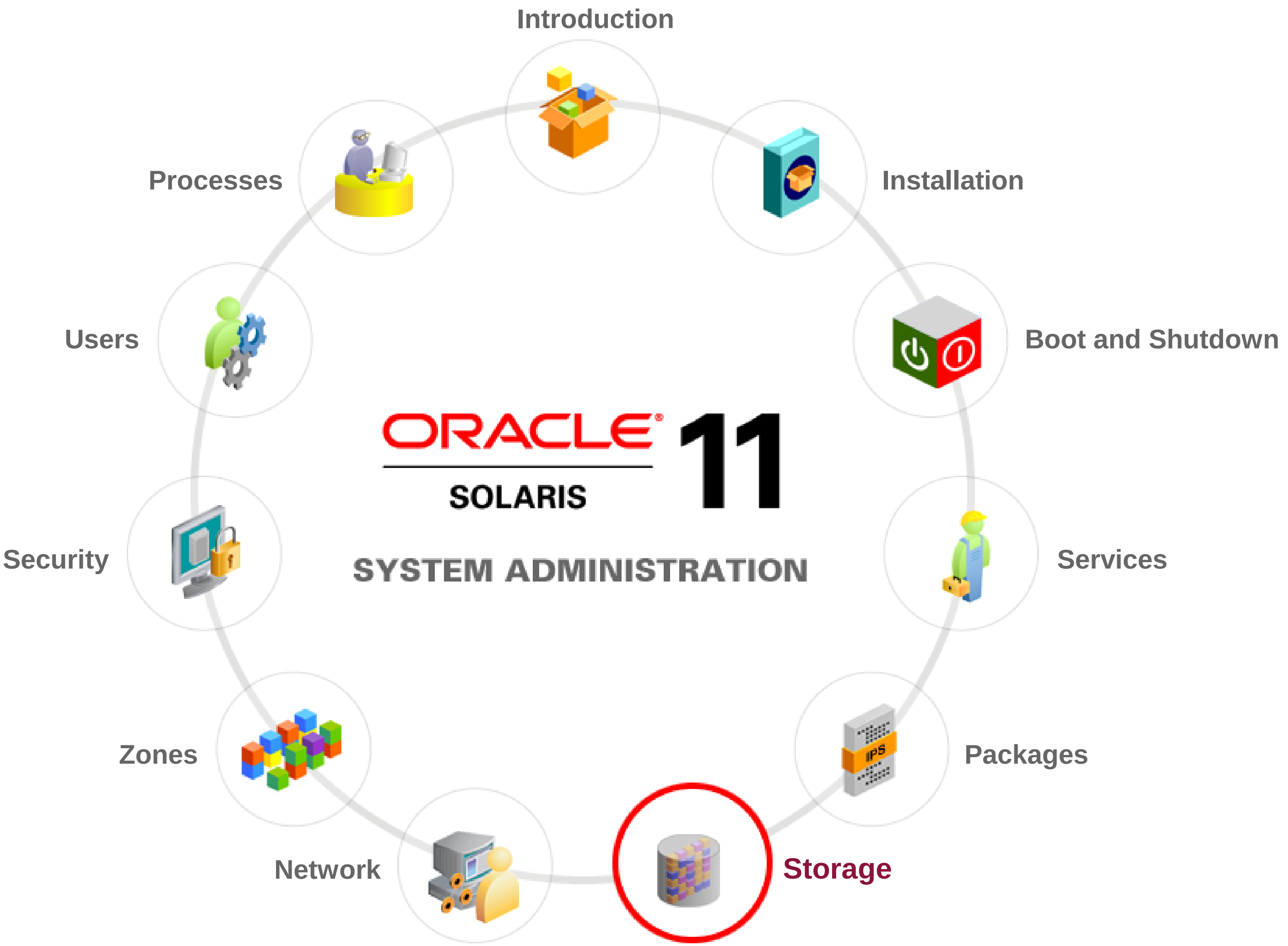
6

Managing Data by Using ZFS

ORACLE

Workflow Orientation



Objectives

After completing this lesson, you should be able to:

- Explain the role of ZFS in data management
- Administer ZFS storage pools
- Administer ZFS file systems
- Administer ZFS properties
- Administer ZFS snapshots and clones

Agenda

- **Introducing ZFS**
- Administering ZFS Storage Pools
- Administering ZFS File Systems
- Administering ZFS Properties
- Administering ZFS Snapshots and Clones

Importance of Data Management

Data management is required to ensure that:

- The appropriate type of storage pool configuration is selected that supports data redundancy and growth
- Data can be accessed, backed up, and restored quickly and easily

Introduction to ZFS

- Two products in one:
  - Volume manager
  - File system
- Most scalable file system ever:
  - 128-bit file system
  - Up to 256 trillion directory entries allowed
  - No limit to the number of file systems or number of files that can be contained within a file system
- Transactional file system:
  - File system state is always consistent on disk.
  - Data is never overwritten.
  - The file system can never be corrupted through accidental loss of power or a system crash.

ZFS Terms

Task	Description
Checksum	A 256-bit hash of the data in a file system block
Clone	A file system with contents that are identical to the contents of a ZFS snapshot
Dataset	A generic name for ZFS entities such as clones, file systems, snapshots, and volumes
Resilvering	The process of transferring data from one device to another
Scrub	A tool that validates and repairs the ZFS file system (including the metadata) while the file system is online and mounted
Snapshot	A read-only image of a file system or volume at a given point of time
zpool	A ZFS dataset that is mounted within the standard system namespace and that behaves like other traditional file systems

ZFS Storage Pools

- Storage pools are used to manage physical storage.
- No volume manager is required.
- Devices are aggregated into a storage pool.
- The storage pool:
  - Describes the physical characteristics of the storage
  - Acts as an arbitrary data store

ZFS Storage Pool Components

The following components can be used in a ZFS storage pool:

- Disks
- Slices
- Files
- Virtual devices

ZFS Storage Pool Components: Disks

- Any block device that is at least 128 MB in size
- Typically, a hard drive that is visible to the system in the `/dev/dsk` directory
- Whole disk (`c1t0d0`) or an individual slice (`c0t0d0s7`)
- Recommended mode of operation: Entire disk
  - No special formatting required
  - EFI label used to contain a single, large slice
  - Simplest way to create ZFS storage pools

ZFS Storage Pool Components: Disks

- To use whole disks:
  - Use the `/dev/dsk/cXtXdX` naming convention
  - Specify the disk by using either the full path (`/dev/dsk/c1t0d0`) or a shorthand name that consists of the device name within the `/dev/dsk` directory (`c1t0d0`)
- Examples of valid disk names:
  - `c1t0d0`
  - `/dev/dsk/c1t0d0`

ZFS Storage Pool Components: Slices

- Disks can be labeled with an SMI label.
- For the bootable ZFS root pool:
  - A disk must contain slices
  - An SMI label is required

Examples

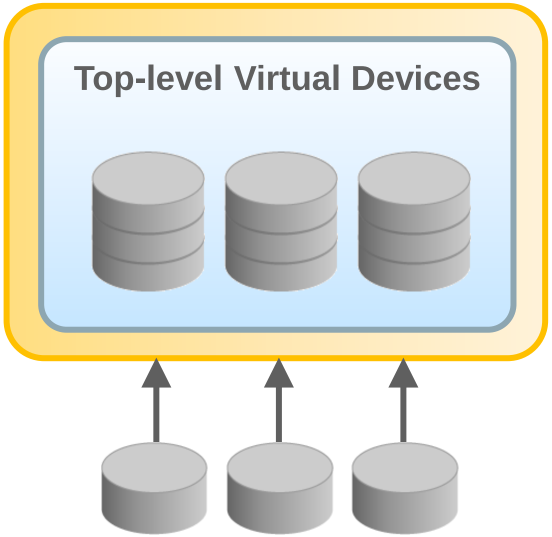
- On a SPARC-based system with a 72-GB disk:
  - 68 GB of usable space in slice 0
- On an x86-based system with a 72-GB disk:
  - 68 GB of usable space in slice 0
  - Small amount of boot information contained in slice 8
    - No administration required
    - Not changeable

ZFS Storage Pool Components: Files

- Not intended for production use
- Recommended for:
  - Testing
  - Simple experimentation
- Complete file path specification required
- File size: At least 128 MB

ZFS Storage Pool Components: Virtual Devices

- A virtual device is a logical device in a pool:
  - Disks
  - Disks slices
  - Files
- Virtual devices at the top of a configuration are referred to as "top-level virtual devices" or "top-level vdevs."
- Possible configurations:
  - Stand-alone (non-redundant)
  - Mirrored
  - RAID-Z



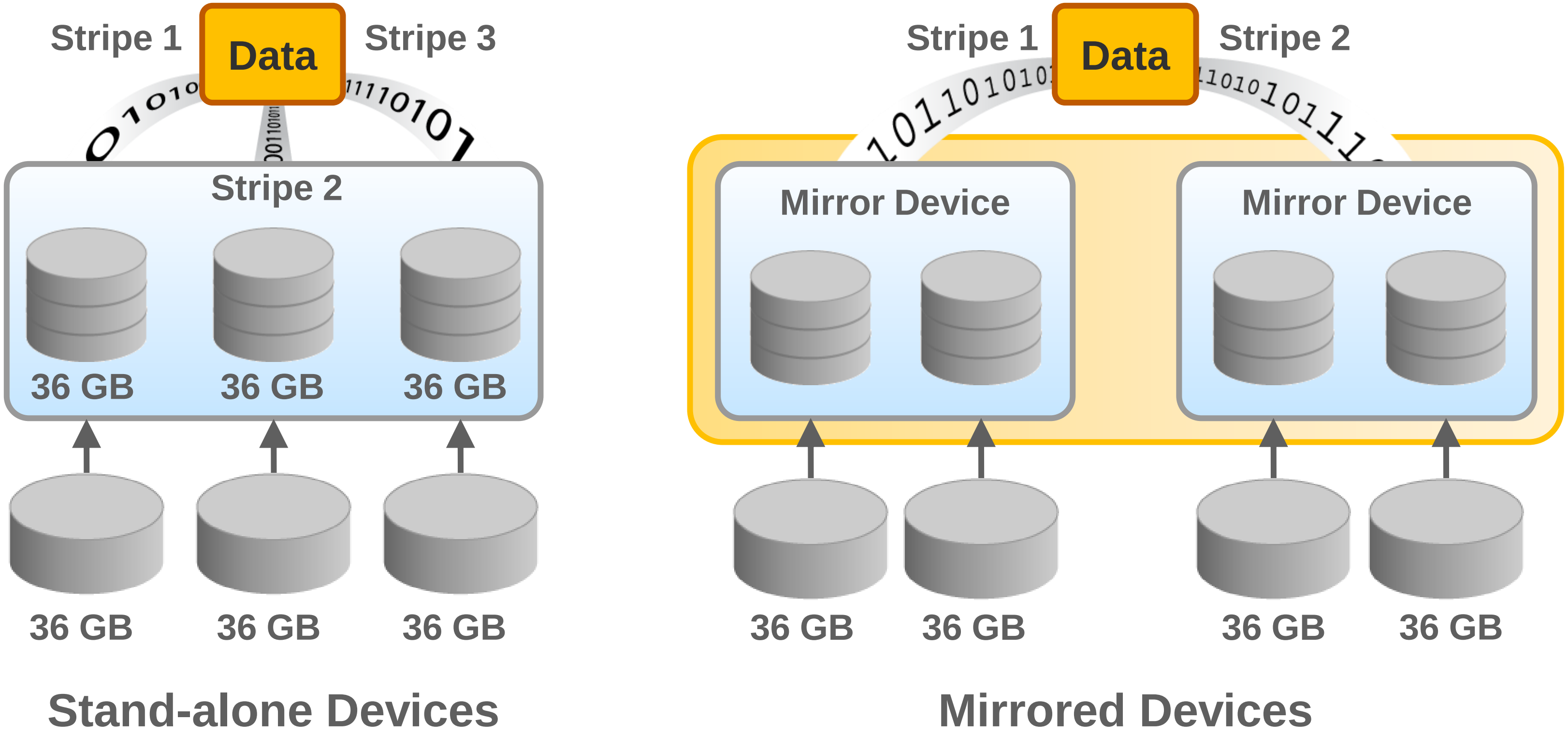
Virtual Devices and Dynamic Striping

- Data is dynamically striped across all top-level virtual devices.
- Data placement is done at the time of “write.”
- When a new virtual device is added, data is gradually allocated to the new device.

Note: Although ZFS supports combining different types of virtual devices within the same pool, the recommended practice is to use top-level virtual devices of the same type with the same redundancy level in each device.

Virtual Devices and Dynamic Striping

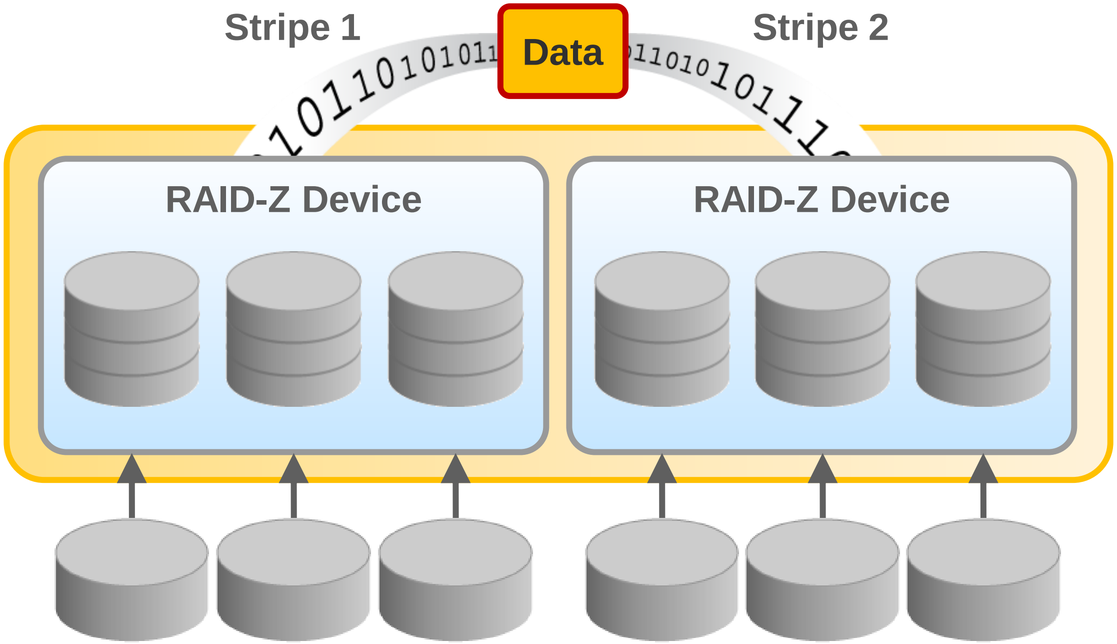
ZFS dynamically stripes data across all the top-level virtual devices.



Virtual Devices and Dynamic Striping : RAID-Z

Data is:

- Dynamically striped across all virtual devices in a RAID-Z pool
- Redundant within each virtual device in the RAID-Z pool



ZFS Storage Pool Types

Storage Pool Type	Description
Basic Storage Pool	A storage pool with a configuration that has a minimum of one disk
Mirrored Storage Pool	A storage pool with a configuration that has at least two disks, preferably on separate controllers. Many disks can be used in a mirrored configuration.
RAID-Z Storage Pool	A storage pool that consists of a configuration with single-, double-, or triple-parity fault tolerance, which means that one, two, or three device failures can be sustained respectively without any data loss.

ZFS File Systems

- Are default disk-based and root file systems in Oracle Solaris 11
- Share space with all the file systems in the pool
- Grow automatically within the space allocated to the storage pool
- Immediately use additional space when new storage is added

Managing Data

As part of learning how to manage data, you will learn about implementing the following in the next section:

- ZFS storage pool functionality
- ZFS file system functionality
- ZFS snapshot and clone functionality

Agenda

- Introducing ZFS
- **Administering ZFS Storage Pools**
- Administering ZFS File Systems
- Administering ZFS Properties
- Administering ZFS Snapshots and Clones

Determining Your ZFS Storage Pool Requirements

As part of data management, you should identify your storage pool device requirements:

- Disks that are at least 128 MB in size
- Disks not in use by other parts of the operating system
- Entire disks that are formatted as a single, large slice or individual slices on a preformatted disk

Creating ZFS Storage Pools

- Use the `zpool create` command to create a basic storage pool.
- The `zpool create` command accepts:
  - A pool name
  - Any number of virtual devices

Creating a Basic Storage Pool

To create a basic ZFS pool, enter `zpool create`, followed by the pool name and disks to include in the pool.

```
# zpool create hrpool c1t0d0 c1t1d0
```

Both disks are:

- Found in `/dev/dsk`
- Labeled by ZFS to contain a single, large slice
- Dynamically striped across with data

Determining Local Storage Disk Availability

To display disk availability, run `format`.

```
# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
       0. c1t0d0 <Unknown-Unknown-0001-16.00GB>
          /xpvd/xdf@51712
       1. c1t2d0 <Unknown-Unknown-0001 cyl 1024 alt 0 hd 128 sec 32>
          /xpvd/xdf@51744
       2. c1t3d0 <Unknown-Unknown-0001 cyl 1024 alt 0 hd 128 sec 32>
          /xpvd/xdf@51760
       3. c1t4d0 <Unknown-Unknown-0001 cyl 1024 alt 0 hd 128 sec 32>
          /xpvd/xdf@51776
       4. c1t5d0 <Unknown-Unknown-0001 cyl 1024 alt 0 hd 128 sec 32>
          /xpvd/xdf@51792
       5. c1t6d0 <Unknown-Unknown-0001 cyl 1024 alt 0 hd 128 sec 32>
          /xpvd/xdf@51808
       6. c1t7d0 <Unknown-Unknown-0001 cyl 1024 alt 0 hd 128 sec 32>
          /xpvd/xdf@51824
<output truncated>
```

Creating a Mirrored Storage Pool

To create a mirrored storage pool, enter `zpool create`, followed by the pool name, the `mirror` keyword, and the storage devices that will comprise the mirror.

```
# zpool create hrpool mirror c1t3d0 c1t4d0 mirror c1t5d0 c1t6d0
```

Data is:

- Dynamically striped across both mirrors
- Redundant between each disk within a mirror

Creating a RAID-Z Storage Pool

To create a ZFS RAID-Z storage pool, enter `zpool create`, followed by the pool name, the `raidz` keyword, and the storage devices that will be part of each RAID-Z pool.

```
# zpool create hrpool raidz c1t2d0 c1t3d0 c1t4d0 c1t5d0
/dev/dsk/c1t6d0
```

```
# zpool create datapool raidz2 c1t2d0 c1t3d0 c1t4d0 c1t5d0
c1t6d0 c1t7d0 raidz2 c1t8d0 c1t9d0 c1t10d0 c1t11d0 c1t12d0
c1t13d0
```

Default Mount Point for Storage Pools

- The default mount point is `/`*`pool-name`*.
- A directory is automatically created if it does not exist.
- If a directory exists, it must be empty.

```
# zpool create home c1t2d0
default mountpoint '/home' exists and is not empty
use '-m' option to provide a different default
```

- To change the default mount point, use `-m` with `zpool create`.

```
# zpool create -m /export/zfs home c1t2d0
```

Destroying a ZFS Storage Pool

To destroy a pool, enter `zpool destroy`, followed by the pool name.

```
# zpool destroy testpool
```

Caution: Be very careful when you destroy a pool. Make sure that you are destroying the right pool and that you always have copies of your data. If you accidentally destroy the wrong pool, you can attempt to recover the pool.

ZFS Storage Pool Properties

Pool properties:

- Determine the behavior of a pool feature, such as whether:
  - A pool is bootable
  - A property is enabled
- Identify read-only attributes, such as:
  - Current pool size
  - Unique pool identifier (GUID)

Displaying Pool Properties

Use `zpool get all`, followed by the pool name to display all property information for a pool.

```
# zpool get all hrpool
NAME    PROPERTY        VALUE                 SOURCE
hrpool  allocated       85K                   -
hrpool  altroot         -                     default
hrpool  autoexpand      off                   default
hrpool  autoreplace     off                   default
hrpool  bootfs          -                     default
hrpool  cachefile       -                     default
hrpool  capacity        0%                    -
hrpool  dedupditto      0                     default
hrpool  dedupratio      1.00x                 -
hrpool  delegation      on                    default
hrpool  failmode        wait                  default
hrpool  free            15.9G                 -
hrpool  guid            13211416720083688767  -
hrpool  health          ONLINE                -
hrpool  listshares      off                   default
hrpool  listsnapshots   off                   default
hrpool  readonly        off                   -
hrpool  size            15.9G                 -
hrpool  version         37                    default
```

Displaying Pool Properties

```
# zpool get all hrpool
NAME    PROPERTY       VALUE                 SOURCE
hrpool  allocated      85K                   -
hrpool  altroot        -                     default
hrpool  autoexpand     off                   default
hrpool  autoreplace    off                   default
hrpool  bootfs         -                     default
hrpool  cachefile      -                     default
hrpool  capacity       0%                    -
hrpool  dedupditto     0                     default
hrpool  dedupratio     1.00x                 -
hrpool  delegation     on                    default
hrpool  failmode       wait                  default
hrpool  free           15.9G                 -
hrpool  guid           13211416720083688767  -
hrpool  health         ONLINE                -
hrpool  listshares     off                   default
hrpool  listsnapshots  off                   default
hrpool  readonly       off                   -
hrpool  size           15.9G                 -
hrpool  version        37                    default
```

Displaying Pool Properties

```
# zpool get all hrpool
NAME    PROPERTY       VALUE                 SOURCE
hrpool  allocated      85K                   -
hrpool  altroot        -                     default
hrpool  autoexpand     off                   default
hrpool  autoreplace    off                   default
hrpool  bootfs         -                     default
hrpool  cachefile      -                     default
hrpool  capacity       0%                    -
hrpool  dedupditto     0                     default
hrpool  dedupratio     1.00x                 -
hrpool  delegation     on                    default
hrpool  failmode       wait                  default
hrpool  free           15.9G                 -
hrpool  guid           13211416720083688767  -
hrpool  health         ONLINE                -
hrpool  listshares     off                   default
hrpool  listsnapshots  off                   default
hrpool  readonly       off                   -
hrpool  size           15.9G                 -
hrpool  version        37                    default
```

Querying ZFS Pool Status

You can request the following types of information about a pool:

- Basic usage information
- I/O statistics
- Health status
- Command history

Displaying Basic Pool Usage Information

To display usage information about all the pools in the system, use `zpool list`.

```
# zpool list
NAME        SIZE    ALLOC     FREE     CAP   DEDUP   HEALTH   ALTROOT
hrpool     80.0G   22.3G    47.7G     28%   1.00x   ONLINE    -
datapool    1.2T   384G      816G     32%   1.00x   ONLINE    -
```

Note: To gather statistics for a specific pool, specify the pool name, as in the following example: `zpool list hrpool`.

Displaying Specific Pool Statistics

To display specific statistics, use `zpool list` with the `-o` option.

```
# zpool list -o name,size
NAME       SIZE
hrpool     80.0G
datapool   1.2T
```

Displaying Specific Pool Statistics

Use the `–H` option to suppress column headings and to separate fields by tabs rather than by spaces.

```
# zpool list -H -o name,size
hrpool    80.0G
datapool  1.2T
```

Viewing Pool I/O Statistics

Use `zpool iostat` with no options to display accumulated statistics since boot for all the pools in the system.

```
# zpool iostat
                capacity       operations        bandwidth
pool          alloc  free     read   write     read   write
----------    -----  -----    -----  -----     -----  -----
hrpool        100G   20.0G    1.2M   102K      1.2M   3.45K
datapool      12.3G  67.7G    132K   15.2K     32.1K  1.20K
----------    -----  -----    -----  -----     -----  -----
```

Note: You can request a more accurate view of current bandwidth usage by specifying an interval.

Viewing Pool I/O Statistics

Use `zpool iostat -v` to request the complete virtual device layout, as well as all I/O statistics.

```
# zpool iostat -v hrpool
               capacity        operations        bandwidth
pool         alloc  free      read   write     read  write
----------   -----  -----     -----  -----     -----  -----
hrpool       20.4G  59.6G         0     22         0   6.00
mirror       20.4G  59.6G         0     22         0   6.00
  c1t3d0         -      -         1    295     11.2K    148
  c1t4d0         -      -         1    299     11.2K    148
----------   -----  -----     -----  -----     -----  -----
```

Viewing Pool I/O Statistics

When viewing I/O statistics on a virtual device, remember the following:

- Space usage is available for top-level virtual devices only.
- The numbers might not always add up as you expect.
  - This is most noticeable immediately after pool creation.
  - It should gradually equalize.
  - Broken, unresponsive, or offlined devices can affect symmetry as well.

Note: You can also specify intervals when examining virtual device statistics.

Determining the Health Status of a Pool

- The health of a pool is:
  - Determined from the state of all its devices
  - Displayed by using the `zpool status` command
- Potential pool and device failures are:
  - Reported by `fmd`
  - Displayed on the system console
  - Logged in the `/var/adm/messages` file

Determining the Health Status of a Pool

A pool's health status is described by one of the following four states:

State	Description
DEGRADED	A pool with one or more failed devices, whose data is still available due to a redundant configuration
ONLINE	A pool that has all devices operating normally
SUSPENDED	A pool that is waiting for device connectivity to be restored. A SUSPENDED pool remains in the wait state until the device issue is resolved.
UNAVAIL	A pool with corrupted metadata, or one or more unavailable devices, and insufficient replicas to continue functioning

Determining the Health Status of a Pool

Each device can fall into one of the following states:

State	Description
`ONLINE`	The device or virtual device is in normal working order.
`DEGRADED`	The virtual device has experienced failure but is still able to function.
`OFFLINE`	The device has been explicitly taken offline by the administrator.
`REMOVED`	The device was physically removed while the system was running.
`UNAVAIL`	The device or virtual device cannot be opened (that is, it is unavailable).

Determining the Health Status of a Pool

The health of a pool is determined from the health of all its top-level virtual devices:

State of Top-Level vdevs	State of Pool	Result
All vdevs `ONLINE`	`ONLINE`	No issues are present.
One or more vdevs `DEGRADED` or `UNAVAIL`	`DEGRADED`	The pool continues to run but the level of redundancy or data throughput might be affected.
One or more vdevs `UNAVAIL` or `OFFLINE`	`UNAVAIL` or `SUSPENDED`	The pool is completely inaccessible. No data recovery is possible until devices are attached or repaired.

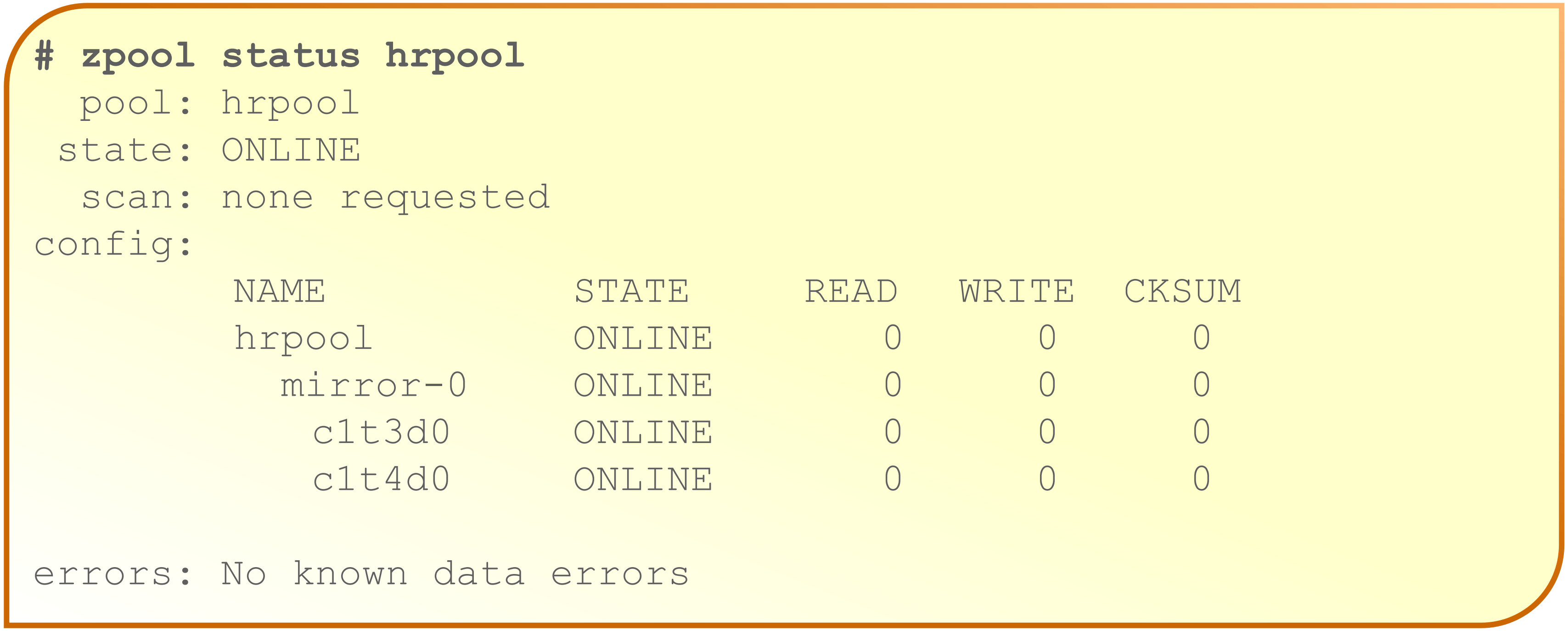
Determining the Health Status of a Pool

Use `zpool status -x` to request a quick overview of pool health status.

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

Determining the Health Status of a Pool

To examine the health of a specific pool, use `zpool status`, followed by the pool name.



```
# zpool status hrpool
  pool: hrpool
 state: ONLINE
  scan: none requested
config:
        NAME           STATE     READ  WRITE  CKSUM
        hrpool         ONLINE       0      0      0
          mirror-0     ONLINE       0      0      0
            c1t3d0     ONLINE       0      0      0
            c1t4d0     ONLINE       0      0      0

errors: No known data errors
```

Determining the Health Status of a Pool

Use `zpool status -v`, followed by the pool name to request a more detailed summary of a pool's health status.

```
# zpool status -v hrpool
  pool: hrpool1
 state: DEGRADED
status: One or more devices are unavailable in response to persistent errors.
Sufficient replicas exist for the pool to continue functioning in a degraded state.
action: Determine if the device needs to be replaced, and clear the errors using
'zpool clear' or 'fmadm repaired', or replace the device with 'zpool replace'.
Run 'zpool status -v' to see device specific details.
  scan: none requested
config:
NAME        STATE     READ WRITE CKSUM
hrpool1     DEGRADED     0     0     0
  mirror-0  DEGRADED     0     0     0
    c1t3d0  ONLINE       0     0     0
    c1t4d0  UNAVAIL      0     0     0
errors: No known data errors
```

Displaying Pool Command History

Use `zpool history` to display `zpool` commands that modify pool state information.

```
# zpool history hrpool
History for ’hrpool’:
2015-10-24.11:20:57 zpool create hrpool c1t3d0 c1t4d0
```

Quiz

What command is used to create a ZFS storage pool?

a. `zpool start new pool`

b. `zpool storagepool`

c. `zpool create`

d. `zpool make`

Quiz

What command is used to create a ZFS storage pool?

a. `zpool start new pool`

b. `zpool storagepool`

c. `zpool create`

d. `zpool make`

Quiz

After you have created a pool, you must manually create the mount point for the pool.

a. True

b. False

Quiz

After you have created a pool, you must manually create the mount point for the pool.

a. True

b. False

Quiz

Which command is used to display all the property settings within a pool?

a. zpool show all <*poolname*>

b. zpool get all <*poolname*>

c. zpool display all <*poolname*>

d. zpool set all <*poolname*>

Quiz

Which command is used to display all the property settings within a pool?

a. `zpool show all <`*`poolname`*`>`

b. `zpool get all <`*`poolname`*`>`

c. `zpool display all <`*`poolname`*`>`

d. `zpool set all <`*`poolname`*`>`

Quiz

Which command is used to display basic pool usage information?

a. `zpool list`

b. `zpool iostat`

c. `zpool history`

d. `zpool status`

Quiz

Which command is used to display basic pool usage information?

a. `zpool list`

b. `zpool iostat`

c. `zpool history`

d. `zpool status`

Quiz

If a pool is in `DEGRADED` state, the data is completely inaccessible.

a. True

b. False

Quiz

If a pool is in `DEGRADED` state, the data is completely inaccessible.

a. True

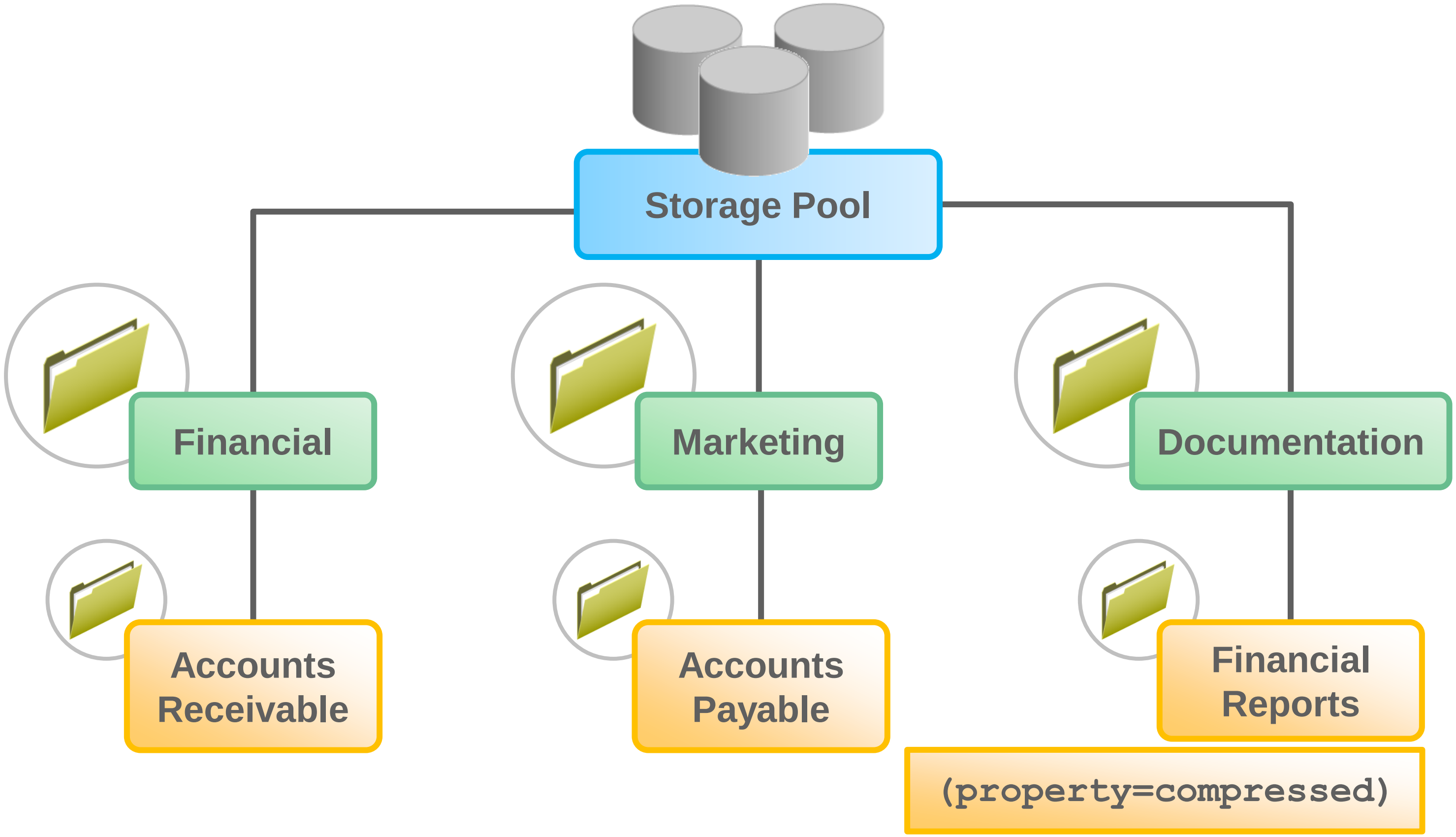
b. False

Agenda

- Introducing ZFS
- Administering ZFS Storage Pools
- **Administering ZFS File Systems**
- Administering ZFS Properties
- Administering ZFS Snapshots and Clones

Determining ZFS File System Configuration Requirements

As part of data management, the file system configuration requirements are determined.



Creating a ZFS File System

To create a file system, enter `zfs create`, followed by the file system path name.

```
# zfs create hrpool/home/reports
```

- The file system name is specified as a path name:

  `pool-name/[filesystem-name/]filesystem-name`
- The pool name and initial file system names identify the location in the hierarchy where a new file system will be created.
- The last name identifies the file system to be created.

Note: You can create missing intermediate file system names automatically by using the `zfs create -p` command.

Creating a ZFS File System

A file system is mounted:

- Automatically if it is created successfully
- As `/dataset`
- By using the path provided in the `create` subcommand

Destroying a ZFS File System

To destroy a file system, enter `zfs destroy`, followed by the file system path name.

```
# zfs destroy hrpool/home/oldreports
```

Caution: No confirmation prompt appears with the `destroy` subcommand.

Destroying a ZFS File System

If the `zfs destroy` command fails, use one of the options shown in the following table:

Condition	Option	Results
File system is busy.	`-f`	Can unmount, unshare, and destroy active file systems, causing unexpected application behavior
File system has children.	`-r`	Recursively destroys a file system and all its descendants. This option also destroys snapshots.
File system has indirect dependents.	`-R`	Recursively destroys all dependents, including cloned file systems outside the target hierarchy

Caution: No confirmation prompts appear with the `-f`, `-r`, and `-R` options.

Renaming a ZFS File System

To rename a file system, enter `zfs rename`, followed by the file system path name.

```
# zfs rename hrpool/home/reviews hrpool/home/reviews_2015
```

You can use the `rename` subcommand to:

- Change the name of a file system
- Relocate the file system to a new location within the ZFS hierarchy

Renaming a ZFS File System – change path

Example of file system relocation:

```
# zfs rename hrpool/home/jobdesc hrpool/ws/jobdesc
```

- The new location:
  - Must be within the same pool
  - Must have enough space to hold the new file system

Renaming a ZFS File System

- The renaming operation attempts an unmount or remount sequence for:
  - The file system
  - Any descendant file systems
- If it is unable to unmount an active file system:
  - The rename operation fails
  - A forced unmount is required

Listing Basic ZFS Information

To display basic dataset information, use `zfs list` with no options.

```
# zfs list
NAME                      USED   AVAIL  REFER      MOUNTPOINT
pool                      476K   16.5G    21K      /pool
pool/clone                 18K   16.5G    18K      /pool/clone
pool/home                 296K   16.5G    19K      /pool/home
pool/home/data            277K   16.5G   277K      /pool/home/data
pool/test                  18K   16.5G    18K      /test
...
<output truncated>
```

Listing Basic ZFS Information – with -r

You can also use `zfs list` to display the following:

- Specific datasets by using the dataset name
- Dataset descendants recursively with `-r`

```
# zfs list -r pool/home/data
NAME                        USED   AVAIL  REFER      MOUNTPOINT
pool/home/data              277K   16.5G   277K      /pool/home/data
```

Mounting ZFS File Systems

Use the `zfs mount` command to:

- View ZFS-managed mounted file systems
- Change mount options
- Explicitly mount a file system

Mounting ZFS File Systems

To view all file systems currently mounted and managed by ZFS, use `zfs mount` with no arguments.

```
# zfs mount
...
hrpool                             /hrpool
hrpool/home                        /hrpool/home
hrpool/home/reports                /hrpool/home/reports
...
```

To mount all ZFS-managed file systems, use `zfs mount -a`.

```
# zfs mount -a
```

Note: A mounted file system uses a set of mount options based on the property values associated with the dataset.

Unmounting a ZFS File System

To unmount a ZFS file system, use `zfs unmount`, followed by either the file system name or the mount point.

```
# zfs unmount hrpool/home/qarpts
```

```
# zfs unmount /export/home/qarpts
```

Note: If the file system is active or busy, `zfs unmount` fails. You can use `–f` to force the unmount, but you should use this option with caution.

Quiz

Which command is used to create a ZFS file system?

a. `zfs make`

b. `zfs create`

c. `zpool create`

d. `zpool make`

Quiz

Which command is used to create a ZFS file system?

a. `zfs make`

b. `zfs create`

c. `zpool create`

d. `zpool make`

Quiz

Which option, when used with the `zfs destroy` command, can destroy an active ZFS file system?

a. `-a`

b. `-f`

c. `-r`

d. `-R`

Quiz

Which option, when used with the `zfs destroy` command, can destroy an active ZFS file system?

a. `-a`

b. `-f`

c. `-r`

d. `-R`

Quiz

When you relocate a file system through `rename`, the new location must be within the same pool.

a. True

b. False

Quiz

When you relocate a file system through `rename`, the new location must be within the same pool.

a. True

b. False

Agenda

- Introducing ZFS
- Administering ZFS Storage Pools
- Administering ZFS File Systems
- **Administering ZFS Properties**
- Administering ZFS Snapshots and Clones

Administering ZFS Properties

- ZFS Properties: Overview
- Types of Native ZFS Properties
- Querying ZFS Properties
- Setting ZFS Properties
- Inheriting ZFS Properties

ZFS Properties: Overview

- Properties allow you to control the following:
  - File systems
  - Volumes
  - Snapshots
  - Clones
- There are two property types:
  - Native
    - Export internal statistics
    - Control ZFS file system behavior
  - User-defined
    - Have no effect on ZFS file system behavior
    - Can be used to annotate datasets

Types of Native ZFS Properties

- Read-only statistics
  - Can be retrieved but not set
  - Are not inherited
- Settable
  - Can be both retrieved and set
  - Are inheritable (exceptions: quotas and reservations)

Note: An inheritable property is a property that, when set on a parent, is propagated to all of its descendants.

Identifying Native ZFS Properties

Property Name	Type	Default Value	Description
`compression`	String	`off`	Enables or disables compression for a dataset
`mountpoint`	String	N/A	Controls the mount point used for this file system
`quota`	Number (or none)	`none`	Limits the amount of disk space that a dataset and its descendants can consume
`readonly`	Boolean	`off`	Controls whether a dataset can be modified. When it is set to `on`, no modifications can be made.
`sharenfs`	String	`off`	Controls whether a ZFS dataset is published as an NFS share

Querying ZFS Properties

You can query property values with:

- `zfs list`
- `zfs get`
  - Complex queries
  - Scripting
  - Any dataset property

Retrieving ZFS Properties

To retrieve any dataset property, use `zfs get`, followed by the property name and the dataset name.

```
# zfs get checksum hrpool/ws
NAME                PROPERTY        VALUE        SOURCE
hrpool/ws           checksum        on           default
```

Retrieving ZFS Properties

The source values in `zfs get` are shown in the following table:

Source Value	Definition
`Default`	The property setting was not inherited or set locally.
`Local`	The property was explicitly set on the dataset by using the `zfs set` command.
`inherited` **from** ***dataset-name***	The property was inherited from the named ancestor.
`Temporary`	This property value was set by using the `zfs mount -o` option, and is valid only for the lifetime of the mount.
`- (none)`	This property is a read-only property. Its value is generated by ZFS.

Retrieving ZFS Properties

To retrieve all properties for a specified dataset, use `zfs get all`, followed by the dataset name.

```
# zfs get all hrpool
NAME     PROPERTY        VALUE      SOURCE
hrpool   aclinherit      restricted default
hrpool   aclmode         discard    default
hrpool   atime           on         default
hrpool   available       15.6G      -
hrpool   canmount        on         default
...
<output truncated>
```

Retrieving ZFS Properties

To specify the property types to display, use `zfs get -s`, followed by the source value and the dataset name.

```
# zfs get -s local all hrpool
NAME      PROPERTY          VALUE     SOURCE
hrpool    compression       on        local
```

With the `-s` option, you can:

- Specify the desired source types with a comma-separated list
- Use the following source types: `default`, `local`, `inherited`, `temporary`, and `none`

Retrieving ZFS Properties

The following `zfs get` options are designed for scripting:

- `-H`
  - Omits header information
  - Presents all white space as tabs
- `-o`
  - Allows customization of output
  - Takes a comma-separated list of literal fields to display, together with a separate list of properties

```
# zfs get -H -o value compression hrpool/home
on
```

Setting ZFS Properties

To modify any settable dataset property, use `zfs set`, followed by *`property=value`* and the dataset name.

```
# zfs set atime=off hrpool/home
```

Note: Only one property can be set or modified during each `zfs set` invocation.

You can also set a property during the creation of a dataset by using `zfs create`.

```
# zfs create -o atime=off hrpool/home
```

Inheriting ZFS Properties

- All settable properties inherit their values from their parents.
- All inheritable properties have an associated source.

Source Value	**Definition**
`default`	The property setting was not inherited or set locally.
`local`	The property was explicitly set on the dataset by using the `zfs set` command.
`inherited` from *dataset-name*	The property was inherited from the named ancestor.

Inheriting ZFS Properties – get

```
# zfs list
NAME                           USED  AVAIL  REFER  MOUNTPOINT
datapool                       176K  1.95G    23K  /export/share
datapool/software               65K  1.95G    23K  /export/share/software
datapool/software/solaris       42K  1.95G    21K  /export/share/software/solaris
datapool/software/solaris/ar   21K  1.95G    21K  /export/share/software/solaris/ar
```

```
# zfs get -r compression datapool
NAME                            PROPERTY     VALUE      SOURCE
datapool                        compression  off        default
datapool/software               compression  off        default
datapool/software/solaris       compression  off        default
datapool/software/solaris/ar    compression  off        default
```

Inheriting ZFS Properties - set

```
# zfs set compression=on datapool/software/solaris
# zfs get -r compression datapool
NAME                           PROPERTY     VALUE   SOURCE
datapool                       compression  off     default
datapool/software              compression  off     default
datapool/software/solaris      compression  on      local
datapool/software/solaris/ar   compression  on      inherited from
    datapool/software/solaris
```

Inheriting ZFS Properties

To clear a property setting and have the setting inherited from the parent, use `zfs inherit`, followed by the property name and the system file name path.

```
# zfs inherit compression datapool/software/solaris
# zfs get -r compression datapool
NAME                         PROPERTY     VALUE      SOURCE
datapool                     compression  off        default
datapool/software            compression  off        default
datapool/software/solaris    compression  off        default
datapool/software/solaris/ar compression  off        default
```

Agenda

- Introducing ZFS
- Administering ZFS Storage Pools
- Administering ZFS File Systems
- Administering ZFS Properties
- **Administering ZFS Snapshots and Clones**

Administering ZFS Snapshots and Clones

- Administering ZFS Snapshots
  - Creating a ZFS Snapshot
  - Displaying a ZFS Snapshot
  - Renaming a ZFS Snapshot
  - Holding a ZFS Snapshot
  - Rolling Back a ZFS Snapshot
  - Destroying a ZFS Snapshot
  - Snapshot Space Accounting
  - Identifying ZFS Snapshot Differences
- Administering ZFS Clones
  - Creating a ZFS Clone
  - Relationship of Clone and Snapshot
  - Replacing a ZFS File System with a ZFS Clone
  - Destroying a ZFS Clone

ZFS Snapshots

- Are a read-only view of a file system or volume
- Can be created quickly and easily
  - Unlimited number of snapshots are allowed.
- Consume no additional space initially
- Consume space as data within the active dataset changes
- Prevent data from being freed back to the pool
  - When snapshots are destroyed, the consumed space is released.

Creating a ZFS Snapshot

To create a snapshot, enter `zfs snapshot`, followed by the snapshot name.

The snapshot name is specified as follows:

- *`filesystem@snapname`*
- *`volume@snapname`*

```
# zfs snapshot hrpool/home/reports@friday
```

Creating a ZFS Snapshot

To create snapshots for all descendant file systems, use `zfs snapshot -r` and the snapshot name.

```
# zfs snapshot -r hrpool/home@now
# zfs list -t snapshot
NAME                         USED   AVAIL   REFER  MOUNTPOINT
hrpool/home@now                0       -    29.5K  -
hrpool/home/reports@now        0       -    2.15M  -
hrpool/home/reviews@now        0       -    1.89M  -
hrpool/home/jobdesc@now        0       -    1.89M  -
hrpool/home/bonus@now          0       -    2.15M  -
```

Note: Snapshots have no modifiable properties and dataset properties cannot be applied to a snapshot.

Displaying a ZFS Snapshot

To display snapshots, enter `zfs list -t snapshot`.

```
# zfs list -t snapshot
NAME                             USED  AVAIL   REFER   MOUNTPOINT
hrpool/home/reports@tuesday       18K      -     21K   -
hrpool/home/reports@wednesday     19K      -    280K   -
hrpool/home/reports@thursday        0      -    538K   -
```

The `listsnapshots` pool property:

- Is used to enable or disable the display of snapshots
- Is disabled by default
- Is enabled by using `zpool set listsnapshot=on` *`<poolname>`*

Displaying a ZFS Snapshot – for specific file system

To list the snapshots created for a specific file system, enter `zfs list -r -t snapshot`, followed by the file system name.

```
# zfs list -r -t snapshot -o name,creation hrpool/home
NAME                            CREATION
hrpool/home/reports@tuesday      Thur Nov 28 10:08 2015
hrpool/home/reports@wednesday    Fri  Nov 30 08:05 2015
hrpool/home/reports@thursday     Mon  Dec  2 07:03 2015
hrpool/home/bonus@now            Tue  Dec  3 06:15 2015
```

Renaming a ZFS Snapshot

To rename a snapshot, use `zfs rename`, followed by the snapshot name.

```
# zfs rename hrpool/home/report@121014 hrpool/home/report@today
```

Note: Snapshots must be renamed within the same pool and dataset from which they were created.

Renaming a ZFS Snapshot

To recursively rename snapshots, use `zfs rename -r`, followed by the snapshot name.

```
# zfs list
NAME                          USED    AVAIL REFER    MOUNTPOINT
users                         270K    16.5G   22K    /users
users/home                    76K     16.5G   22K    /users/home
users/home@yesterday          0       -      22K     -
users/home/jjones             18K     16.5G  18K     /users/home/jjones
users/home/jjones@yesterday   0       -      18K     -
# zfs rename -r users/home@yesterday @2daysago
# zfs list -r users/home
NAME                          USED    AVAIL REFER    MOUNTPOINT
users/home                    76K     16.5G  22K     /users/home
users/home@2daysago           0       -      22K     -
users/home/jjones             18K     16.5G  18K     /users/home/jjones
users/home/jjones@2daysago    0       -      18K     -
```

Holding a ZFS Snapshot

- The snapshot hold feature:
  - Prevents a snapshot from being destroyed by using `zfs destroy`
  - Allows a snapshot with clones to be deleted pending the removal of the last clone by using `zfs destroy -d`
- The snapshot user-reference count:
  - Is initialized to zero
  - Increases by one whenever a hold is put on the snapshot
  - Decreases by one whenever a hold is released
  - Must be at zero before the snapshot can be destroyed

Holding a ZFS Snapshot – using tag

To hold a snapshot or a set of snapshots, use `zfs hold tag_keep`, followed by the snapshot name.

```
# zfs hold tag_keep hrpool/home/report@snap1
```

To recursively hold the snapshots of all descendant file systems, use `zfs hold` with `-r`, followed by `tag_keep` and the snapshot name.

```
# zfs hold -r tag_keep hrpool/home@now
```

Note: Each snapshot has its own tag namespace, and tags must be unique within that space. `tag_keep` is only a tag.

Display Held snapshots

To display a list of held snapshots, use `zfs holds`, followed by the snapshot name.

```
# zfs holds hrpool/home@now
NAME             TAG    TIMESTAMP
hrpool/home@now  keep   Mon Mar 10 12:40:12 2015
```

To display a recursive list of held snapshots, use `zfs holds` with `-r`, followed by the snapshot name.

```
# zfs holds -r hrpool/home@now
NAME                     TAG    TIMESTAMP
hrpool/home/report@now   keep   Mon Mar 10 12:40:12 2015
hrpool/home/jjones@now   keep   Mon Mar 10 12:40:12 2015
hrpool/home@now          keep   Mon Mar 10 12:40:12 2015
```

Release a Held ZFS Snapshot

Two options to destroy a held ZFS snapshot:

1. Use `zfs destroy -d`, followed by the snapshot name, and then release the snapshot hold, which removes the snapshot.
2. Release the held snapshot, and then destroy it by using `zfs destroy`.

To release a held snapshot or a set of snapshots, use `zfs release -r`, followed by `keep` and the snapshot name.

```
# zfs release -r keep hrpool/home@now
```

Note: `-r` enables a recursive release of the hold.

Properties: defer_destroy and Userrefs

Snapshot hold properties:

- **`defer_destroy`**: Set to `on` if the snapshot is marked for deferred destruction by using the `zfs destroy -d` command
- **`Userrefs`**: Set to the number of holds on the snapshot

Display properties: defer_destroy and Userrefs

To view the ZFS snapshot hold properties, use `zfs get -r defer_destroy,userrefs`, followed by the file system name.

```
# zfs get -r defer_destroy,userrefs hrpool/home
NAME                      PROPERTY       VALUE     SOURCE
hrpool                    defer_destroy  -         -
hrpool                    userrefs       -         -
hrpool/home               defer_destroy  -         -
hrpool/home               userrefs       -         -
hrpool/home/report@now    defer_destroy  off       -
hrpool/home/report@now    userrefs       1         -
hrpool/home/jjones@now    defer_destroy  off       -
hrpool/home/jjones@now    userrefs       1         -
hrpool/home@now           defer_destroy  off       -
hrpool/home@now           userrefs       1         -
```

Rolling Back a ZFS Snapshot

To discard all the changes made since a specific snapshot, enter `zfs rollback`, followed by the snapshot name.

```
# zfs rollback hrpool/home/qarpt@thursday
```

By default, `zfs rollback` rolls back only to the most recent snapshot.

To destroy more recent snapshots, enter `zfs rollback` with `-r`, followed by the snapshot name.

```
# zfs rollback -r hrpool/home/qarpt@tuesday
```

Destroying a ZFS Snapshot

To destroy a snapshot, use `zfs destroy`, followed by the snapshot name.

```
# zfs destroy hrpool/home/reports@now
```

Things to know when attempting to destroy a snapshot:

- The dataset cannot be destroyed if snapshots of it exist.
- Clones created from a snapshot must be destroyed before the snapshot can be destroyed.

Snapshot Space Accounting

- When a snapshot is created, its space:
  - Is initially shared between the snapshot and the file system
  - Is possibly shared with previous snapshots
- As the file system changes, the previously shared space:
  - Becomes unique to the snapshot
  - Is counted in the snapshot's `used` property
- Deleting snapshots can increase the amount of space that is unique to (and thus used by other) snapshots.

Note: A snapshot's space `referenced` property is the same as that of the file system when the snapshot was created.

Identifying ZFS Snapshot Differences

To determine ZFS snapshot differences, use `zfs diff`, followed by the snapshot names.

```
# zfs snapshot datapool/hrdata@before
# touch /datapool/hrdata/newfile
# zfs snapshot datapool/hrdata@after
# zfs list -r -t snapshot -o name,creation
NAME                              CREATION
datapool/hrdata@before            Thu Oct 24 14:54 2015
datapool/hrdata@after             Thu Oct 24 14:59 2015
rpool/ROOT/solaris@install        Tue Oct 24 22:33 2015
# zfs diff datapool/hrdata@before datapool/hrdata@after
M/datapool/hrdata/
+/datapool/hrdata/newfile
#
```

Identifying ZFS Snapshot Differences

File or Directory Change	Identifier
A file or directory is modified, or a file or directory link has changed.	M
A file or directory is present in the older snapshot but not in the newer snapshot.	–
A file or directory is present in the newer snapshot but not in the older snapshot.	+
A file or directory is renamed.	R

ZFS Clones

- Writable volume or file system
- Created from a snapshot
- Nearly instantaneous creation
- Initially consumes no additional disk space

Creating a ZFS Clone

To create a clone, enter `zfs clone`, followed by the snapshot name from which the clone is to be created and the name of the new file system or volume.

```
# zfs snapshot hrpool/ws/gate@yesterday
# zfs clone hrpool/ws/gate@yesterday hrpool/home/reports/bug123
```

The new file system or volume:

- Can be located anywhere in the ZFS hierarchy
- Has the same dataset type (for example, file system or volume) as the snapshot from which the clone was created

Note: A clone of a file system must be created in the same pool where the original file system snapshot resides.

Relationship of Clone and Snapshot

- A clone can be created only from a snapshot.
- An implicit dependency exists between the clone and the snapshot.
- The original snapshot cannot be destroyed as long as the clone exists.
  - The `origin` property exposes this dependency.
  - The `zfs destroy` command lists any such dependencies (if they exist).
- A clone does not inherit the properties of the dataset from which it was created.

Note: Use the `zfs get` and `zfs set` commands to view and change the properties of a cloned dataset.

Replacing a ZFS File System with a ZFS Clone

With the clone replacement process, you can:

- Clone and replace file systems so that the original file system becomes the clone of the newly created file system
- Destroy the file system from which the clone was originally created

Note: Without clone promotion, you cannot destroy the original file system of active clones.

Replacing a ZFS File System with a ZFS Clone

To replace an active ZFS file system with a clone of that file system, use `zfs promote`, followed by the clone name.

```
# zfs snapshot hrpool/reviews/q4@today
# zfs clone hrpool/reviews/q4@today hrpool/reviews/q4sum
# zfs list -r hrpool/reviews
NAME                       USED    AVAIL   REFER   MOUNTPOINT
hrpool/reviews             314K    8.24G   25.5K   /hrpool/reviews
hrpool/reviews/q4           288K   8.24G   288K    /hrpool/reviews/q4
hrpool/reviews/q4@today       0    -       288K    -
hrpool/reviews/q4sum          0    8.24G   288K    /hrpool/reviews/q4sum
```

```
# zfs promote hrpool/reviews/q4sum
# zfs list -r hrpool/reviews
NAME                       USED    AVAIL   REFER   MOUNTPOINT
hrpool/reviews             316K    8.24G   27.5K   /hrpool/reviews
hrpool/reviews/q4             0    8.24G   288K    /hrpool/reviews/q4
hrpool/reviews/q4sum        288K   8.24G   288K    /hrpool/reviews/q4sum
hrpool/reviews/q4@today       0        -   288K    -
```

Replacing a ZFS File System with a ZFS Clone

To rename the promoted file system to the original name, use `zfs rename`, followed by the current file system name and a new file system name.

```
# zfs rename hrpool/reviews/q4 hrpool/reviews/q4legacy
# zfs rename hrpool/reviews/q4sum hrpool/reviews/q4
# zfs list -r hrpool/reviews
NAME                            USED  AVAIL    REFER   MOUNTPOINT
hrpool/reviews                  316K  8.24G    27.5K   /hrpool/reviews
hrpool/reviews/q4               288K  8.24G    288K    /hrpool/reviews/q4
hrpool/reviews/q4@today            0      -    288K    -
hrpool/reviews/q4legacy            0  8.24G    288K    /hrpool/reviews/q4legacy
```

Destroying a ZFS Clone

To destroy a clone, use `zfs destroy`, followed by the clone name.

```
# zfs destroy hrpool/reviews/q4legacy
```

Quiz

You want to create a snapshot named `thursday` of the file system `/hrpool/home/smith`. Which of the following commands would you use to do this?

a. `zfs snapshot thursday hrpool/home/smith`

b. `zfs snapshot hrpool/home/smith thursday`

c. `zfs snapshot hrpool/home/smith@thursday`

d. `zfs snapshot hrpool/home/smith_thursday`

Quiz

You want to create a snapshot named `thursday` of the file system `/hrpool/home/smith`. Which of the following commands would you use to do this?

a. `zfs snapshot thursday hrpool/home/smith`

b. `zfs snapshot hrpool/home/smith thursday`

c. `zfs snapshot hrpool/home/smith@thursday`

d. `zfs snapshot hrpool/home/smith_thursday`

Summary

In this lesson, you should have learned how to:

- Describe ZFS and its features
- Administer ZFS storage pools
- Administer ZFS file systems
- Administer ZFS properties
- Administer ZFS snapshots and clones

Practice 6: Overview

- 6-1: Administering ZFS Storage Pools
- 6-2: Administering ZFS Pools by Using Disk Slices
- 6-3: Administering ZFS File Systems
- 6-4: Administering ZFS Snapshots and Clones