`
3. Exit the first debug shell by typing `Ctrl + D` or `exit`
4. The second debug shell contains all the necessary binaries for the following steps
5. Import the root pool (usually named `rpool`) with an alternative mountpoint of `/mnt`:

```
zpool import -f -R /mnt rpool
```

6. Find the partition to use for `proxmox-boot-tool`, following the instructions from Finding potential ESPs

7. Bind-mount all virtual filesystems needed for running `proxmox-boot-tool`:

```
mount -o rbind /proc /mnt/proc
mount -o rbind /sys /mnt/sys
mount -o rbind /dev /mnt/dev
mount -o rbind /run /mnt/run
```

8. change root into `/mnt`

```
chroot /mnt /bin/bash
```

9. Format and initialize the partitions in the chroot - see Switching to proxmox-boot-tool
10. Exit the chroot-shell (`Ctrl + D` or `exit`) and reset the system (for example by pressing `CTRL + ALT + DEL`)
11. **Note**: The next boot can end up in an `initramfs` shell, due to the `hostid` mismatch (from importing the pool in the installer).

    If this is the case, simply import it again with using the force `-f` flag:

```
# zpool import -f rpool
```

    After the import you can just reboot.

The system should now boot successfully with the new, more robust, boot setup.

Background

Grub has a limited implementation for reading data from ZFS.

zpool Features and GRUB

- ZFS changes the on-disk format with zpool features, documented in the `zpool-features(5)` man-page (https://openzfs.github.io/openzfs-docs/man/5/zpool-features.5.html)
- Features are only added in minor version upgrades (e.g. from 0.7.x -> 0.8.x , or 0.8.x -> 2.0.x) - see https://github.com/openzfs/zfs/blob/master/RELEASES.md
- Running `zpool upgrade` on a pool *enables* the features
- READ-ONLY COMPATIBLE features (https://openzfs.github.io/openzfs-docs/man/5/zpool-features.5.html) should not cause any problems
- A new feature should only be problematic if it is *active* (e.g. setting the compression to zstd on a dataset will cause the `feature@zstd_compress` feature to become *active*)
- GRUB is not able to read data from a pool which has an incompatible (and not read-only compatible) *active* feature
- The list of features supported by grub can be seen on:

- Debian Buster Root on ZFS (https://openzfs.github.io/openzfs-docs/Getting%20Started/Debian/Debian%20Buster%20Root%20on%20ZFS.html)
- http://git.savannah.gnu.org/cgit/grub.git/tree/grub-core/fs/zfs/zfs.c#n276

To check which features are active on your `rpool` run:

```
# zpool get all rpool |grep active
```

While it usually takes active actions by an administrator to cause a system to become unbootable, these can happen by accident. Examples of actions and circumstances that will render a pool unbootable:

- Running `zpool upgrade rpool`, then setting the compression feature to use `zstd` on any dataset in `rpool`
- Setting the `dnodesize` property of any dataset on `rpool` to `auto` (or any value apart from `legacy`)
- The drivers for certain disk controllers (e.g. some HP SmartArray models) in GRUB can only read the first 2TB of the disk - combined with the nature of ZFS Copy-on-Write, this means that the system can become unbootable simply by installing a new kernel-image (which ends up after 2TB on disk).

The fragility of booting from ZFS with GRUB is the reason why recent Proxmox systems read the kernel and initrd image from a 512MB vfat partition, which is created in front of the ZFS partition (since PVE 5.4).

The kernel and initrd images are copied to the vfat partition by `proxmox-boot-tool` (before Proxmox VE 6.4, the utility was called `pve-efiboot-tool`)
