

?? 3. ?(Pools)

[illegible]

ZFS ??

[illegible]

ZFSは、ファイルシステム、ボリューム管理、RAID機能、データ圧縮、暗号化などを統合的に提供し、データの信頼性とパフォーマンスを向上させる。ZFSは、データの整合性を保つために、チェックサムとエラー訂正機能を備えている。また、ZFSは、データのバックアップと復元を簡便に行うことができる。ZFSは、LinuxやFreeBSDなどのオペレーティングシステムで広く利用されている。

00 00 00 00 00 , ZFS 00 00000 0000 000 ! 00 000 000 000
 00000 . ZFS 0000000 00 000 000 00 000 000 00 0000 00000 . ZFS
 0 0000 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 0000000 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 00
 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 64KB 的 大 小 来 存 储 的， 而 txg 的 大 小 是 64KB， 这 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 d[文件系统] 文件系统 格式 文件系统 格式 文件系统 格式 文件系统 格式 文件系统 格式 文件系统 格式 . 文件系统 格式 文件系统 格式 UFS2 文件系统 格式 extfs 文件系统 格式 文件系统 格式 .

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

[illegible]

RAID 10 的寫入速度會比 RAID 5 快，因為 RAID 10 的寫入是並行的，而 RAID 5 的寫入是串行的。RAID 10 的讀取速度也會比 RAID 5 快，因為 RAID 10 的讀取是並行的，而 RAID 5 的讀取是串行的。RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。RAID 10 的優點是它的讀取和寫入速度都比 RAID 5 快，而且它的容錯能力也比 RAID 5 強。

RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。RAID 10 的優點是它的讀取和寫入速度都比 RAID 5 快，而且它的容錯能力也比 RAID 5 強。

ZFS 的 RAID 10 的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都比 RAID 5 快，而且它的容錯能力也比 RAID 5 強。ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。

ZFS 的 RAID 10 的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都比 RAID 5 快，而且它的容錯能力也比 RAID 5 強。ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。

ZFS 的 RAID 10 的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都比 RAID 5 快，而且它的容錯能力也比 RAID 5 強。ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。

ZFS 的 RAID 10 的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都比 RAID 5 快，而且它的容錯能力也比 RAID 5 強。ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。

ZFS 的 RAID 10 的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。ZFS 的 RAID 10 的優點是它的讀取和寫入速度都比 RAID 5 快，而且它的容錯能力也比 RAID 5 強。ZFS 的 RAID 10 的缺點是它的儲存空間利用率只有 50%，而 RAID 5 的儲存空間利用率是 66%。

?? (Viewing Pools)

要查看 zpool 的狀態，可以使用 `zpool list` 命令。

```
$ zpool list
NAME      SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
db        2.72T  1.16G   2.72T        -     0%   0%  1.00x  ONLINE  -
zroot     920G   17.3G   903G        -     2%   1%  1.00x  ONLINE  -
```


在 2015 年之前，ZFS 的 VDEV 只能由 RAID-Z 组成。这意味着，如果你有一个 VDEV，它必须包含至少三个磁盘，并且它们必须以 RAID-Z 的方式配置。这限制了 VDEV 的灵活性和性能。

随着 OpenZFS 的发展，VDEV 的组成方式变得更加灵活。现在，你可以使用 RAID-Z3 来配置 VDEV，这意味着你可以使用更多的磁盘，并且可以容忍更多的磁盘故障。此外，你还可以使用 ZFS 的快照功能来保护你的数据。

ZFS 的 VDEV 可以包含任意数量的磁盘，并且可以配置为不同的 RAID 级别。这使得 ZFS 非常适合用于需要高可靠性和高性能的存储系统。

VDEV ?? (Removing VDEVs)

在 ZFS 中，VDEV 是存储数据的基本单位。你可以添加或删除 VDEV，但需要注意的是，删除 VDEV 可能会导致数据丢失。因此，在删除 VDEV 之前，请务必备份你的数据。

如果你需要删除 VDEV，可以使用 `zpool destroy` 命令。这将删除 VDEV 及其包含的所有数据。

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

ZFS 的 VDEV 可以包含任意数量的磁盘，并且可以配置为不同的 RAID 级别。这使得 ZFS 非常适合用于需要高可靠性和高性能的存储系统。

在 ZFS 中，VDEV 的组成方式变得更加灵活。现在，你可以使用 RAID-Z3 来配置 VDEV，这意味着你可以使用更多的磁盘，并且可以容忍更多的磁盘故障。

??? ?? (Partition Alignment)

在 ZFS 中，VDEV 的组成方式变得更加灵活。现在，你可以使用 RAID-Z3 来配置 VDEV，这意味着你可以使用更多的磁盘，并且可以容忍更多的磁盘故障。

在 ZFS 中，VDEV 的组成方式变得更加灵活。现在，你可以使用 RAID-Z3 来配置 VDEV，这意味着你可以使用更多的磁盘，并且可以容忍更多的磁盘故障。

`/etc/sysctl.conf` 文件 添加 `sysctl vfs.zfs.min_auto_ashift=12` 并 重启 ashift 服务。

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

在 系统 启动 时 添加 , 在 系统 启动 时 添加 `/etc/sysctl.conf` 文件 并 重启 服务。

在 系统 启动 时 添加 FreeBSD 10.1 系统 启动 时 添加 。在 FreeBSD 系统 启动 时 , sysctl 文件 添加 ashift 文件 添加 。

?? FreeBSD Ashift (Older FreeBSD Ashift)

10.1 系统 FreeBSD 系统 启动 时 添加 ashift sysctl 文件 添加 , ZFS 文件 添加 文件 添加 文件 添加 。在 系统 启动 时 添加 , 在 系统 启动 时 添加 文件 添加 。

在 系统 启动 时 添加 文件 添加 文件 添加 。FreeBSD 系统 GEOM 文件 **gnop(8)** 文件 添加 文件 添加 文件 添加 。gnop 文件 添加 文件 添加 文件 添加 文件 添加 (在 系统 启动 时 添加 文件 添加)。"在 系统 启动 时 添加 文件 添加 , 4096 文件 添加 文件 添加 。"在 gnop 文件 添加 文件 添加 。在 系统 启动 时 添加 zpool 文件 添加 。在 `/dev/gpt/zfs0` 文件 添加 gnop 文件 添加 。

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

在 系统 启动 时 `/dev/gpt/zfs0.nop` 文件 添加 。在 系统 启动 时 VDEV 文件 添加 文件 添加 ZFS 文件 添加 VDEV 文件 添加 文件 添加 。在 系统 启动 时 添加 文件 添加 ZFS 文件 添加 文件 添加 文件 添加 , 在 系统 启动 时 添加 文件 添加 文件 添加 文件 添加 。

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

gnop(8) 文件 添加 文件 添加 文件 添加 文件 添加 。在 **gnop(8)** 文件 添加 文件 添加 文件 添加 ZFS 文件 添加 文件 添加 文件 添加 文件 添加 。在 系统 启动 时 添加 文件 添加 ZFS 文件 添加 文件 添加 文件 添加 文件 添加 。

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

zpool(8) 文件 添加 文件 添加 文件 添加 文件 添加 。在 **zpool(8)** 文件 添加 文件 添加 VDEV 文件 添加 文件 添加 文件 添加 文件 添加 , 在 系统 启动 时 5 文件 添加 文件 添加 。在 RAID-Z 文件 添加 文件 添加 , 在 系统 启动 时 2 文件 添加 文件 添加 VDEV 文件 添加 文件 添加 。

在 系统 启动 时 添加 ashift 文件 添加 文件 添加 文件 添加 文件 添加 。在 系统 启动 时 添加 文件 添加 文件 添加 文件 添加 文件 添加 文件 添加 文件 添加 文件 添加 文件 添加 文件 添加 "set ashift" 文件 添加 。

?? ???? (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    . 

????? ? (Striped Pools)

```

[[{"id": 1, "name": "Alice", "age": 30, "gender": "F", "height": 160, "weight": 55, "hair_color": "Brown", "eye_color": "Blue"},
{"id": 2, "name": "Bob", "age": 25, "gender": "M", "height": 175, "weight": 70, "hair_color": "Black", "eye_color": "Green"},
{"id": 3, "name": "Charlie", "age": 35, "gender": "M", "height": 180, "weight": 80, "hair_color": "Grey", "eye_color": "Blue"},
{"id": 4, "name": "Diana", "age": 28, "gender": "F", "height": 165, "weight": 60, "hair_color": "Blonde", "eye_color": "Green"},
{"id": 5, "name": "Eve", "age": 22, "gender": "F", "height": 155, "weight": 50, "hair_color": "Red", "eye_color": "Blue"},
{"id": 6, "name": "Frank", "age": 40, "gender": "M", "height": 190, "weight": 90, "hair_color": "Black", "eye_color": "Brown"},
{"id": 7, "name": "Grace", "age": 32, "gender": "F", "height": 170, "weight": 65, "hair_color": "Brown", "eye_color": "Green"},
{"id": 8, "name": "Henry", "age": 27, "gender": "M", "height": 178, "weight": 75, "hair_color": "Black", "eye_color": "Blue"},
{"id": 9, "name": "Ivy", "age": 24, "gender": "F", "height": 162, "weight": 58, "hair_color": "Blonde", "eye_color": "Brown"},
{"id": 10, "name": "Jack", "age": 38, "gender": "M", "height": 185, "weight": 85, "hair_color": "Grey", "eye_color": "Green"}]]

```

```
$ 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 . 個 磁 磁頭 VDEV個 .

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

?? ? (Mirrored Pools)

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

個 **zpool create** 個 個 磁頭 . **mirror** 個 個 個 個 個
磁頭 . 個 磁頭 個 磁頭 ashift個 磁頭 .

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

zpool status 個 個 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 個 個 mirror-0個 個 磁頭 . mirror-0 個 VDEV個 . 個 VDEV個
個 個 , 個 **gpt/zfs0** **gpt/zfs1** 個 磁頭 . 個 個 個 個 磁頭 個 個
個 磁頭 . 個 個 個 個 個 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` 选项来指定 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` 命令的选项包括：

- `reflect`：指定 RAID 级别为 RAID-Z。
- `mirror`：指定 RAID 级别为 RAID-Z。
- `raidz`：指定 RAID 级别为 RAID-Z。

 RAID-Z 池的创建命令如下所示：
 `zpool create` 命令的选项包括：

- `reflect`：指定 RAID 级别为 RAID-Z。
- `mirror`：指定 RAID 级别为 RAID-Z。
- `raidz`：指定 RAID 级别为 RAID-Z。

 RAID-Z 池的创建命令如下所示：
 `zpool create` 命令的选项包括：

- `reflect`：指定 RAID 级别为 RAID-Z。
- `mirror`：指定 RAID 级别为 RAID-Z。
- `raidz`：指定 RAID 级别为 RAID-Z。

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` 命令的选项包括：

- `reflect`：指定 RAID 级别为 RAID-Z。
- `mirror`：指定 RAID 级别为 RAID-Z。
- `raidz`：指定 RAID 级别为 RAID-Z。

 RAID-Z 池的创建命令如下所示：
 `zpool create` 命令的选项包括：

- `reflect`：指定 RAID 级别为 RAID-Z。
- `mirror`：指定 RAID 级别为 RAID-Z。
- `raidz`：指定 RAID 级别为 RAID-Z。

```
$ 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。
 `zpool status` 命令的输出如下所示：

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

```
$ 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。
 `zpool status` 命令的输出如下所示：

```
$ 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 [[[[. [[[[[[VDEV [[[[[[[[?

?? VDEV ? (Multi-VDEV Pools)

[[[[VDEV [[[[[[[[[[[[. *mirror, raidz, raidz2, raidz3* [[[[**zpool(8)** [[[[
VDEV [[[[[[[[. [[[[[[[[[[[[[[[[[[VDEV [[
[[[[[[[[. [[[[[[[[[[[[**zpool(8)** [[[[VDEV [[[[.

[[[[[[[[[[RAID-10 [[[[[[[[[[[[[[[[[[[[
[[[[. [[[[[[[[[[.

```
$ 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**
[[**gpt/zfs1** [[[[. [[[[[[[[[[[[[[[[[[[[[[. **mirror** [[[[[[
[[[[[[VDEV [[[[[[VDEV [[[[[[**zpool(8)** [[[[. [[[[VDEV
[[[[[[[[[[, [[**gpt/zfs2** [[**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，但 ZFS 的 RAID 功能更强大。ZFS 的 RAID-Z 功能可以创建 RAID-Z1、RAID-Z2 和 RAID-Z3。RAID-Z1 使用奇偶校验，RAID-Z2 使用双奇偶校验，RAID-Z3 使用三奇偶校验。

```
$ 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。我们使用 **gpt/zfs0, gpt/zfs1, gpt/zfs2** 三个 VDEV 来创建 RAID-Z1。我们使用 **gpt/zfs3, gpt/zfs4, gpt/zfs5** 三个 VDEV 来创建 RAID-Z1。zpool 命令可以创建 RAID-Z1。

```
$ 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。我们使用 RAID-Z1 来创建 RAID-Z1。

我们使用 RAIDZ 来创建 RAID-Z1。我们使用 RAIDZ 来创建 RAID-Z1。我们使用 RAIDZ 来创建 RAID-Z1。

我们使用 RAIDZ 来创建 RAID-Z1。我们使用 RAIDZ 来创建 RAID-Z1。我们使用 RAIDZ 来创建 RAID-Z1。

?? ?? ?? (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) 这个命令用于创建和管理 ZFS 存储池。它接受多个设备或虚拟设备 (VDEV) 作为参数。如果设备已经属于另一个池，使用 **zpool create -f** 可以强制覆盖。ZFS 支持多种 RAID 级别，如 RAIDZ、RAIDZ2、RAIDZ3、Mirror 和 RAID0。创建池时，需要指定池名和 VDEV 列表。

ZFS 池的创建和配置可以通过 **zpool create** 命令完成。如果设备已经存在，使用 **-f** 选项可以强制覆盖。ZFS 池的命名通常遵循一定的命名规范，如 `poolname`。

??? ??? (Reusing Providers)

在创建 ZFS 池时，如果指定的 VDEV 已经属于另一个池，会报错。使用 **zpool create -f** 可以强制覆盖。ZFS 池的命名通常遵循一定的命名规范，如 `poolname`。

```
$ 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'
```

在创建 ZFS 池时，如果指定的 VDEV 已经属于另一个池，会报错。使用 **zpool create -f** 可以强制覆盖。ZFS 池的命名通常遵循一定的命名规范，如 `poolname`。

在创建 ZFS 池时，如果指定的 VDEV 已经属于另一个池，会报错。使用 **zpool create -f** 可以强制覆盖。ZFS 池的命名通常遵循一定的命名规范，如 `poolname`。

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

ZFS 池的创建和配置可以通过 **zpool create** 命令完成。如果设备已经存在，使用 **-f** 选项可以强制覆盖。ZFS 池的命名通常遵循一定的命名规范，如 `poolname`。

? ??? (Pool Integrity)

ZFS 池的完整性检查可以通过 **fsck(8)** 命令完成。ZFS 池的命名通常遵循一定的命名规范，如 `poolname`。

ZFS ??? (ZFS Integrity)

在创建 ZFS 池时，如果指定的 VDEV 已经属于另一个池，会报错。使用 **zpool create -f** 可以强制覆盖。ZFS 池的命名通常遵循一定的命名规范，如 `poolname`。


```
$ 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 scrub -s 命令用于强制启动池的扫描。它会在池的根目录下启动扫描，并显示扫描进度。

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

zpool scrub -s 命令会在池的根目录下启动扫描，并显示扫描进度。

??? ?? (Scrub Frequency)

ZFS 池的扫描频率可以通过 zpool scrub 命令来设置。默认情况下，池的扫描频率是每周一次。可以通过 zpool scrub 命令来设置扫描频率，例如：

zpool scrub 命令会在池的根目录下启动扫描，并显示扫描进度。

? ?? (Pool Properties)

ZFS 池的属性可以通过 zpool get 命令来查看。它显示了池的名称、池的大小、池的扫描进度、池的扫描速度、池的扫描时间以及池的扫描结果。

zpool get 命令会在池的根目录下启动扫描，并显示扫描进度。

? ?? ?? (Viewing Pool Properties)

zpool get 命令会在池的根目录下启动扫描，并显示扫描进度。

```
$ zpool get all zroot
NAME    PROPERTY  VALUE     SOURCE
zroot    SIZE      17.3G     -
```

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

[illegible][illegible]

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ **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
```

我們來看看 zpool 的屬性，以及 zpool 的運作原理。

zpool 的運作原理，是將多個物理磁碟，組合成一個邏輯磁碟。這個邏輯磁碟，可以當作一個普通的磁碟使用。zpool 的運作原理，是將多個物理磁碟，組合成一個邏輯磁碟。這個邏輯磁碟，可以當作一個普通的磁碟使用。zpool 的運作原理，是將多個物理磁碟，組合成一個邏輯磁碟。這個邏輯磁碟，可以當作一個普通的磁碟使用。

```
$ zpool create -o altroot=/mnt -0 canmount=off -m none zroot /dev/gpt/disk0
```

這裡的 **altroot** 是指 /mnt，表示 zpool 的根目錄。而 **canmount** 是指 off，表示 zpool 不能被掛載。而 **off** 是指 off，表示 zpool 不能被掛載。而 **off** 是指 off，表示 zpool 不能被掛載。而 **off** 是指 off，表示 zpool 不能被掛載。

? (Pool History)

zpool 的歷史記錄，可以透過 zpool history 來查看。這裡的 zpool history 是指 zpool 的歷史記錄，可以透過 zpool history 來查看。這裡的 zpool history 是指 zpool 的歷史記錄，可以透過 zpool history 來查看。這裡的 zpool history 是指 zpool 的歷史記錄，可以透過 zpool history 來查看。

我們來看看 zpool history 的輸出結果。

```
$ zpool history zroot
History for 'zroot':
2014-01-07.04:12:05 zpool create -o altroot=/mnt -0 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
...
```

這裡的 FreeBSD 是指 FreeBSD 的系統，而 zfs 是指 zfs 的系統。這裡的 FreeBSD 是指 FreeBSD 的系統，而 zfs 是指 zfs 的系統。這裡的 FreeBSD 是指 FreeBSD 的系統，而 zfs 是指 zfs 的系統。這裡的 FreeBSD 是指 FreeBSD 的系統，而 zfs 是指 zfs 的系統。

我們來看看 zpool 的屬性，以及 zpool 的運作原理。

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

comment 是指 comment 的屬性，而 zpool 是指 zpool 的系統。comment 是指 comment 的屬性，而 zpool 是指 zpool 的系統。comment 是指 comment 的屬性，而 zpool 是指 zpool 的系統。comment 是指 comment 的屬性，而 zpool 是指 zpool 的系統。

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` 工具可以根据配置文件 `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` 工具可以根据配置文件 `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。例如，我们可以配置 `daily_scrub_zfs_pools="zroot prod test"` 来检查 `zroot`、`prod` 和 `test` 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。例如，我们可以配置 `daily_scrub_zfs_default_threshold="10"` 来设置阈值为 10。

```
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。例如，我们可以配置 `daily_scrub_zfs_prod_threshold="7"` 来设置 `prod` 池的阈值为 7。

```
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。例如，我们可以配置 `daily_scrub_zfs_test_threshold="10"` 来设置 `test` 池的阈值为 10。

? ?? (Removing Pools)

在 `periodic.conf` 中，我们可以配置 `zpool destroy` 来删除池。例如，我们可以配置 `zpool destroy test` 来删除 `test` 池。

```
$ zpool destroy test
```

zpool 的 destroy 命令会强制删除 zpool，即使它包含数据。如果 zpool 处于 active 状态，则必须先将其设置为 standby 状态。如果 zpool 处于 standby 状态，则可以直接删除。如果 zpool 处于 active 状态，则必须先将其设置为 standby 状态，然后再删除。

zpool 的 destroy 命令会强制删除 zpool，即使它包含数据。如果 zpool 处于 active 状态，则必须先将其设置为 standby 状态。如果 zpool 处于 standby 状态，则可以直接删除。如果 zpool 处于 active 状态，则必须先将其设置为 standby 状态，然后再删除。

Zpool ?? ??? (Zpool Feature Flags)

ZFS 的 feature flags 是用于控制 ZFS 功能的标志。它们可以分为三类：enabled、active 和 disabled。enabled 标志是默认启用的，active 标志是需要手动启用的，disabled 标志是默认禁用的。ZFS 的 feature flags 可以通过 zpool 命令进行管理。

zpool 命令的 feature 子命令用于管理 ZFS 的 feature flags。它包括 feature on、feature off、feature list 和 feature status 等子命令。feature on 用于启用 feature flag，feature off 用于禁用 feature flag，feature list 用于列出所有 feature flags，feature status 用于查看 feature flags 的状态。

OpenZFS 的 feature flags 是用于控制 OpenZFS 功能的标志。它们可以分为三类：enabled、active 和 disabled。enabled 标志是默认启用的，active 标志是需要手动启用的，disabled 标志是默认禁用的。OpenZFS 的 feature flags 可以通过 zpool 命令进行管理。

FreeBSD 的 zpool-features(7) 手册页提供了关于 ZFS 的 feature flags 的详细信息。它包括 feature on、feature off、feature list 和 feature status 等子命令。FreeBSD 的 zpool-features(7) 手册页还提供了关于 feature flags 的详细说明。

zpool 命令的 feature 子命令用于管理 ZFS 的 feature flags。它包括 feature on、feature off、feature list 和 feature status 等子命令。feature on 用于启用 feature flag，feature off 用于禁用 feature flag，feature list 用于列出所有 feature flags，feature status 用于查看 feature flags 的状态。

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

zpool 命令的 get 子命令用于查看 zpool 的属性。它包括 feature 属性，用于查看 feature flags 的状态。feature 属性的值可以是 enabled、active 或 disabled。

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

1. 在创建 ZFS 池之前，需要确保所有磁盘都已正确识别。使用 `zpool list` 命令可以查看当前系统中已有的 ZFS 池。

2. 使用 `zpool create` 命令来创建新的 ZFS 池。例如，要创建一个名为 `my_pool` 的池，可以使用以下命令：

```
zpool create my_pool /dev/disk/by-id/ata-XXXXXX /dev/disk/by-id/ata-YYYYYY
```

3. 创建池时，可以指定一些选项，如 `-d` 用于指定数据块大小，`-o` 用于指定其他属性。

4. 创建完成后，可以使用 `zpool status` 命令来检查池的状态。

5. 如果需要，可以在池上创建文件系统，例如使用 `zfs create` 命令。