

?? 3. ?(Pools)

[illegible]

ZFS ??

UFS extfs .
 (inode) , .
 NTFS FAT .

ZFS 是 一个 跨 平台 的 文件 系统 ， 它 是 由 开放 存储 基金会 开发 的 。 它 是 一个 跨 平台 的 文件 系统 ， 它 是 由 开放 存储 基金会 开发 的 。 ZFS 是 一个 跨 平台 的 文件 系统 ， 它 是 由 开放 存储 基金会 开发 的 。 ZFS 是 一个 跨 平台 的 文件 系统 ， 它 是 由 开放 存储 基金会 开发 的 。

00 00 00 00 00 , ZFS 00 00000 0000 000 ! 00 000 000
 0000 . ZFS 000000 00 00 00 00 00 00 000 0000 . ZFS
 0 000 0000 0000 000 , 000 000 0000 000 0000 000 000 000
 0000 000 . (0 00 0 000 000 0000 6000 000 00000 .) 000 00 12800
 000 000000 0000 , 00 00 000 0 00000 00000 . 000 00000 0000
 ZFS 0000 00000 .

00000000 0000 0000 0000 . ZFS 000000 000000 0 0 0000 00 0000 00
 0000 0000 00 00 0000 0000 0000 . 00 0000 0000 ZFS 00 00 (ditto block,
 0000 0000 0000)00 00 0000 0000 . 0000 0000 0000 0000 00 00 0000 00
 0000 0000 0 00 00 0000 0000 . (ZFS 0000 000000 000000 0 0 00 0000
 0000 0000 00 0000 0 0000 .)

ZFS 的 数据 块 大小 是 128KB， 而 txg 的 大小 是 64KB。 这 意 味 着 每 个 txg 只 能 存 储 一 半 的 数 据 块。 这 是 为 什 么 呢？ 这 是 因 为 ZFS 的 数 据 块 是 按 照 128KB 的 大 小 来 划 分 的， 而 txg 的 大 小 是 64KB， 这 就 意 味 着 每 个 txg 只 能 存 储 一 半 的 数 据 块。

ZFS 文件系统 支持 多种 文件系统 格式 , 包括 zfs ddb 文件系统 格式 的 文件系统 格式 . 文件系统 格式 包括 UFS2 和 extfs 文件系统 格式 .

?????, RAID ? ? (Stripes, RAID, and Pools)

1. 数据冗余：数据在多个存储设备上复制，提高可靠性。
 2. 性能提升：通过并行读写，提高数据访问速度。
 3. 容错能力：即使部分存储设备故障，数据仍可访问。
 4. 灵活性：支持多种 RAID 级别，满足不同需求。
 5. 易于扩展：可以根据需要增加存储设备。

1 个池 1 个 VDEV . 1 个池 **db** **zroot** 1 个池 1 个 VDEV .
 1 个池 1 个 VDEV 1 个池 1 个 VDEV 1 个池 . 1 个池 , 1 个池 1 个池 , 1 个池 1 个池 1 个池 .
EXPANDSZ 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 . 5 个池 1 个池 1 个池 1 个池 1 个池 1 个池 . 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 .
FRAG 1 个池 1 个池 1 个池 1 个池 1 个池 . 1 个池 1 个池 1 个池 1 个池 .
CAP 1 个池 1 个池 1 个池 1 个池 1 个池 .
DEDUP 1 个池 1 个池 1 个池 1 个池 1 个池 . 6 个池 1 个池 1 个池 1 个池 .
 1 个池 **HEALTH** 1 个池 VDEV 1 个池 1 个池 . 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 . 5 个池 1 个池 1 个池 1 个池 .
 1 个池 , **ALTROOT** 1 个池 1 个池 1 个池 , 1 个池 "1 个池 1 个池" 1 个池 .

4 个池 1 个池 1 个池 1 个池 . 1 个池 1 个池 1 个池 1 个池 1 个池 **zpool list** 1 个池 1 个池 1 个池 .
 1 个池 1 个池 1 个池 **prod** 1 个池 **test** 1 个池 1 个池 .

```
$ zpool list prod test
```

1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 **-v** 1 个池 1 个池 . 1 个池 1 个池 1 个池 1 个池 .

```
$ zpool list -v zroot
```

-p 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 , **-H** 1 个池 1 个池 1 个池 .
 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 .

1 个池 VDEV 1 个池 1 个池 1 个池 1 个池 1 个池 **zpool status** 1 个池 1 个池 . 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 .

1 个池 1 个池 1 个池 **zpool status -x** 1 个池 1 个池 .

```
$ zpool status -x
all pools are healthy
```

1 个池 1 个池 1 个池 1 个池 .

?? ?? VDEV (Multiple VDEVs)

1 个池 1 个池 1 个池 VDEV 1 个池 1 个池 . VDEV 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 1 个池 . 1 个池 1 个池 1 个池 VDEV 1 个池 1 个池 . 1 个池 1 个池 1 个池 1 个池 1 个池 , 1 个池 1 个池 1 个池 1 个池 1 个池 **ZFS** 1 个池 VDEV 1 个池 1 个池 .

2 个池 1 个池 VDEV 1 个池 1 个池 1 个池 . 1 个池 1 个池 1 个池 1 个池 . 1 个池 VDEV 1 个池 1 个池 1 个池 1 个池 , 1 个池 1 个池 1 个池 (1 个池 1 个池) 1 个池 1 个池 1 个池 .

但是，如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。

如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。

ZFS 的 RAID-Z3 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。

VDEV ?? (Removing VDEVs)

如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。

如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。

??? ???? ?? ?? (Pools Alignment and Disk Sector Size)

ZFS 的 RAID-Z3 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。

如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。

??? ?? (Partition Alignment)

如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。

如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。如果 VDEV 是 RAID-Z3 的，那么 VDEV 的冗余度就更高了。

/etc/sysctl.conf 文件 添加 **sysctl vfs.zfs.min_auto_ashift** 参数 设置 ashift 参数 值 。

```
$ sysctl vfs.zfs.min_auto_ashift=12
```

保存 文件 内容 后 , 重启 系统 或 重新 加载 **/etc/sysctl.conf** 文件 即可 生效 。

在 较 早期 的 FreeBSD 10.1 版本 中 , 默认 的 ashift 参数 值 为 12 , sysctl 命令 可以 查看 该 参数 的 当前 值 。

?? FreeBSD Ashift (Older FreeBSD Ashift)

10.1 版本 的 FreeBSD 系统 中 , ashift sysctl 参数 值 为 12 , ZFS 文件系统 的 默认 块 大小 为 1MB 。 在 较 早期 的 FreeBSD 版本 中 , ashift 参数 值 为 9 , 块 大小 为 512KB 。

在 较 早期 的 FreeBSD 系统 中 , 默认 的 ashift 参数 值 为 9 , 块 大小 为 512KB 。 FreeBSD 系统 中 的 **gnop(8)** 命令 可以 用于 创建 块 大小 为 512KB 的 块 设备 。 **gnop** 命令 的 用法 如下 : **gnop** (块 大小 为 512KB) 。 "块 大小 为 512KB , 4096KB 的 块 大小 为 4096KB 。" **gnop** 命令 的 用法 如下 : **gnop** (块 大小 为 512KB) 。 在 较 早期 的 FreeBSD 系统 中 , **gnop** 命令 的 用法 如下 : **gnop** (块 大小 为 512KB) 。

```
$ gnop create -S 4096 /dev/gpt/zfs0
```

在 较 早期 的 FreeBSD 系统 中 , **gnop** 命令 的 用法 如下 : **gnop** (块 大小 为 512KB) 。 在 较 早期 的 FreeBSD 系统 中 , **gnop** 命令 的 用法 如下 : **gnop** (块 大小 为 512KB) 。

```
$ zpool create compost mirror gpt/zfs0.nop gpt/zfs1
```

gnop(8) 命令 的 用法 如下 : **gnop** (块 大小 为 512KB) 。 在 较 早期 的 FreeBSD 系统 中 , **gnop** 命令 的 用法 如下 : **gnop** (块 大小 为 512KB) 。

?? ?? ???? (Creating Pools and VDEVs)

zpool(8) 命令 的 用法 如下 : **zpool** (块 大小 为 512KB) 。 在 较 早期 的 FreeBSD 系统 中 , **zpool** 命令 的 用法 如下 : **zpool** (块 大小 为 512KB) 。


















在 较 早期 的 FreeBSD 系统 中 , **zpool** 命令 的 用法 如下 : **zpool** (块 大小 为 512KB) 。

?? ???? (Sample Drives)

[illegible]

```
$ gpart create -s gpt da0
$ gpart add -a 1m -slg -l sw0 -t freebsd-swap da0
$ gpart add -a 1m -l zfs0 -t freebsd-zfs da0
```

```
$ gpart show -l da0
=>      40  1953525088  da0  GPT  (932G)
      40           2008  -   free -  (1.0M)
     2048       2097152  1  sw0  (1.0G)
    2099200  1951424512  2  zfs0 (931G)
1953523712       1416  -   free -  (708K)
```

 GPT  ZFS  gpt/zfs0  gpt/zfs5  gpt/zfs6  gpt/zfs7  gpt/zfs8 .  gpt/zfs9  gpt/zfs10  gpt/zfs11  gpt/zfs12  gpt/zfs13  gpt/zfs14  gpt/zfs15  gpt/zfs16  gpt/zfs17  gpt/zfs18 .

????? ? (Striped Pools)

[illegible]

```
$ sysctl vfs.zfs.min_auto_ashift=12
$ zpool create compost gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs4 gpt/zfs4
```







.  **zpool status**   .

```
$ zpool status
  pool: compost
state: ONLINE
  scan: none requested
config:

NAME        STATE  READ  WRITE CKSUM
compost     ONLINE    0     0     0
gpt/zfs0    ONLINE    0     0     0
```

```
gpt/zfs1  ONLINE  0      0      0
gpt/zfs2  ONLINE  0      0      0
gpt/zfs3  ONLINE  0      0      0
gpt/zfs4  ONLINE  0      0      0
```

5個 物理 磁 磁頭 . 磁 磁頭 VDEV磁 磁 .

磁 磁 磁 磁頭 磁 磁 磁頭 . 磁 磁 磁 磁 VDEV磁 磁 磁頭 磁頭 , VDEV磁 磁頭 磁頭 . 磁頭 磁 磁頭 磁頭 磁頭 . 磁 磁 磁 磁頭 磁頭 .

?? ? (Mirrored Pools)

磁頭 磁 磁 磁頭 磁 磁頭 磁頭 磁頭 . 磁 磁 磁頭 磁頭 磁頭 磁 磁 磁頭 磁頭 磁頭 . 磁 磁頭 磁 磁頭 磁頭 磁頭 , 磁 磁頭 磁頭 .

磁 **zpool create** 磁 磁 磁頭 . **mirror** 磁頭 磁頭 磁頭 磁頭 磁頭 磁頭 . 磁 磁頭 磁 磁頭 ashift磁 磁頭 .

```
$ sysctl vfs.zfs.min_auto_ashift=12
$ zpool create reflect mirror gpt/zfs0 gpt/zfs1
```

zpool status 磁 磁 磁頭 .

```
$ zpool status
pool: reflect
state: ONLINE
scan: none requested
config:

NAME          STATE READ WRITE CKSUM
reflect       ONLINE  0      0      0
mirror-0      ONLINE  0      0      0
  gpt/zfs0     ONLINE  0      0      0
  gpt/zfs1     ONLINE  0      0      0

errors: No known data errors
```

zpool 磁 磁 mirror-0磁 磁 磁頭 . mirror-0 磁 VDEV磁 . 磁 VDEV磁 磁 磁 , 磁 **gpt/zfs0** **gpt/zfs1** 磁 磁頭 磁頭 . 磁 磁 磁 磁 磁頭 磁頭 磁頭 磁頭 . 磁 磁 磁 磁 磁頭 磁頭 .

```
$ zpool create reflect mirror gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs3
```

我们使用 `zpool create` 命令来创建 RAID-Z 池。在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 (FreeBSD Mastery: Advanced ZFS 第 10 章 第 10.1 节)。

RAID-Z Pools

在创建 RAID-Z 池时，我们使用 `zpool create` 命令。RAID-Z 池的创建命令如下所示：
 `zpool create` 命令的语法如下所示：
 `zpool create <poolname> <raidtype> <disks>`
 其中，`<poolname>` 是池的名称，`<raidtype>` 是 RAID 级别，`<disks>` 是磁盘的列表。
 在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 RAID-Z 池的创建命令如下所示：
 `zpool create reflect mirror gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs3`
 其中，`reflect` 是 RAID 级别，`mirror` 是 RAID 类型，`gpt/zfs0`、`gpt/zfs1`、`gpt/zfs2` 和 `gpt/zfs3` 是磁盘的列表。

在创建 RAID-Z 池时，我们使用 `zpool create` 命令。RAID-Z 池的创建命令如下所示：
 `zpool create` 命令的语法如下所示：
 `zpool create <poolname> <raidtype> <disks>`
 其中，`<poolname>` 是池的名称，`<raidtype>` 是 RAID 级别，`<disks>` 是磁盘的列表。
 在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 RAID-Z 池的创建命令如下所示：
 `zpool create reflect mirror gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs3`
 其中，`reflect` 是 RAID 级别，`mirror` 是 RAID 类型，`gpt/zfs0`、`gpt/zfs1`、`gpt/zfs2` 和 `gpt/zfs3` 是磁盘的列表。

```
$ sysctl vfs.zfs.min_auto_ashift=12
$ zpool create raidz1 gpt/zfs0 gpt/zfs1 gpt/zfs2
```

我们使用 `zpool status` 命令来查看 RAID-Z 池的状态。
 在创建 RAID-Z 池时，我们使用 `zpool create` 命令。RAID-Z 池的创建命令如下所示：
 `zpool create` 命令的语法如下所示：
 `zpool create <poolname> <raidtype> <disks>`
 其中，`<poolname>` 是池的名称，`<raidtype>` 是 RAID 级别，`<disks>` 是磁盘的列表。
 在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 RAID-Z 池的创建命令如下所示：
 `zpool create reflect mirror gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs3`
 其中，`reflect` 是 RAID 级别，`mirror` 是 RAID 类型，`gpt/zfs0`、`gpt/zfs1`、`gpt/zfs2` 和 `gpt/zfs3` 是磁盘的列表。

```
$ zpool status bucket
pool: bucket
state: ONLINE
scan: none requested
config:

NAME        STATE READ WRITE CKSUM
bucket      ONLINE  0     0     0
raidz1-0    ONLINE  0     0     0
gpt/zfs0    ONLINE  0     0     0
gpt/zfs1    ONLINE  0     0     0
gpt/zfs2    ONLINE  0     0     0
```

我们使用 `zpool status` 命令来查看 RAID-Z 池的状态。
 在创建 RAID-Z 池时，我们使用 `zpool create` 命令。RAID-Z 池的创建命令如下所示：
 `zpool create` 命令的语法如下所示：
 `zpool create <poolname> <raidtype> <disks>`
 其中，`<poolname>` 是池的名称，`<raidtype>` 是 RAID 级别，`<disks>` 是磁盘的列表。
 在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 RAID-Z 池的创建命令如下所示：
 `zpool create reflect mirror gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs3`
 其中，`reflect` 是 RAID 级别，`mirror` 是 RAID 类型，`gpt/zfs0`、`gpt/zfs1`、`gpt/zfs2` 和 `gpt/zfs3` 是磁盘的列表。

```
$ zpool create bucket raidz3 gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4 gpt/zfs5
```

我们使用 `zpool status` 命令来查看 RAID-Z 池的状态。
 在创建 RAID-Z 池时，我们使用 `zpool create` 命令。RAID-Z 池的创建命令如下所示：
 `zpool create` 命令的语法如下所示：
 `zpool create <poolname> <raidtype> <disks>`
 其中，`<poolname>` 是池的名称，`<raidtype>` 是 RAID 级别，`<disks>` 是磁盘的列表。
 在创建 RAID-Z 池时，我们使用 `reflect` 关键字来指定 RAID 级别为 RAID-Z。
 RAID-Z 池的创建命令如下所示：
 `zpool create reflect mirror gpt/zfs0 gpt/zfs1 gpt/zfs2 gpt/zfs3`
 其中，`reflect` 是 RAID 级别，`mirror` 是 RAID 类型，`gpt/zfs0`、`gpt/zfs1`、`gpt/zfs2` 和 `gpt/zfs3` 是磁盘的列表。

```
$ zpool status
pool: bucket
```

```
state: ONLINE
  scan: none requested
config:
```

NAME	STATE	READ	WRITE	CKSUM
bucket	ONLINE	0	0	0
raidz3-0	ONLINE	0	0	0
gpt/zfs0	ONLINE	0	0	0
gpt/zfs1	ONLINE	0	0	0
...				

?? VDEV ? (Multi-VDEV Pools)

?? VDEV ? (Multi-VDEV Pools)

?? VDEV ? (Multi-VDEV Pools)

```
$ sysctl vfs.zfs.min_auto_ashift=12
$ zpool create barrel mirror gpt/zfs0 gpt/zfs1 mirror gpt/zfs2 gpt/zfs3
```

```
zpool create barrel zpool(8) barrel mirror gpt/zfs0
mirror gpt/zfs1 mirror gpt/zfs2 mirror gpt/zfs3
zpool status barrel
pool: barrel
state: ONLINE
  scan: none requested
config:
```

NAME	STATE	READ	WRITE	CKSUM
barrel	ONLINE	0	0	0
mirror-0	ONLINE	0	0	0
gpt/zfs0	ONLINE	0	0	0
gpt/zfs1	ONLINE	0	0	0

```
mirror-1  ONLINE   0      0      0
gpt/zfs2  ONLINE   0      0      0
gpt/zfs3  ONLINE   0      0      0
```

我们使用 mirror-0 和 mirror-1 两个 VDEV 来创建 RAID-10。ZFS 的 VDEV 可以包含多个物理设备，也可以包含其他 VDEV。FreeBSD 的 GEOM 系统支持 RAID 功能，但 ZFS 的 RAID 功能更强大。ZFS 的 RAID 功能支持 RAID-Z、RAID-Z1 和 RAID-Z2。RAID-Z 使用奇偶校验，而 RAID-Z1 和 RAID-Z2 使用更复杂的奇偶校验算法。

```
$ zpool create vat raidz1 gpt/zfs0 gpt/zfs1 gpt/zfs2 raidz1 gpt/zfs3 gpt/zfs4 gpt/zfs5
```

我们使用 RAID-Z1 VDEV 来创建 RAID-Z1。ZFS 的 RAID-Z1 使用奇偶校验，而 RAID-Z 使用更复杂的奇偶校验算法。ZFS 的 RAID 功能支持 RAID-Z、RAID-Z1 和 RAID-Z2。RAID-Z 使用奇偶校验，而 RAID-Z1 和 RAID-Z2 使用更复杂的奇偶校验算法。

```
$ zpool status vat
```

```
...
```

```
config:
```

NAME	STATE	READ	WRITE	CKSUM
vat	ONLINE	0	0	0
raidz1-0	ONLINE	0	0	0
gpt/zfs0	ONLINE	0	0	0
gpt/zfs1	ONLINE	0	0	0
gpt/zfs2	ONLINE	0	0	0
raidz1-1	ONLINE	0	0	0
gpt/zfs3	ONLINE	0	0	0
gpt/zfs4	ONLINE	0	0	0
gpt/zfs5	ONLINE	0	0	0

我们使用 VDEV 来创建 RAID-Z1。ZFS 的 RAID-Z1 使用奇偶校验，而 RAID-Z 使用更复杂的奇偶校验算法。

RAIDZ 使用奇偶校验，而 RAID-Z1 和 RAID-Z2 使用更复杂的奇偶校验算法。ZFS 的 RAID 功能支持 RAID-Z、RAID-Z1 和 RAID-Z2。RAID-Z 使用奇偶校验，而 RAID-Z1 和 RAID-Z2 使用更复杂的奇偶校验算法。

我们使用 VDEV 来创建 RAID-Z1。ZFS 的 RAID-Z1 使用奇偶校验，而 RAID-Z 使用更复杂的奇偶校验算法。ZFS 的 RAID 功能支持 RAID-Z、RAID-Z1 和 RAID-Z2。RAID-Z 使用奇偶校验，而 RAID-Z1 和 RAID-Z2 使用更复杂的奇偶校验算法。

?? ?? ?? (Using Log Devices)

2個 1個 1個 , ZFS 1個 1個 1個 1個 /1個 1個 1個 1個 1個 1個 1個 1個
1個 1個 . 1個 1個 1個 1個 1個 1個 1個 1個 SSD 1個 . **zpool(8)** 1個 1個
1個 **log**, 1個 1個 **cache** 1個 1個 . 1個 1個 1個 **log** 1個 **cache** 1個 1個 1個
1個 1個 . 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個
1個 .

```
$ zpool create scratch gpt/zfs0 log gtp/zlog0 cache gpt/zcache1
```

1個 1個 1個 1個 1個 .

```
$ zpool status scratch
```

...

config:

NAME	STATE	READ	WRITE	CKSUM
scratch	ONLINE	0	0	0
gpt/zfs0	ONLINE	0	0	0
logs				
gpt/zlog0	ONLINE	0	0	0
cache				
gpt/zcache1	ONLINE	0	0	0

1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 . 1個 1個 1個 1個 1個 1個
1個 1個 . 1個 1個 1個 ZFS 1個 1個 1個 1個 1個 1個 . 1個 ZIL 1個
1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 . 1個 *gpt/zfs0* 1個
gpt/zfs3 1個 1個 1個 1個 1個 1個 1個 , 1個 1個 1個 *gpt/zlog0*
gpt/zlog1 1個 1個 .

```
$ zpool create db mirror gpt/zfs0 gpt/zfs1 mirror gpt/zfs2 gpt/zfs3 log mirror gpt/zlog0  
gpt/zlog1
```

1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 1個 . 1個 1個 1個 1個 1個 1個
1個 1個 1個 1個 , 1個 1個 1個 1個 1個 . 1個 1個 SSD 1個 1個 1個 1個 1個
1個 1個 1個 1個 !

???? ?? VDEV (Mismatched VDEVs)

1個 1個 1個 1個 VDEV 1個 1個 1個 1個 , **zpool(8)** 1個 1個 1個 1個
1個 .

```
$ zpool create daftie raidz gpt/zfs0 gpt/zfs1 gpt/zfs2 mirror gpt/zfs3 gpt/zfs4 gpt/zfs5  
invalid vdev specification  
use '-f' to override the following errors:
```

mismatched replication level: both raidz and mirror vdevs are present

zpool(8) 提供了一個簡單的方法來創建和管理 ZFS 池。要創建一個池，請使用 **zpool create** 命令。例如，要創建一個名為 **db** 的池，並使用四個 VDEV（每個 VDEV 由一個 16TB 的 SATA 硬碟組成），請執行以下命令：

ZFS 池的創建過程如下：

```
zpool create db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
```

??? ??? (Reusing Providers)

在創建池時，如果指定的 VDEV 已經被另一個池使用，會收到以下錯誤：

```
$ zpool create db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
invalid vdev specification
use '-f' to override the following errors:
/dev/gpt/zfs3 is part of exported pool 'db'
```

要解決這個問題，可以使用 **-f** 選項來強制創建池，即使指定的 VDEV 已經被另一個池使用。例如，要強制創建池 **db**，請執行以下命令：

```
zpool create -f db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
```

```
$ zpool create -f db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
```

ZFS 池的創建過程如下：

```
zpool create -f db gpt/zfs1 gpt/zfs2 gpt/zfs3 gpt/zfs4
```

? ??? (Pool Integrity)

ZFS 池的完整性檢查可以通過 **fsck(8)** 命令來執行。要檢查池 **db** 的完整性，請執行以下命令：

```
fsck -f db
```

ZFS ??? (ZFS Integrity)

要檢查池 **db** 的完整性，請執行以下命令：

```
zpool status db
```



```
$ zpool scrub zroot
```

zpool status 命令用于检查 zpool 池的状态。它显示了池的扫描进度、扫描速度、扫描时间以及扫描结果。

```
$ zpool status
...
scan: scrub in progress since Tue Feb 24 11:52:23 2015
12.8G scanned out of 17.3G at 23.0M/s, 0h3m to go
0 repaired, 74.08% done
...
```

zpool status 命令的输出显示了池的扫描进度、扫描速度、扫描时间以及扫描结果。如果池处于扫描状态，输出会显示扫描的进度和预计完成时间。如果池处于空闲状态，输出会显示池的健康状况。

```
$ zpool scrub -s zroot
```

zpool scrub -s 命令用于手动启动池的扫描。它会在指定的池上启动扫描，并返回扫描的进度和结果。

??? ?? (Scrub Frequency)

ZFS 池的扫描频率是由池的容量和池的扫描策略决定的。默认情况下，池的扫描频率是每周一次。如果池的容量较大，扫描频率会降低。如果池的扫描策略是“按需扫描”，扫描频率会根据池的使用情况自动调整。

在 FreeBSD 中，ZFS 池的扫描频率是由池的容量和池的扫描策略决定的。默认情况下，池的扫描频率是每周一次。如果池的容量较大，扫描频率会降低。如果池的扫描策略是“按需扫描”，扫描频率会根据池的使用情况自动调整。

? ?? (Pool Properties)

ZFS 池的属性是指池的元数据，包括池的名称、池的容量、池的扫描策略等。这些属性可以通过 zpool get 命令来查看。zpool get 命令的输出显示了池的名称、池的容量、池的扫描策略以及池的其他属性。

zpool get 命令的输出显示了池的名称、池的容量、池的扫描策略以及池的其他属性。如果池的属性发生变化，zpool get 命令的输出也会相应地发生变化。

? ?? ?? (Viewing Pool Properties)

zpool get all 命令用于查看池的所有属性。它会在指定的池上执行，并返回池的所有属性的名称、属性值以及属性的来源。zpool get all 命令的输出显示了池的名称、池的容量、池的扫描策略以及池的其他属性。

```
$ zpool get all zroot
NAME      PROPERTY  VALUE      SOURCE
zroot     SIZE      17.3G      none
zroot     USED      12.8G      none
zroot     AVAIL     4.5G       none
zroot     CAPACITY   74.08%     none
zroot     HEALTH    ONLINE    none
zroot     SCANNED   74.08%     none
zroot     CLUSTERS  0          none
zroot     FRAGMENTS 0          none
zroot     MOUNTED   0          none
zroot     MOUNTPOINT /          none
```

```
zroot  size      920G    -
zroot  capacity  1%      -
zroot  altroot    -        default
zroot  health     ONLINE  -
...
```

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ .

[illegible]

SOURCE [] [] [] [] [] . [] [] [] [] [] [] [] [] []
[] [] [] [] . [] [] [] [] [] [] [] [] [] [] []
[] [] . [] [] [] [] [] [] [] [] [] [] [] [] [] [] FreeBSDD [] [] [] []
[] [] . [] [] [] [] [] [] [] [] [] [] , [] [] [] [] []
[] [] [] [] [] [] .

□□ □□□ □□□□□ □□ □□□ □□ **zpool get**□ □□□□□ .

```
$ zpool get size
```

NAME	PROPERTY	VALUE	SOURCE
db	size	2.72T	-
zroot	size	920G	-

?? ?? (Changing Pool Properties)

```

| | |||   | ||   ||||   | |   ||   |||||||   . zpool set |||   ||||   |   |||
|||||. ||||   |    comment |||   |||||.

```

```
$ zpool set comment="Main OS files" zroot
```

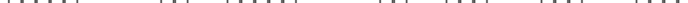
```
$ zpool get comment
```
















NAME	PROPERTY	VALUE	SOURCE
db	comment	-	default
zroot	comment	Main OS files	local




[illegible]

```
$ zpool set comment="-" zroot
# zpool get comment
```

NAME	PROPERTY	VALUE	SOURCE
db	comment	-	default
zroot	comment	-	local






-0






.






-0





.

```
$ zpool create -o altroot=/mnt -O canmount=off -m none zroot /dev/gpt/disk0
```

```

# altroot /mnt , canmount off
. ZFS

```

? ???? (Pool History)

zpool 是 一个 命令， 用于 管理 存储池。 它 可以 创建、 删除、 修改、 以及 监控 存储池。 存储池 是 一个 逻辑上的 存储单元， 它 可以 由 一个 或多个 物理存储设备 组成。 zpool 可以 管理 各种 类型的 存储设备， 包括 硬盘、 固态硬盘、 以及 网络存储设备。 zpool 还可以 管理 存储池 的 性能、 容量、 以及 冗余性。 zpool 是 一个 非常 强大 且 灵活 的工具， 它 可以 帮助 您 轻松 地 管理 您的 存储系统。

zpool history

```
$ zpool history zroot
History for 'zroot':
2014-01-07.04:12:05 zpool create -o altroot=/mnt -O canmount=off -m none zroot mirror /
dev/gpt/disk0.nop /dev/gpt/disk1.nop
2014-01-07.04:12:50 zfs set checksum=fletcher4 zroot
2014-01-07.04:13:00 zfs set atime=off zroot
...
```

FreeBSD ZFS

```
...
2015-03-12.14:36:35 zpool set comment=Main OS files zroot
2015-03-12.14:43:45 zpool set comment=- zroot
```

comment

zpool 是 ZFS 的池，是 ZFS 文件系统的数据容器。zpool 是 ZFS 的池，是 ZFS 文件系统的数据容器。zpool 是 ZFS 的池，是 ZFS 文件系统的数据容器。

Zpool ?? ?? ??? (Zpool Maintenance Automation)

FreeBSD 的 `periodic(8)` 是用于定期执行任务的工具。它可以根据配置文件 `periodic.conf` 中的配置来执行任务。ZFS 的 `daily_status_zfs_enable` 是一个配置项，用于启用 ZFS 的每日状态检查。

```
daily_status_zfs_enable="YES"
```

在 `periodic(8)` 中，我们可以配置 `"all pools are healthy."` 来检查所有池的健康状态。使用 `zpool status -x` 命令可以查看池的状态。

在 `periodic.conf` 中，我们可以配置 `daily_status_zfs_zpool_list` 来指定要检查的池。例如，我们可以配置 `daily_status_zfs_zpool_list="yes"` 来检查所有池。

FreeBSD 的 `periodic(8)` 是用于定期执行任务的工具。它可以根据配置文件 `periodic.conf` 中的配置来执行任务。ZFS 的 `daily_scrub_zfs_enable` 是一个配置项，用于启用 ZFS 的每日数据完整性检查。

```
daily_scrub_zfs_enable="YES"
```

FreeBSD 的 `periodic(8)` 是用于定期执行任务的工具。它可以根据配置文件 `periodic.conf` 中的配置来执行任务。ZFS 的 `daily_scrub_zfs_pools` 是一个配置项，用于指定要检查的池。

```
daily_scrub_zfs_pools="zroot prod test"
```

在 `periodic.conf` 中，我们可以配置 `daily_scrub_zfs_default_threshold` 来指定默认的阈值。

```
daily_scrub_zfs_default_threshold="10"
```

在 `periodic.conf` 中，我们可以配置 `daily_scrub_zfs_${poolname}_threshold` 来指定特定池的阈值。

```
daily_scrub_zfs_prod_threshold="7"
```

在 `periodic.conf` 中，我们可以配置 `daily_scrub_zfs_test_threshold` 来指定特定池的阈值。

? ?? (Removing Pools)

在 `zpool destroy` 命令中，我们可以指定要删除的池。

```
$ zpool destroy test
```

zpool 命令使用 zpool destroy 来删除 zpool。删除 zpool 时，zpool 中的所有数据将被永久删除。删除 zpool 时，zpool 中的所有数据将被永久删除。

删除 zpool 时，zpool 中的所有数据将被永久删除。删除 zpool 时，zpool 中的所有数据将被永久删除。

Zpool ?? ??? (Zpool Feature Flags)

ZFS 池中的特性标志（feature flags）用于控制池的行为。特性标志分为三类：池特性标志（pool features）、文件系统特性标志（filesystem features）和卷特性标志（volume features）。池特性标志用于控制池的行为，例如，特性标志 `async_destroy` 用于控制池的异步删除行为。

池特性标志（pool features）用于控制池的行为。池特性标志分为三类：池特性标志（pool features）、文件系统特性标志（filesystem features）和卷特性标志（volume features）。池特性标志用于控制池的行为，例如，特性标志 `async_destroy` 用于控制池的异步删除行为。

OpenZFS 池中的特性标志（feature flags）用于控制池的行为。OpenZFS 池中的特性标志分为三类：池特性标志（pool features）、文件系统特性标志（filesystem features）和卷特性标志（volume features）。池特性标志用于控制池的行为，例如，特性标志 `async_destroy` 用于控制池的异步删除行为。

FreeBSD 池中的特性标志（feature flags）用于控制池的行为。FreeBSD 池中的特性标志分为三类：池特性标志（pool features）、文件系统特性标志（filesystem features）和卷特性标志（volume features）。池特性标志用于控制池的行为，例如，特性标志 `async_destroy` 用于控制池的异步删除行为。

池特性标志（pool features）用于控制池的行为。池特性标志分为三类：池特性标志（pool features）、文件系统特性标志（filesystem features）和卷特性标志（volume features）。池特性标志用于控制池的行为，例如，特性标志 `async_destroy` 用于控制池的异步删除行为。

?? ??? ?? (Viewing Feature Flags)

要查看池中的特性标志，可以使用 `zpool get` 命令。例如，要查看池 `zroot` 中的特性标志，可以使用以下命令：

```
$ zpool get all zroot | grep feature
zroot  feature@async_destroy  enabled  local
zroot  feature@empty_bpobj     active   local
zroot  feature@lz4_compress    active   local
...
```

[illegible]

$\frac{1}{2} + \frac{1}{3} = \frac{3}{6} + \frac{2}{6} = \frac{5}{6}$

0 0000 0000 00 00 00 00 0000 00 00 00 0000 . 0000 00 0
 00 0000 00 00000 0 00 00 0 0000 . 00 00000 00 00 00
 0000 00 00 00 00 00 00 00 0000 0000 .

此选项默认为 "read-only compatible"。如果设置为 "read-write"，则允许在只读模式下写入数据，但可能会导致数据不一致。如果设置为 "read-only"，则只允许读取数据。

[illegible]