

# 4. ZFS Datasets (ZFS Datasets)

在 ZFS 中，数据集（dataset）是文件系统层次结构的基本单位。每个数据集代表一个文件系统，可以包含文件、目录和其他数据集。数据集的创建和配置通常通过 `zfs` 命令完成。例如，可以在 `/usr/local/var` 下创建一个名为 `mydataset` 的数据集，并设置其权限和属性。

ZFS 数据集的创建和配置通常通过 `zfs` 命令完成。例如，可以在 `/usr/local/var` 下创建一个名为 `mydataset` 的数据集，并设置其权限和属性。数据集的创建和配置通常通过 `zfs` 命令完成。

在 ZFS 中，数据集（dataset）是文件系统层次结构的基本单位。每个数据集代表一个文件系统，可以包含文件、目录和其他数据集。数据集的创建和配置通常通过 `zfs` 命令完成。例如，可以在 `/usr/local/var` 下创建一个名为 `mydataset` 的数据集，并设置其权限和属性。

在 ZFS 中，数据集（dataset）是文件系统层次结构的基本单位。每个数据集代表一个文件系统，可以包含文件、目录和其他数据集。数据集的创建和配置通常通过 `zfs` 命令完成。

## 数据集 (Datasets)

在 ZFS 中，数据集（dataset）是文件系统层次结构的基本单位。每个数据集代表一个文件系统，可以包含文件、目录和其他数据集。数据集的创建和配置通常通过 `zfs` 命令完成。

ZFS 数据集的创建和配置通常通过 `zfs` 命令完成。例如，可以在 `/usr/local/var` 下创建一个名为 `mydataset` 的数据集，并设置其权限和属性。数据集的创建和配置通常通过 `zfs` 命令完成。

在 ZFS 中，数据集（dataset）是文件系统层次结构的基本单位。每个数据集代表一个文件系统，可以包含文件、目录和其他数据集。数据集的创建和配置通常通过 `zfs` 命令完成。





```
$ zfs list
NAME                USED AVAIL REFER MOUNTPOINT
mypool              420M 17.9G  96K none
mypool/ROOT         418M 17.9G  96K none
mypool/ROOT/default 418M 17.9G  418M /
...
```

이제 `zfs list` 명령을 사용하여 ZFS 풀과 데이터셋의 상태를 확인합니다.

`USED` 및 `REFER` 열은 각각 데이터셋이 차지하고 있는 공간과 참조하고 있는 공간의 크기를 나타냅니다. ZFS는 데이터를 블록 단위로 저장하며, 각 블록은 1KB의 크기를 가집니다. 이 예제에서 `mypool/ROOT/default` 데이터셋은 418M의 공간을 사용하고 있으며, 이는 `mypool/ROOT` 데이터셋의 참조를 받고 있습니다.

`AVAIL` 열은 데이터셋이 사용할 수 있는 공간의 크기를 나타냅니다. 이 예제에서 `mypool` 풀은 17.9G의 공간을 사용할 수 있습니다.

데이터셋의 상태를 더 자세히 보려면 `zfs mount` 명령을 사용합니다. 이 명령은 데이터셋이 마운트된 상태를 표시하며, ZFS 데이터셋의 마운트 상태를 확인할 수 있습니다.

데이터셋의 상태를 더 자세히 보려면 `zfs list` 명령을 사용합니다. 이 명령은 데이터셋의 상태를 표시하며, ZFS 데이터셋의 상태를 확인할 수 있습니다.

```
$ zfs list mypool/lamb
NAME                USED AVAIL REFER MOUNTPOINT
mypool/lamb         192K 17.9G  96K /lamb
```

`-t` 옵션은 데이터셋의 상태를 더 자세히 보려면 `zfs list -t snapshot` 명령을 사용합니다. 이 명령은 데이터셋의 상태를 표시하며, ZFS 데이터셋의 상태를 확인할 수 있습니다.

```
$ zfs list -t snapshot
NAME                USED AVAIL REFER MOUNTPOINT
zroot/var/log/db@backup 0    - 10.0G -
```

이제 `zfs list -t snapshot` 명령을 사용하여 ZFS 풀과 데이터셋의 상태를 확인합니다.

## 데이터셋 생성, 이동, 및 파괴 (Creating, Moving, and Destroying Datasets)

`zfs create` 명령은 새로운 데이터셋을 생성합니다. 이 명령은 데이터셋의 이름을 지정하며, ZFS 데이터셋의 상태를 확인할 수 있습니다.

## 파일시스템 생성 (Creating Filesystems)

zfs create mypool/lamb  
zfs create mypool/lamb

```
$ zfs create mypool/lamb
```

zfs create mypool ZFS lamb . zfs create mypool lamb (zfs "ZFS" " ").

```
$ mount | grep lamb  
mypool/lamb on /lamb (zfs, local, noatime, nfsv4acls)
```

zfs create mypool/lamb/baby  
zfs create mypool/lamb/baby

```
$ zfs create mypool/lamb/baby
```

zfs create mypool/lamb/baby 'zfs /lamb' 'zfs' , zfs create mypool/lamb/baby 'zfs' 'zfs' .

## zfs create (Creating Volumes)

-V zfs create mypool/avolume zfs create mypool/avolume . zfs create mypool/avolume

```
$ zfs create -V 4G mypool/avolume
```

Zvols zfs create mypool/avolume -t volume zfs list  
zvol mypool/avolume .

```
$ zfs list mypool/avolume  
NAME      USED AVAIL REFER MOUNTPOINT  
ypool/avolume 4.13G 17.9G 64K -
```

Zvols zfs create mypool/avolume -t volume zfs list  
zvol mypool/avolume . 4GB zvol 4.13GB

zvol mypool/avolume /dev/zvol mypool/avolume . /dev/zvol mypool/avolume .

```
$ ls -al /dev/zvol/mypool/avolume  
crw-r----- 1 root operator 0x4d Mar 27 20:22 /dev/zvol/mypool/avolume
```

newfs(8) newfs(8) newfs(8) newfs(8) newfs(8) newfs(8) newfs(8) newfs(8)

# zfs rename (Renaming Datasets)

zfs rename 命令用于重命名 ZFS 数据集。它的基本语法如下：

```
$ zfs rename db/production db/old
$ zfs rename db/testing db/production
```

zfs rename 命令的选项包括：-f 强制重命名，即使目标数据集存在也会覆盖；-u 更新数据集的元数据。例如，使用 -f 选项重命名 db/production 数据集：

zfs rename -f db/production db/old

# zfs move (Moving Datasets)

zfs move 命令用于移动 ZFS 数据集。它的基本语法如下：

zfs move 命令的选项包括：-f 强制移动，即使目标数据集存在也会覆盖；-u 更新数据集的元数据。例如，使用 -f 选项移动 db/production 数据集：

```
$ zfs rename zroot/var/db/mysql zroot/important/mysql
```

zfs move 命令的选项包括：-f 强制移动，即使目标数据集存在也会覆盖；-u 更新数据集的元数据。例如，使用 -f 选项移动 db/production 数据集：

zfs move -f db/production db/old

# zfs destroy (Destroying Datasets)

zfs destroy 命令用于销毁 ZFS 数据集。它的基本语法如下：

```
$ zfs destroy db/old
```

-r 选项用于递归销毁数据集及其所有子数据集。-R 选项用于递归销毁数据集及其所有子数据集，包括快照。例如，使用 -r 选项销毁 db/old 数据集：

zfs(8) 命令 使用 选项 来 指定 要 查看 的 属性 名称 和 值 。 -v 选项 用于 显示 属性 的 值 ， -n 选项 用于 显示 属性 的 名称 。 -v 选项 和 -n 选项 可以 一起 使用 ， 例如 zfs(8) 命令 的 输出 如下 所示 。

## ZFS 属性 (ZFS Properties)

ZFS 属性 是 用于 描述 文件系统 的 元数据 的 集合 。 它们 包括 文件系统 的 名称 、 大小 、 容量 、 使用 情况 等 信息 。 ZFS 属性 可以 通过 zfs(8) 命令 来 查看 和 修改 。

要 查看 文件系统 的 属性 ， 可以使用 zfs get 命令 。 例如 ， 要 查看 mypool/lamb 文件系统 的 属性 ， 可以使用以下 命令 。

## 查看属性 (Viewing Properties)

zfs(8) 命令 的 输出 如下 所示 。 zfs get 命令 的 输出 格式 如下 所示 ， 其中 NAME 是 属性 名称 ， PROPERTY 是 属性 类型 ， VALUE 是 属性 值 ， SOURCE 是 属性 来源 。

```
$ zfs get compression mypool/lamb
NAME          PROPERTY      VALUE          SOURCE
mypool/lamb   compression   lz4             inherited from mypool
```

NAME 属性 是 文件系统 的 名称 ， PROPERTY 属性 是 属性 类型 ， VALUE 属性 是 属性 值 ， SOURCE 属性 是 属性 来源 。

SOURCE 属性 是 属性 来源 ， 它 可以是 文件系统 的 名称 或 属性 名称 。 例如 ， 如果 属性 名称 是 mypool/lamb ， 那么 SOURCE 属性 就是 mypool/lamb 。 如果 属性 名称 是 mypool/lamb:compression ， 那么 SOURCE 属性 就是 mypool/lamb:compression 。

要 查看 文件系统 的 所有 属性 ， 可以使用 zfs get all 命令 。 例如 ， 要 查看 mypool/lamb 文件系统 的 所有 属性 ， 可以使用以下 命令 。

zfs(8) 命令 的 输出 如下 所示 。 zfs get all 命令 的 输出 格式 如下 所示 ， 其中 NAME 是 属性 名称 ， PROPERTY 是 属性 类型 ， VALUE 是 属性 值 ， SOURCE 是 属性 来源 。

```
$ zfs get all mypool/lamb
NAME          PROPERTY      VALUE          SOURCE
mypool/lamb   type          filesystem      -
mypool/lamb   creation      Fri Mar 27 20:05 2015 -
mypool/lamb   used          192K           -
...
```

all 属性 是 文件系统 的 名称 ， PROPERTY 属性 是 属性 类型 ， VALUE 属性 是 属性 值 ， SOURCE 属性 是 属性 来源 。





但是，如果我们在 `setuid` 属性上设置 `setuid` 属性，那么...

在 `setuid` 属性上设置 `setuid` 属性，那么...

ZFS 的 `setuid` 属性，`on` 属性，`off` 属性...

(User-Defined Properties)

ZFS 的 `org` 属性，`org.freebsd:swap` 属性...

在 `org.freebsd:swap` 属性上设置 `org.freebsd:swap` 属性...

在 `com.allanjude:backup_ignore` 属性上设置 `com.allanjude:backup_ignore` 属性...

在 `com.allanjude:backup_ignore` 属性上设置 `com.allanjude:backup_ignore` 属性...

```
$ zfs set com.allanjude:backup_ignore=on mypool/lamb
```

Jude 的 `com.allanjude:backup_ignore` 属性，`true` 属性...

(Parent/Child Relationships)

在 `zfs(8)` 属性上设置 `zfs(8)` 属性...

```
$ zfs get -r compression mypool/lamb
NAME          PROPERTY  VALUE  SOURCE
mypool/lamb   compression  lz4    inherited from mypool
```

```
mypool/lamb/baby compression off local
```

Now we can see that the `compression` property of `mypool/lamb` is `off`. This is because the `compression` property of `mypool/lamb` is `off` and the `compression` property of `mypool/lamb/baby` is `off`.

**zfs inherit** `compression` `lz4` `inherited from mypool`.

```
$ zfs inherit compression mypool/lamb/baby
$ zfs get -r compression mypool/lamb
NAME          PROPERTY      VALUE SOURCE
mypool/lamb    compression    lz4  inherited from mypool
mypool/lamb/baby compression    lz4  inherited from mypool
```

Now we can see that the `compression` property of `mypool/lamb` is `lz4`, and the `compression` property of `mypool/lamb/baby` is `lz4`.

Now we can see that the `compression` property of `mypool/lamb` is `lz4` and the `compression` property of `mypool/lamb/baby` is `lz4`.

```
$ zfs set compression=gzip-9 mypool/lamb
$ zfs get -r compression mypool/lamb
NAME          PROPERTY      VALUE SOURCE
mypool/lamb    compression    gzip-9 local
mypool/lamb/baby compression    gzip-9 inherited from mypool/lamb
```

Now we can see that the `compression` property of `mypool/lamb` is `gzip-9` and the `compression` property of `mypool/lamb/baby` is `gzip-9`.

## zfs (Inheritance and Renaming)

Now we can see that the `compression` property of `mypool/lamb` is `gzip-9` and the `compression` property of `mypool/lamb/baby` is `gzip-9`.

Now we can see that the `compression` property of `mypool/lamb` is `gzip-9` and the `compression` property of `mypool/lamb/baby` is `gzip-9`.

```
$ zfs create mypool/second
$ zfs get compress mypool/second
NAME          PROPERTY      VALUE SOURCE
mypool/second  compression    lz4  inherited from mypool
```

Now we can see that the `compression` property of `mypool/second` is `lz4` and the `compression` property of `mypool/lamb/baby` is `gzip-9`.

```
$ zfs rename mypool/lamb/baby mypool/second/baby

$ zfs get -r compression mypool/second

NAME          PROPERTY  VALUE  SOURCE
mypool/second  compression  lz4    inherited from mypool
mypool/second/baby  compression  lz4    inherited from mypool
```

在 ZFS 中，属性（properties）是用于配置和管理文件系统的重要工具。它们可以分为继承属性（inherited properties）和局部属性（local properties）。继承属性是从父文件系统继承而来的，而局部属性则是直接在当前文件系统上设置的。ZFS 支持多种属性，如压缩（compression）、加密（encryption）等。在本节中，我们将重点介绍如何查看和设置 ZFS 属性。

### 查看属性 (Viewing Properties)

要查看 ZFS 文件系统的属性，可以使用 `zfs get` 命令。该命令允许您指定要查看的属性名称，并递归地应用于指定的文件系统及其所有子文件系统。例如，要查看 `mypool/second` 及其所有子文件系统的压缩属性，可以使用以下命令：

```
$ zfs inherit com.allanjude:backup_ignore mypool/lamb
```

在上面的命令中，`com.allanjude:backup_ignore` 是一个用户自定义的属性名称，而 `mypool/lamb` 是目标文件系统的名称。ZFS 会自动将属性应用到该文件系统及其所有子文件系统上。您可以通过 `zfs get` 命令来验证属性的设置情况。

```
$ zfs inherit -r compression mypool
```

在上面的命令中，`-r` 选项表示递归地应用于所有子文件系统。通过这种方式，您可以轻松地批量设置或修改 ZFS 属性。

## ZFS 文件系统挂载 (Mounting ZFS Filesystems)

在 Linux 系统中，ZFS 文件系统通常通过 `/etc/fstab` 文件进行挂载配置。该文件是一个文本文件，其中每一行代表一个需要挂载的文件系统及其相关的配置信息。ZFS 的挂载配置通常包括文件系统名称、挂载点、文件系统类型（`zfs`）以及挂载选项。

在 ZFS 中，挂载点（mountpoint）是指文件系统被挂载到系统中的具体位置。您可以通过 `zfs mountpoint` 命令来查看指定文件系统的挂载点。例如，要查看 `zroot/usr/home` 的挂载点，可以使用以下命令：

```
$ zfs get mountpoint zroot/usr/home

NAME          PROPERTY  VALUE  SOURCE
zroot/usr/home  mountpoint  /usr/home  inherited from zroot/usr
```

zroot 的 mountpoint 是 `/usr/home` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

在 FreeBSD 中，zroot 的 mountpoint 是 `/usr`。zroot 的 mountpoint 是 `/usr` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

zroot 的 mountpoint 是 `/usr`。zroot 的 mountpoint 是 `/usr` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

zroot 的 mountpoint 是 `/usr`。zroot 的 mountpoint 是 `/usr` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

zroot 的 mountpoint 是 `/usr`。zroot 的 mountpoint 是 `/usr` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

zroot 的 mountpoint 是 `/usr`。zroot 的 mountpoint 是 `/usr` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

zroot 的 mountpoint 是 `/usr`。zroot 的 mountpoint 是 `/usr` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

zroot 的 mountpoint 是 `/usr`。zroot 的 mountpoint 是 `/usr` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

zroot 的 mountpoint 是 `/usr`。zroot 的 mountpoint 是 `/usr` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

## Datasets without Mount Points

zroot 的 mountpoint 是 `/usr`。zroot 的 mountpoint 是 `/usr` 的 mountpoint。zroot 的 mountpoint 是 `/usr` 的 mountpoint。

```
$ zfs mount
zroot/ROOT/default /
zroot/tmp /tmp
zroot/usr/home /usr/home
zroot/usr/ports /usr/ports z
root/usr/src /usr/src
```

...

/usr 目录 是否 挂载 到 /usr 目录 下 ？

**zfs list** 命令 可以 查看 /usr 目录 是否 挂载 到 zroot/usr 目录 下 。  
**canmount** 属性 表示 是否 可以 挂载 。

```
$ zfs list -o name,canmount,mountpoint -r zroot/usr
```

```
NAME          CANMOUNT MOUNTPOINT
```

```
zroot/usr      off      /usr
```

```
zroot/usr/home on      /usr/home
```

```
zroot/usr/ports on      /usr/ports
```

```
zroot/usr/src  on      /usr/src
```

**canmount** 属性 为 off 表示 不能 挂载 到 zroot/usr 目录 下 。usr/bin 目录 不能 挂载 到 /usr/local 目录 下 ，usr/src 目录 不能 挂载 到 /usr 目录 下 。

挂载 到 /var 目录 下 的 数据集 可以 挂载 到 /usr 目录 下 。

## Multiple Datasets with the Same Mount Point

**canmount** 属性 为 off 表示 不能 挂载 到 zroot/usr 目录 下 。usr/bin 目录 不能 挂载 到 /usr/local 目录 下 ，usr/src 目录 不能 挂载 到 /usr 目录 下 。

FreeBSD 系统 中 的 zroot 数据集 可以 挂载 到 /usr 目录 下 。

数据集 的 **mountpoint** 属性 表示 挂载 点 ，zroot 数据集 的 /usr 目录 下 的 数据集 的 **canmount** 属性 为 off 表示 不能 挂载 到 /usr 目录 下 。

数据集 的 **mountpoint** 属性 表示 挂载 点 ，zroot 数据集 的 /usr 目录 下 的 数据集 的 **canmount** 属性 为 off 表示 不能 挂载 到 /usr 目录 下 。

```
$ zfs create db/programs # zfs create db/data
```

数据集 的 **mountpoint** 属性 表示 挂载 点 ，zroot 数据集 的 /usr 目录 下 的 数据集 的 **canmount** 属性 为 off 表示 不能 挂载 到 /usr 目录 下 。

```
$ zfs set canmount=off db/programs
$ zfs set mountpoint=/opt db/programs
```

zfs 的 mountpoint 可以透過 zfs 的 set 指令來設定。

```
$ zfs set readonly=on db/programs
```

*db/data* 的 `readonly` 屬性可以透過 `zfs set` 指令來設定。此外，`exec` 和 `setuid` 屬性也可以透過 `zfs set` 指令來設定。

```
$ zfs set canmount=off db/data
$ zfs set mountpoint=/opt db/data
$ zfs set setuid=off db/data
$ zfs set exec=off db/data
```

zfs 的 `canmount` 屬性可以透過 `zfs set` 指令來設定。此外，`exec` 和 `setuid` 屬性也可以透過 `zfs set` 指令來設定。

```
$ zfs create db/programs/bin
$ zfs create db/programs/sbin
$ zfs create db/data/test
$ zfs create db/data/production
```

zfs 的 `canmount` 屬性可以透過 `zfs set` 指令來設定。此外，`exec` 和 `setuid` 屬性也可以透過 `zfs set` 指令來設定。

## zfs 的 Pools without Mount Points (Pools without Mount Points)

zfs 的 `canmount` 屬性可以透過 `zfs set` 指令來設定。此外，`exec` 和 `setuid` 屬性也可以透過 `zfs set` 指令來設定。

```
$ zfs set mountpoint=none mypool
```

zfs 的 `canmount` 屬性可以透過 `zfs set` 指令來設定。此外，`exec` 和 `setuid` 屬性也可以透過 `zfs set` 指令來設定。

```
$ zfs set mountpoint=/someplace mypool/lamb
```

zfs 的 `canmount` 屬性可以透過 `zfs set` 指令來設定。此外，`exec` 和 `setuid` 屬性也可以透過 `zfs set` 指令來設定。

# Manual Mounting and Unmounting Filesystems (Manually Mounting and Unmounting Filesystems)

For manual mounting, use `zfs mount` or `canmount`. The `noauto` flag in `/etc/fstab` indicates that the filesystem should not be mounted automatically.

```
$ zfs mount mypool/usr/src
```

To unmount, use `zfs unmount`.

```
$ zfs unmount mypool/second
```

For mounting with options, use `zfs mount -o mountpoint=/mnt mypool/lamb`. The `-o` flag is used to specify options.

```
$ zfs mount -o mountpoint=/mnt mypool/lamb
```

The `mountpoint` option is used to specify the mount point. The `legacy` option is used to specify the legacy mount point.

## ZFS and /etc/fstab (ZFS and /etc/fstab)

The `/etc/fstab` file contains information about the filesystems. The `mountpoint` option is used to specify the mount point.

```
$ zfs set mountpoint=legacy mypool/second
```

The `mount(8)` command is used to mount filesystems.

```
$ mount -t zfs mypool/second /tmp/second
```

The `/etc/fstab` file contains information about the filesystems. The `zfs` filesystem type is used. The `noatime`, `noexec`, `readonly`, `ro`, `nosuid`, `fsck`, and `scratch/junk` options are used.

```
scratch/junk /tmp nosuid 2 0
```

The `/etc/fstab` file contains information about the filesystems. The `zfs` filesystem type is used.

## ZFS Tweaking ZFS Volumes (Tweaking ZFS Volumes)





```
$ dd if=/dev/random of=/lamb/random1 bs=1m count=10
10+0 records in
10+0 records out
10485760 bytes transferred in 0.144787 secs (72421935 bytes/sec)

$ zfs set copies=2 mypool/lamb
```

이제 10MB의 데이터를 랜덤으로 생성하고, 이를 mypool/lamb에 복사합니다. ZFS는 이 데이터를 2개의 복사본으로 저장합니다. (이제 mypool/lamb에 2개의 복사본이 있습니다.)

```
$ zfs list mypool/lamb

NAME      USED AVAIL REFER MOUNTPOINT
mypool/lamb 10.2M 13.7G 10.1M /lamb
```

이제 10MB의 데이터를 랜덤으로 생성하고, 이를 mypool/lamb에 복사합니다. ZFS는 이 데이터를 2개의 복사본으로 저장합니다. (이제 mypool/lamb에 2개의 복사본이 있습니다.)

```
$ dd if=/dev/random of=/lamb/random2 bs=1m count=10
10+0 records in
10+0 records out
10485760 bytes transferred in 0.141795 secs (73950181 bytes/sec)
```

이제 10MB의 데이터를 랜덤으로 생성하고, 이를 mypool/lamb에 복사합니다. ZFS는 이 데이터를 2개의 복사본으로 저장합니다. (이제 mypool/lamb에 2개의 복사본이 있습니다.)

```
$ zfs list mypool/lamb

NAME      USED AVAIL REFER MOUNTPOINT
mypool/lamb 30.2M 13.7G 30.1M /lamb
```

이제 30MB의 데이터를 랜덤으로 생성하고, 이를 mypool/lamb에 복사합니다. ZFS는 이 데이터를 2개의 복사본으로 저장합니다. (이제 mypool/lamb에 2개의 복사본이 있습니다.)

```
$ ls -l /lamb/random*

-rw-r--r-- 1 root wheel 10485760 Apr 6 15:27 /lamb/random1
-rw-r--r-- 1 root wheel 10485760 Apr 6 15:29 /lamb/random2
```

이제 30MB의 데이터를 랜덤으로 생성하고, 이를 mypool/lamb에 복사합니다. ZFS는 이 데이터를 2개의 복사본으로 저장합니다. (이제 mypool/lamb에 2개의 복사본이 있습니다.)

### 데이터 중복성 (Metadata Redundancy)

이제 30MB의 데이터를 랜덤으로 생성하고, 이를 mypool/lamb에 복사합니다. ZFS는 이 데이터를 2개의 복사본으로 저장합니다. (이제 mypool/lamb에 2개의 복사본이 있습니다.)

