NAME

zpool **-** configure ZFS storage pools

SYNOPSIS

zpool -?V
zpool version
zpool subcommand [*arguments*]

DESCRIPTION

The **zpool** command configures ZFS storage pools. A storage pool is a collection of devices that provides physical storage and data replication for ZFS datasets. All datasets within a storage pool share the same space. See zfs(8) for information on managing datasets.

For an overview of creating and managing ZFS storage pools see the zpoolconcepts(7) manual page.

SUBCOMMANDS

All subcommands that modify state are logged persistently to the pool in their original form.

The **zpool** command provides subcommands to create and destroy storage pools, add capacity to storage pools, and provide information about the storage pools. The following subcommands are supported:

zpool -?
Displays a help message.

zpool -V, **--version**

zpool version
Displays the software version of the **zpool** userland utility and the ZFS kernel module.

Creation

zpool-create(8)
Creates a new storage pool containing the virtual devices specified on the command line.

zpool-initialize(8)
Begins initializing by writing to all unallocated regions on the specified devices, or all eligible devices in the pool if no individual devices are specified.

Destruction

zpool-destroy(8)
Destroys the given pool, freeing up any devices for other use.

zpool-labelclear(8)
: Removes ZFS label information from the specified *device*.

Virtual Devices

zpool-attach(8)/zpool-detach(8)
: Converts a non-redundant disk into a mirror, or increases the redundancy level of an existing mirror (**attach**), or performs the inverse operation (**detach**).

zpool-add(8)/zpool-remove(8)
: Adds the specified virtual devices to the given pool, or removes the specified device from the pool.

zpool-replace(8)
: Replaces an existing device (which may be faulted) with a new one.

zpool-split(8)
: Creates a new pool by splitting all mirrors in an existing pool (which decreases its redundancy).

Properties

Available pool properties listed in the zpoolprops(7) manual page.

zpool-list(8)
: Lists the given pools along with a health status and space usage.

zpool-get(8)/zpool-set(8)
: Retrieves the given list of properties (or all properties if **all** is used) for the specified storage pool(s).

Monitoring

zpool-status(8)
: Displays the detailed health status for the given pools.

zpool-iostat(8)
: Displays logical I/O statistics for the given pools/vdevs. Physical I/O operations may be observed via iostat(1).

zpool-events(8)
: Lists all recent events generated by the ZFS kernel modules. These events are consumed by the zed(8) and used to automate administrative tasks such as replacing a failed device with a hot spare. That manual page also describes the subclasses and event payloads that can be

generated.

zpool-history(8)
: Displays the command history of the specified pool(s) or all pools if no pool is specified.

Maintenance

zpool-scrub(8)
: Begins a scrub or resumes a paused scrub.

zpool-checkpoint(8)
: Checkpoints the current state of *pool*, which can be later restored by **zpool import --rewind-to-checkpoint**.

zpool-trim(8)
: Initiates an immediate on-demand TRIM operation for all of the free space in a pool. This operation informs the underlying storage devices of all blocks in the pool which are no longer allocated and allows thinly provisioned devices to reclaim the space.

zpool-sync(8)
: This command forces all in-core dirty data to be written to the primary pool storage and not the ZIL. It will also update administrative information including quota reporting. Without arguments, **zpool sync** will sync all pools on the system. Otherwise, it will sync only the specified pool(s).

zpool-upgrade(8)
: Manage the on-disk format version of storage pools.

zpool-wait(8)
: Waits until all background activity of the given types has ceased in the given pool.

Fault Resolution

zpool-offline(8)/zpool-online(8)
: Takes the specified physical device offline or brings it online.

zpool-resilver(8)
: Starts a resilver. If an existing resilver is already running it will be restarted from the beginning.

zpool-reopen(8)
: Reopen all the vdevs associated with the pool.

zpool-clear(8)
Clears device errors in a pool.

Import & Export

zpool-import(8)
Make disks containing ZFS storage pools available for use on the system.

zpool-export(8)
Exports the given pools from the system.

zpool-reguid(8)
Generates a new unique identifier for the pool.

EXIT STATUS

The following exit values are returned:

0
Successful completion.

1
An error occurred.

2
Invalid command line options were specified.

EXAMPLES

Example 1: Creating a RAID-Z Storage Pool

The following command creates a pool with a single raidz root vdev that consists of six disks:

```
# zpool create tank raidz sda sdb sdc sdd sde sdf
```

Example 2: Creating a Mirrored Storage Pool

The following command creates a pool with two mirrors, where each mirror contains two disks:

```
# zpool create tank mirror sda sdb mirror sdc sdd
```

Example 3: Creating a ZFS Storage Pool by Using Partitions

The following command creates a non-redundant pool using two disk partitions:

```
# zpool create tank sda1 sdb2
```

Example 4: Creating a ZFS Storage Pool by Using Files

The following command creates a non-redundant pool using files. While not recommended, a pool based on files can be useful for experimental purposes.

```
# zpool create tank /path/to/file/a /path/to/file/b
```

Example 5: Making a non-mirrored ZFS Storage Pool mirrored

The following command converts an existing single device *sda* into a mirror by attaching a second device to it, *sdb*.

```
# zpool attach tank sda sdb
```

Example 6: Adding a Mirror to a ZFS Storage Pool

The following command adds two mirrored disks to the pool *tank*, assuming the pool is already made up of two-way mirrors. The additional space is immediately available to any datasets within the pool.

```
# zpool add tank mirror sda sdb
```

Example 7: Listing Available ZFS Storage Pools

The following command lists all available pools on the system. In this case, the pool *zion* is faulted due to a missing device. The results from this command are similar to the following:

```
# zpool list
NAME    SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
rpool  19.9G  8.43G  11.4G         -    33%    42%  1.00x  ONLINE  -
tank   61.5G  20.0G  41.5G         -    48%    32%  1.00x  ONLINE  -
zion       -      -      -         -      -      -      -  FAULTED -
```

Example 8: Destroying a ZFS Storage Pool

The following command destroys the pool *tank* and any datasets contained within:

```
# zpool destroy -f tank
```

Example 9: Exporting a ZFS Storage Pool

The following command exports the devices in pool *tank* so that they can be relocated or later imported:

```
# zpool export tank
```

Example 10: Importing a ZFS Storage Pool

The following command displays available pools, and then imports the pool *tank* for use on the system. The results from this command are similar to the following:

```
# zpool import
  pool: tank
    id: 15451357997522795478
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        tank        ONLINE
          mirror    ONLINE
            sda     ONLINE
            sdb     ONLINE
```

# **zpool import** *tank*

Example 11: Upgrading All ZFS Storage Pools to the Current Version

The following command upgrades all ZFS Storage pools to the current version of the software:

```
# zpool upgrade -a
This system is currently running ZFS version 2.
```

Example 12: Managing Hot Spares

The following command creates a new pool with an available hot spare:

```
# zpool create tank mirror sda sdb spare sdc
```

If one of the disks were to fail, the pool would be reduced to the degraded state. The failed device can be replaced using the following command:

```
# zpool replace tank sda sdd
```

Once the data has been resilvered, the spare is automatically removed and is made available for use should another device fail. The hot spare can be permanently removed from the pool using the following command:

```
# zpool remove tank sdc
```

Example 13: Creating a ZFS Pool with Mirrored Separate Intent Logs

The following command creates a ZFS storage pool consisting of two, two-way mirrors and mirrored log devices:

```
# zpool create pool mirror sda sdb mirror sdc sdd log mirror sde sdf
```

Example 14: Adding Cache Devices to a ZFS Pool

The following command adds two disks for use as cache devices to a ZFS storage pool:

```
# zpool add pool cache sdc sdd
```

Once added, the cache devices gradually fill with content from main memory. Depending on the size of your cache devices, it could take over an hour for them to fill. Capacity and reads can be monitored using the **iostat** subcommand as follows:

```
# zpool iostat -v pool 5
```

Example 15: Removing a Mirrored top-level (Log or Data) Device

The following commands remove the mirrored log device **mirror-2** and mirrored top-level data device **mirror-1**.

Given this configuration:

```
              pool: tank
             state: ONLINE
             scrub: none requested
            config:

                     NAME        STATE     READ WRITE CKSUM
                     tank        ONLINE       0     0     0
                       mirror-0  ONLINE       0     0     0
                         sda     ONLINE       0     0     0
                         sdb     ONLINE       0     0     0
                       mirror-1  ONLINE       0     0     0
                         sdc     ONLINE       0     0     0
                         sdd     ONLINE       0     0     0
                     logs
                       mirror-2  ONLINE       0     0     0
                         sde     ONLINE       0     0     0
                         sdf     ONLINE       0     0     0
```

The command to remove the mirrored log *mirror-2* is:

```
# zpool remove tank mirror-2
```

The command to remove the mirrored data *mirror-1* is:

```
# zpool remove tank mirror-1
```

Example 16: Displaying expanded space on a device

The following command displays the detailed information for the pool *data*. This pool is comprised of a single raidz vdev where one of its devices increased its capacity by 10 GiB. In this example, the pool will not be able to utilize this extra capacity until all the devices under the raidz vdev have been expanded.

```
# zpool list -v data
NAME         SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
data        23.9G  14.6G  9.30G         -    48%    61%  1.00x  ONLINE  -
  raidz1    23.9G  14.6G  9.30G         -    48%
    sda         -      -      -         -      -
    sdb         -      -      -       10G      -
    sdc         -      -      -         -      -
```

Example 17: Adding output columns

Additional columns can be added to the **zpool status** and **zpool iostat** output with **-c**.

```
# zpool status -c vendor,model,size
  NAME     STATE  READ WRITE CKSUM vendor  model        size
  tank     ONLINE 0    0     0
  mirror-0 ONLINE 0    0     0
  U1       ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U10      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U11      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U12      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U13      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
  U14      ONLINE 0    0     0     SEAGATE ST8000NM0075 7.3T
```

```
# zpool iostat -vc size
             capacity     operations     bandwidth
pool        alloc   free   read  write   read  write  size
----------  -----  -----  -----  -----  -----  -----  ----
rpool       14.6G  54.9G      4     55   250K  2.69M
  sda1      14.6G  54.9G      4     55   250K  2.69M   70G
----------  -----  -----  -----  -----  -----  -----  ----
```

ENVIRONMENT VARIABLES

ZFS_ABORT Cause **zpool** to dump core on exit for the purposes of running **::findleaks**.

ZFS_COLOR Use ANSI color in **zpool status** and **zpool iostat** output.

ZPOOL_IMPORT_PATH The search path for devices or files to use with the pool. This is a colon-separated list of directories in which **zpool** looks for device nodes and files. Similar to the **-d** option in **zpool import**.

ZPOOL_IMPORT_UDEV_TIMEOUT_MS The maximum time in milliseconds that **zpool import** will wait for an expected device to be available.

ZPOOL_STATUS_NON_NATIVE_ASHIFT_IGNORE
If set, suppress warning about non-native vdev ashift in **zpool status**. The value is not used, only the presence or absence of the variable matters.

ZPOOL_VDEV_NAME_GUID Cause **zpool** subcommands to output vdev guids by default. This behavior is identical to the **zpool status -g** command line option.

ZPOOL_VDEV_NAME_FOLLOW_LINKS
Cause **zpool** subcommands to follow links for vdev names by default. This behavior is identical to the **zpool status -L** command line option.

ZPOOL_VDEV_NAME_PATH Cause **zpool** subcommands to output full vdev path names by default. This behavior is identical to the **zpool status -P** command line option.

ZFS_VDEV_DEVID_OPT_OUT Older OpenZFS implementations had issues when attempting to display pool config vdev names if a **devid** NVP value is present in the pool’s config.

For example, a pool that originated on illumos platform would have a **devid** value in the config and **zpool status** would fail when listing the config. This would also be true for future Linux-based pools.

A pool can be stripped of any **devid** values on import or prevented from adding them on **zpool create** or **zpool add** by setting **ZFS_VDEV_DEVID_OPT_OUT**.

ZPOOL_SCRIPTS_AS_ROOT Allow a privileged user to run **zpool status**/**iostat -c**. Normally, only unprivileged users are allowed to run **-c**.

ZPOOL_SCRIPTS_PATH The search path for scripts when running **zpool status**/**iostat -c**. This is a colon-separated list of directories and overrides the default *~/.zpool.d* and */etc/zfs/zpool.d* search paths.

ZPOOL_SCRIPTS_ENABLED Allow a user to run **zpool status**/**iostat -c**. If **ZPOOL_SCRIPTS_ENABLED** is not set, it is assumed that the user is allowed to run **zpool status**/**iostat -c**.

ZFS_MODULE_TIMEOUT Time, in seconds, to wait for */dev/zfs* to appear. Defaults to **10**, max **600** (10 minutes). If <**0**, wait forever; if **0**, don’t wait.

INTERFACE STABILITY

Evolving

SEE ALSO

zfs(4), zpool-features(7), zpoolconcepts(7), zpoolprops(7), zed(8), zfs(8), zpool-add(8), zpool-attach(8), zpool-checkpoint(8), zpool-clear(8), zpool-create(8), zpool-destroy(8), zpool-detach(8), zpool-events(8), zpool-export(8), zpool-get(8), zpool-history(8), zpool-import(8), zpool-initialize(8), zpool-iostat(8), zpool-labelclear(8), zpool-list(8), zpool-offline(8), zpool-online(8), zpool-reguid(8), zpool-remove(8), zpool-reopen(8), zpool-replace(8), zpool-resilver(8), zpool-scrub(8), zpool-set(8), zpool-split(8), zpool-status(8), zpool-sync(8), zpool-trim(8), zpool-upgrade(8), zpool-wait(8)