Debian Stretch Root on ZFS

George Melikov edited this page 12 days ago · 4 revisions

Caution

- This HOWTO uses a whole physical disk.
- Do not use these instructions for dual-booting.
- Backup your data. Any existing data will be lost.

System Requirements

- 64-bit Debian GNU/Linux Stretch Live CD
- 64-bit computer (amd64, a.k.a. x86_64) computer preferred

Computers that have less than 2 GiB of memory run ZFS slowly. 4 GiB of memory is recommended for normal performance in basic workloads. If you wish to use deduplication, you will need massive amounts of RAM. Enabling deduplication is a permanent change that cannot be easily reverted.

Support

If you need help, reach out to the community using the zfs-discuss mailing list or IRC at #zfsonlinux on freenode. If you have a bug report or feature requests related to this HOWTO, please file a new issue and mention @gmelikov and @rlaager.

Step 1: Prepare The Install Environment

1.1 Boot the Debian GNU/Linux Live CD. Login with the username `user` and password `live`.

1.2 Optional: Start the OpenSSH server in the Live CD environment:

If you have a second system, using SSH to access the target system can be convenient.

```
$ sudo sed -i "s/PasswordAuthentication no/PasswordAuthentication yes/g" \
      /etc/ssh/sshd_config
$ sudo service ssh restart
```

Hint: You can find your IP address with `ip addr show scope global | grep inet`. Then, from your main machine, connect with `ssh user@IP`.

1.3 Become root:

```
$ sudo -i
```

1.4 Add `contrib` archive area:

```
# echo "deb http://ftp.debian.org/debian stretch main contrib" > /etc/apt
/sources.list
# apt update
```

1.5 Install ZFS in the Live CD environment:

```
# apt install --yes debootstrap gdisk linux-headers-$(uname -r)
# apt install --yes zfs-dkms
```

Step 2: Disk Formatting

Pages 29

- Home
- Getting Started
  - ArchLinux
  - Debian
  - Fedora
  - Gentoo
  - openSUSE
  - RHEL and CentOS
  - Ubuntu
- Project and Community
  - Admin Documentation
  - FAQ
  - Mailing Lists
  - Releases
  - Signing Keys
  - Issue Tracker
  - Roadmap
- Developer Resources
  - Custom Packages
  - Building ZFS
  - Buildbot Status
  - Buildbot Options
  - OpenZFS Tracking
  - OpenZFS Patches
  - OpenZFS Exceptions
  - OpenZFS Documentation

Clone this wiki locally

https://github.com/zfsonl

2.1 If you are re-using a disk, clear it as necessary:

```
If the disk was previously used in an MD array, zero the superblock:
# apt install --yes mdadm
# mdadm --zero-superblock --force /dev/disk/by-id/scsi-SATA_disk1

Clear the partition table:
# sgdisk --zap-all /dev/disk/by-id/scsi-SATA_disk1
```

2.2 Partition your disk:

```
Run this if you need legacy (BIOS) booting:
# sgdisk -a1 -n2:34:2047  -t2:EF02 /dev/disk/by-id/scsi-SATA_disk1

Run this for UEFI booting (for use now or in the future):
# sgdisk     -n3:1M:+512M -t3:EF00 /dev/disk/by-id/scsi-SATA_disk1

Run these in all cases:
# sgdisk     -n1:0:0      -t1:BF01 /dev/disk/by-id/scsi-SATA_disk1
```

Always use the long `/dev/disk/by-id/*` aliases with ZFS. Using the `/dev/sd*` device nodes directly can cause sporadic import failures, especially on systems that have more than one storage pool.

Hints:

- `ls -la /dev/disk/by-id` will list the aliases.
- Are you doing this in a virtual machine? If your virtual disk is missing from `/dev/disk/by-id` , use `/dev/vda` if you are using KVM with virtio; otherwise, read the troubleshooting section.

2.3 Create the root pool:

```
# zpool create -o ashift=12 \
      -O atime=off -O canmount=off -O compression=lz4 -O normalization=formD \
      -O mountpoint=/ -R /mnt \
      rpool /dev/disk/by-id/scsi-SATA_disk1-part1
```

Notes:

- The use of `ashift=12` is recommended here because many drives today have 4KiB (or larger) physical sectors, even though they present 512B logical sectors. Also, a future replacement drive may have 4KiB physical sectors (in which case `ashift=12` is desirable) or 4KiB logical sectors (in which case `ashift=12` is required).
- Setting `normalization=formD` eliminates some corner cases relating to UTF-8 filename normalization. It also implies `utf8only=on` , which means that only UTF-8 filenames are allowed. If you care to support non-UTF-8 filenames, do not use this option. For a discussion of why requiring UTF-8 filenames may be a bad idea, see The problems with enforced UTF-8 only filenames.
- Make sure to include the `-part1` portion of the drive path. If you forget that, you are specifying the whole disk, which ZFS will then re-partition, and you will lose the bootloader partition(s).

Hints:

- The root pool does not have to be a single disk; it can have a mirror or raidz topology. In that case, repeat the partitioning commands for all the disks which will be part of the pool. Then, create the pool using `zpool create ... rpool mirror /dev/disk/by-id/scsi-SATA_disk1-part1 /dev/disk/by-id/scsi-SATA_disk2-part1` (or replace `mirror` with `raidz` , `raidz2` , or `raidz3` and list the partitions from additional disks). Later, install GRUB to all the disks. This is trivial for MBR booting; the UEFI equivalent is currently left as an exercise for the reader.
- The pool name is arbitrary. On systems that can automatically install to ZFS, the root pool is named `rpool` by default. If you work with multiple systems, it might be wise to use `hostname` , `hostname0` , or `hostname-1` instead.

Step 3: System Installation

3.1 Create a filesystem dataset to act as a container:

```
# zfs create -o canmount=off -o mountpoint=none rpool/ROOT
```

On Solaris systems, the root filesystem is cloned and the suffix is incremented for major system changes through `pkg image-update` or `beadm`. Similar functionality for APT is possible but currently unimplemented. Even without such a tool, it can still be used for manually created clones.

3.2 Create a filesystem dataset for the root filesystem of the Debian system:

```
# zfs create -o canmount=noauto -o mountpoint=/ rpool/ROOT/debian
# zfs mount rpool/ROOT/debian
# zpool set bootfs=rpool/ROOT/debian rpool
```

With ZFS, it is not normally necessary to use a mount command (either `mount` or `zfs mount`). This situation is an exception because of `canmount=noauto`.

3.3 Create datasets:

```
# zfs create                 -o setuid=off              rpool/home
# zfs create -o mountpoint=/root                        rpool/home/root
# zfs create -o canmount=off -o setuid=off  -o exec=off rpool/var
# zfs create -o com.sun:auto-snapshot=false             rpool/var/cache
# zfs create                                            rpool/var/log
# zfs create                                            rpool/var/spool
# zfs create -o com.sun:auto-snapshot=false -o exec=on  rpool/var/tmp

If you use /srv on this system:
# zfs create                                            rpool/srv

If this system will have games installed:
# zfs create                                            rpool/var/games

If this system will store local email in /var/mail:
# zfs create                                            rpool/var/mail

If this system will use NFS (locking):
# zfs create -o com.sun:auto-snapshot=false \
             -o mountpoint=/var/lib/nfs                 rpool/var/nfs
```

Notes: Properties are inherited, if you want to create (for example) `rpool/var/lib` you may need to set `-o exec=on` manually.

The primary goal of this dataset layout is to separate the OS from user data. This allows the root filesystem to be rolled back without rolling back user data such as logs (in `/var/log`). This will be especially important if/when a `beadm` or similar utility is integrated. Since we are creating multiple datasets anyway, it is trivial to add some restrictions (for extra security) at the same time. The `com.sun.auto-snapshot` setting is used by some ZFS snapshot utilities to exclude transient data.

3.4 Install the minimal system:

```
# chmod 1777 /mnt/var/tmp
# debootstrap stretch /mnt
# zfs set devices=off rpool
```

The `debootstrap` command leaves the new system in an unconfigured state. An alternative to using `debootstrap` is to copy the entirety of a working system into the new ZFS root.

Step 4: System Configuration

4.1 Configure the hostname (change `HOSTNAME` to the desired hostname).

```
# echo HOSTNAME > /mnt/etc/hostname

# vi /mnt/etc/hosts
Add a line:
127.0.1.1       HOSTNAME
or if the system has a real name in DNS:
127.0.1.1       FQDN HOSTNAME
```

Hint: Use `nano` if you find `vi` confusing.

4.2 Configure the network interface:

```
Find the interface name:
# ip addr show

# vi /mnt/etc/network/interfaces.d/NAME
auto NAME
iface NAME inet dhcp
```

Customize this file if the system is not a DHCP client.

4.3 Bind the virtual filesystems from the LiveCD environment to the new system and `chroot` into it:

```
# mount --rbind /dev  /mnt/dev
# mount --rbind /proc /mnt/proc
# mount --rbind /sys  /mnt/sys
# chroot /mnt /bin/bash --login
```

Note: This is using `--rbind` , not `--bind` .

4.4 Configure a basic system environment:

```
# vi /etc/apt/sources.list # Add `contrib` archive area:
deb http://ftp.debian.org/debian stretch main contrib
deb-src http://ftp.debian.org/debian stretch main contrib

# ln -s /proc/self/mounts /etc/mtab
# apt update

# apt install --yes locales
# dpkg-reconfigure locales
```

Even if you prefer a non-English system language, always ensure that `en_US.UTF-8` is available.

```
# dpkg-reconfigure tzdata

# apt install --yes gdisk linux-headers-$(uname -r) linux-image-amd64
```

4.5 Install ZFS in the chroot environment for the new system:

```
# apt install --yes zfs-dkms zfs-initramfs
```

4.6 Install GRUB

Choose one of the following options:

4.6a Install GRUB for legacy (MBR) booting

```
# apt install --yes grub-pc
```

4.6b Install GRUB for UEFI booting

```
# apt install dosfstools
# mkdosfs -F 32 -n EFI /dev/disk/by-id/scsi-SATA_disk1-part3
# mkdir /boot/efi
# echo PARTUUID=$(blkid -s PARTUUID -o value \
      /dev/disk/by-id/scsi-SATA_disk1-part3) \
      /boot/efi vfat defaults 0 1 >> /etc/fstab
# mount /boot/efi
# apt install --yes grub-efi-amd64
```

4.7 Set a root password

```
# passwd
```

Step 5: GRUB Installation

5.1 Verify that the ZFS root filesystem is recognized:

```
# grub-probe /
zfs
```

5.2 Refresh the initrd files:

```
# update-initramfs -u -k all
update-initramfs: Generating /boot/initrd.img-3.16.0-4-amd64
```

5.3 Optional (but highly recommended): Make debugging GRUB easier:

```
# vi /etc/default/grub
Remove quiet from: GRUB_CMDLINE_LINUX_DEFAULT
Uncomment: GRUB_TERMINAL=console
Save and quit.
```

Later, once the system has rebooted twice and you are sure everything is working, you can undo these changes, if desired.

5.4 Update the boot configuration:

```
# update-grub
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.16.0-4-amd64
Found initrd image: /boot/initrd.img-3.16.0-4-amd64
done
```

5.5 Install the boot loader

5.5a For legacy (MBR) booting, install GRUB to the MBR:

```
# grub-install /dev/disk/by-id/scsi-SATA_disk1
Installing for i386-pc platform.
Installation finished. No error reported.
```

Do not reboot the computer until you get exactly that result message. Note that you are installing GRUB to the whole disk, not a partition.

If you are creating a mirror, repeat the grub-install command for each disk in the pool.

5.5b For UEFI booting, install GRUB:

```
# grub-install --target=x86_64-efi --efi-directory=/boot/efi \
      --bootloader-id=debian --recheck --no-floppy
```

5.6 Verify that the ZFS module is installed:

```
# ls /boot/grub/*/zfs.mod
```

Step 6: First Boot

6.1 Snapshot the initial installation:

```
# zfs snapshot rpool/ROOT/debian@install
```

In the future, you will likely want to take snapshots before each upgrade, and remove old snapshots (including this one) at some point to save space.

6.2 Exit from the `chroot` environment back to the LiveCD environment:

```
# exit
```

6.3 Run these commands in the LiveCD environment to unmount all filesystems:

```
# mount | grep -v zfs | tac | awk '/\/mnt/ {print $3}' | xargs -i{} umount -lf {}
# zpool export rpool
```

6.4 Reboot:

```
# reboot
```

6.5 Wait for the newly installed system to boot normally. Login as root.

6.6 Create a user account:

```
# zfs create rpool/home/YOURUSERNAME
# adduser YOURUSERNAME
# cp -a /etc/skel/.[!.]* /home/YOURUSERNAME
# chown -R YOURUSERNAME:YOURUSERNAME /home/YOURUSERNAME
```

6.7 Add your user account to the default set of groups for an administrator:

```
# usermod -a -G audio,cdrom,dip,floppy,netdev,plugdev,sudo,video YOURUSERNAME
```

Step 7: Configure Swap

7.1 Create a volume dataset (zvol) for use as a swap device:

```
# zfs create -V 4G -b $(getconf PAGESIZE) -o compression=zle \
      -o logbias=throughput -o sync=always \
      -o primarycache=metadata -o secondarycache=none \
      -o com.sun:auto-snapshot=false rpool/swap
```

You can adjust the size (the `4G` part) to your needs.

The compression algorithm is set to `zle` because it is the cheapest available algorithm. As this guide recommends `ashift=12` (4 kiB blocks on disk), the common case of a 4 kiB page size means that no compression algorithm can reduce I/O. The exception is all-zero pages, which are dropped by ZFS; but some form of compression has to be enabled to get this behavior.

7.2 Configure the swap device:

Choose one of the following options. If you are going to do an encrypted home directory later, you should use encrypted swap.

7.2a Create an unencrypted (regular) swap device:

Caution: Always use long `/dev/zvol` aliases in configuration files. Never use a short `/dev/zdX` device name.

```
# mkswap -f /dev/zvol/rpool/swap
# echo /dev/zvol/rpool/swap none swap defaults 0 0 >> /etc/fstab
```

7.2b Create an encrypted swap device:

```
# echo cryptswap1 /dev/zvol/rpool/swap /dev/urandom \
      swap,cipher=aes-xts-plain64:sha256,size=256 >> /etc/crypttab
# systemctl daemon-reload
# systemctl start systemd-cryptsetup@cryptswap1.service
# echo /dev/mapper/cryptswap1 none swap defaults 0 0 >> /etc/fstab
```

7.3 Enable the swap device:

```
# swapon -av
```

Step 8: Full Software Installation

8.1 Upgrade the minimal system:

```
# apt dist-upgrade --yes
```

8.2 Optional: Disable log compression:

As `/var/log` is already compressed by ZFS, logrotate's compression is going to burn CPU and disk I/O for (in most cases) very little gain. Also, if you are making snapshots of `/var/log`, logrotate's compression will actually waste space, as the uncompressed data will live on in the snapshot. You can edit the files in `/etc/logrotate.d` by hand to comment out `compress`, or use this loop (copy-and-paste highly recommended):

```
# for file in /etc/logrotate.d/* ; do
    if grep -Eq "(^|[^#y])compress" "$file" ; then
        sed -i -r "s/(^|[^#y])(compress)/\1#\2/" "$file"
    fi
done
```

8.3 Reboot:

```
# reboot
```

Step 9: Final Cleanup

9.1 Wait for the system to boot normally. Login using the account you created. Ensure the system (including networking) works normally.

9.2 Optional: Delete the snapshot of the initial installation:

```
$ sudo zfs destroy rpool/ROOT/debian@install
```

9.3 Optional: Disable the root password

```
$ sudo usermod -p '*' root
```

9.4 Optional (not recommended):

If you prefer the graphical boot process, you can re-enable it now. It will make debugging boot problems more difficult, though.

```
$ sudo vi /etc/default/grub
Add quiet to GRUB_CMDLINE_LINUX_DEFAULT
Comment out GRUB_TERMINAL=console
Save and quit.

$ sudo update-grub
```

Troubleshooting

Rescuing using a Live CD

Boot the Live CD and open a terminal.

Become root and install the ZFS utilities:

```
$ sudo -i
# apt install --yes linux-headers-$(uname -r)
# apt install --yes zfs-dkms
```

This will automatically import your pool. Export it and re-import it to get the mounts right:

```
# zpool export -a
# zpool import -N -R /mnt rpool
# zfs mount rpool/ROOT/debian
# zfs mount -a
```

If needed, you can chroot into your installed environment:

```
# mount --rbind /dev  /mnt/dev
# mount --rbind /proc /mnt/proc
# mount --rbind /sys  /mnt/sys
# chroot /mnt /bin/bash --login
```

Do whatever you need to do to fix your system.

When done, cleanup:

```
# mount | grep -v zfs | tac | awk '/\/mnt/ {print $3}' | xargs -i{} umount -lf {}
# zpool export rpool
# reboot
```

MPT2SAS

Most problem reports for this tutorial involve `mpt2sas` hardware that does slow asynchronous drive initialization, like some IBM M1015 or OEM-branded cards that have been flashed to the reference LSI firmware.

The basic problem is that disks on these controllers are not visible to the Linux kernel until after the regular system is started, and ZoL does not hotplug pool members. See https://github.com/zfsonlinux/zfs/issues/330.

Most LSI cards are perfectly compatible with ZoL. If your card has this glitch, try setting rootdelay=X in GRUB_CMDLINE_LINUX. The system will wait up to X seconds for all drives to appear before importing the pool.

Areca

Systems that require the `arcsas` blob driver should add it to the `/etc/initramfs-tools/modules` file and run `update-initramfs -c -k all`.

Upgrade or downgrade the Areca driver if something like `RIP: 0010:[<ffffffff8101b316>] [<ffffffff8101b316>] native_read_tsc+0x6/0x20` appears anywhere in kernel log. ZoL is unstable on systems that emit this error message.

VMware

- Set `disk.EnableUUID = "TRUE"` in the vmx file or vsphere configuration. Doing this ensures that `/dev/disk` aliases are created in the guest.

QEMU/KVM/XEN

Set a unique serial number on each virtual disk using libvirt or qemu (e.g. `-drive if=none,id=disk1,file=disk1.qcow2,serial=1234567890`).

To be able to use UEFI in guests (instead of only BIOS booting), run this on the host:

```
$ sudo apt install ovmf
$ sudo vi /etc/libvirt/qemu.conf
Uncomment these lines:
nvram = [
   "/usr/share/OVMF/OVMF_CODE.fd:/usr/share/OVMF/OVMF_VARS.fd",
   "/usr/share/AAVMF/AAVMF_CODE.fd:/usr/share/AAVMF/AAVMF_VARS.fd"
]
$ sudo service libvirt-bin restart
```