SO303 - Storage Administration

Preparing your Solaris 11 VM

Before continuing further, shutdown your Solaris 11 VM. Using the **Storage** section of the **Oracle VM VirtualBox Manager**, add the following **.vmdk** disks to the **existing** SATA controller of your Solaris 11 VM:

Disk	Size	Name
c7t2d0	200 Mb	Disk1.vmdk
c7t3d0	200 Mb	Disk2.vmdk
c7t4d0	200 Mb	Disk3.vmdk
c7t5d0	200 Mb	Disk4.vmdk
c7t6d0	200 Mb	Disk5.vmdk
c7t7d0	20 Gb	Mirror.vmdk

Using the **System** section of the **Oracle VM VirtualBox Manager**, add a second processor to your Solaris 11 VM.

Finally, boot your Solaris 11 VM.

Introduction

All previous versions of Solaris, including Solaris 10 use the **Unix File System** (UFS) as their default file system. Solaris 11 uses the **Zettabyte File System** (ZFS), introduced with Solaris 10, as its default file system.

Solaris 11 and ZFS

The Solaris 11 implementation of ZFS includes the following capabilities:

- 128-bit addressing,
- data integrity assurance,
- automated data corruption detection and repair,
- encryption,
- compression,
- de-duplication,
- quotas,
- file system migration between pools,
- snapshots.

ZFS Vocabulary

The introduction of ZFS was obviously accompanied by a new vocabulary:

Term	Description
pool	A storage element regrouping one or more disk partitions containing one or more file systems
file system	A dataset containing directories and files
clone	A copy of a file system
snapshot	A read-only copy of the state of a file system
compression	The reduction of storage space achieved by the removal of duplicate data blocks
de-duplication	The reduction of storage space achieved by the removal of redundant data patterns
checksum	A 256-bit number used to validate data when read or written
encryption	The protection of data using a password
quota	The maximum amount of disk space used by a group or user
reservation	A preallocated amount of disk space assigned to a user or file system
mirror	An exact duplicate of a disk or partition
RAID-Z	ZFS implementation of RAID-5
RAID-Z2	ZFS implementation of RAID-6
RAID-Z3	ZFS implementation of Triple Parity RAID

ZFS Commands

The ZFS commands are as follows:

Command	Description
zpool	Used to manage ZFS pools
zfs	Used to manage ZFS file systems

The zpool Command

The **zpool** command uses a set of subcommands:

Command	Description
create	Creates a storage pool and configures its mount point
destroy	Destroys a storage pool
list	Displays the health and storage usage of a pool
get	Displays a list of pool properties
set	Sets a property for a pool
status	Displays the health of a pool
history	Displays the commands issued for a pool since its creation
add	Adds a disk to an existing pool
remove	Removes a disk from an existing pool
replace	Replaces a disk in a pool by another disk
scrub	Verifies the checksums of a pool and repairs any damaged data blocks

The zfs Command

The **zfs** command use a set of subcommands:

Command	Description
create	Creates a ZFS file system, sets its properties and automatically mounts it

Command	Description
destroy	Destroys a ZFS file system or snapshot
list	Displays the properties and storage usage of a ZFS file system
get	Displays a list of ZFS file system properties
set	Sets a property for a ZFS file system
snapshot	Creates a read-only copy of the state of a ZFS file system
rollback	Returns the file system to the state of the **last** snapshot
send	Creates a file from a snapshot in order to migrate it to another pool
receive	Retrieves a file created by the subcommand **send**
clone	Creates a copy of a snapshot
promote	Transforms a clone into a ZFS file system
diff	Displays the file differences between two snapshots or a snapshot and its parent file system
mount	Mounts a ZFS file system at a specific mount point
unmount	Unmounts a ZFS file system

Solaris Slices

Those familiar with UFS on Solaris will remember having to manipulate Solaris **slices**. Those slices still exist:

```
root@solaris:~# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
       0. c7t0d0 <ATA-VBOX HARDDISK-1.0-20.00GB>
          /pci@0,0/pci8086,2829@d/disk@0,0
       1. c7t2d0 <ATA-VBOX HARDDISK-1.0-200.00MB>
          /pci@0,0/pci8086,2829@d/disk@2,0
       2. c7t3d0 <ATA-VBOX HARDDISK-1.0-200.00MB>
          /pci@0,0/pci8086,2829@d/disk@3,0
       3. c7t4d0 <ATA-VBOX HARDDISK-1.0-200.00MB>
          /pci@0,0/pci8086,2829@d/disk@4,0
```

```
       4. c7t5d0 <ATA-VBOX HARDDISK-1.0-200.00MB>
          /pci@0,0/pci8086,2829@d/disk@5,0
       5. c7t6d0 <ATA-VBOX HARDDISK-1.0-200.00MB>
          /pci@0,0/pci8086,2829@d/disk@6,0
       6. c7t7d0 <ATA-VBOX HARDDISK-1.0 cyl 2608 alt 2 hd 255 sec 63>
          /pci@0,0/pci8086,2829@d/disk@7,0
Specify disk (enter its number): 0
selecting c7t0d0
[disk formatted]
/dev/dsk/c7t0d0s1 is part of active ZFS pool rpool. Please see zpool(1M).


FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        fdisk      - run the fdisk program
        repair     - repair a defective sector
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        verify     - read and display labels
        inquiry    - show disk ID
        volname    - set 8-character volume name
        !<cmd>     - execute <cmd>, then return
        quit
format> part


PARTITION MENU:
        0      - change `0' partition
```

```
        1      - change `1' partition
        2      - change `2' partition
        3      - change `3' partition
        4      - change `4' partition
        5      - change `5' partition
        6      - change `6' partition
        select - select a predefined table
        modify - modify a predefined partition table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        !<cmd> - execute <cmd>, then return
        quit
partition> pri
Current partition table (original):
Total disk sectors available: 41926589 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector         Size         Last Sector
  0  BIOS_boot    wm               256     255.88MB          524287
  1        usr    wm            524288      19.74GB        41913087
  2 unassigned    wm                 0          0               0
  3 unassigned    wm                 0          0               0
  4 unassigned    wm                 0          0               0
  5 unassigned    wm                 0          0               0
  6 unassigned    wm                 0          0               0
  8   reserved    wm          41913088       8.00MB        41929471

partition>
```

Note the following line in the above output:

/dev/dsk/c7t0d0s1 is part of active ZFS pool rpool. Please see zpool(1M).

Since you are using ZFS for storage management, you no longer need to bother about slices !

iSCSI Storage

In Solaris 10 the configuration of iSCSI LUNs was accomplished using the **iscsitadm** command and the ZFS **shareiscsi** property. In Solaris 11 these have been replaced by by the use of **Common Multiprotocol SCSI Target** (COMSTAR). COMSTAR is a **framework** that turns a Solaris host into a SCSI target.

COMSTAR includes the following features:

- scalability,
- compatibility with generic host adapters,
- multipathing,
- LUN masking and mapping functions.

An iSCSI target is an **endpoint** waiting for connections from clients called **initiators**. A target can provide multiple **Logical Units** which provides classic read and write data operations.

Each logical unit is backed by a **storage device**. You can create a logical unit backed by any one of the following:

- a file,
- a thin-provisioned file,
- a disk partition,
- a ZFS volume.

LAB #1 - Managing ZFS Storage

Displaying Online Help

Both the **zpool**and **zfs** commands have built-in online help:

```
root@solaris:~# zpool help
The following commands are supported:
add      attach   clear    create   destroy  detach   export   get
help     history  import   iostat   list     offline  online   remove
replace  scrub    set      split    status   upgrade
For more info, run: zpool help <command>
root@solaris:~# zfs help
The following commands are supported:
allow       clone       create      destroy     diff        get
groupspace  help        hold        holds       inherit     key
list        mount       promote     receive     release     rename
rollback    send        set         share       snapshot    unallow
unmount     unshare     upgrade     userspace
For more info, run: zfs help <command>
```

Note that you can get help on subcommands by either using **zpool help <subcommand>** or **zfs help <subcommand>**.

Checking Pool Status

Use the **zpool** command with the **list** subcommand to display the details of your pool:

```
root@solaris:~# zpool list
NAME    SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
```

```
rpool  19.6G  6.96G  12.7G  35%  1.00x  ONLINE  -
```

Now use the **status** subcommand:

```
root@solaris:~# zpool status
  pool: rpool
 state: ONLINE
  scan: none requested
config:

    NAME        STATE     READ WRITE CKSUM
    rpool       ONLINE       0     0     0
      c7t0d0s1  ONLINE       0     0     0

errors: No known data errors
```

Creating a Mirrored Pool

Create a ZFS mirrored pool called **mypool** using the first two of the five disks you recently created:

```
root@solaris:~# zpool create mypool mirror c7t2d0 c7t3d0
```

Check that your pool has been created:

```
root@solaris:~# zpool list
NAME     SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
mypool   187M   346K   187M   0%  1.00x  ONLINE  -
rpool   19.6G  6.97G  12.7G  35%  1.00x  ONLINE  -
```

Display the file systems using the **zfs** command and the **list** subcommand:

```
root@solaris:~# zfs list
```

```
NAME                              USED  AVAIL  REFER  MOUNTPOINT
mypool                            272K   155M    31K  /mypool
rpool                            7.02G  12.3G  4.58M  /rpool
rpool/ROOT                       4.87G  12.3G    31K  legacy
rpool/ROOT/solaris               4.86G  12.3G  3.92G  /
rpool/ROOT/solaris-backup-1      2.47M  12.3G  1.98G  /
rpool/ROOT/solaris-backup-1/var    46K  12.3G   758M  /var
rpool/ROOT/solaris/var             865M  12.3G   207M  /var
rpool/VARSHARE                     100K  12.3G   100K  /var/share
rpool/dump                       1.03G  12.3G  1.00G  -
rpool/export                     87.6M  12.3G    32K  /export
rpool/export/home                87.5M  12.3G    32K  /export/home
rpool/export/home/trainee        87.5M  12.3G  87.5M  /export/home/trainee
rpool/swap                       1.03G  12.3G  1.00G  -
```

Note that the zpool command automatically creates a file system on **mypool** and mounts it at **/mypool**.

Adding File Systems to an Existing Pool

Now create two file systems in your pool called **/home** and **/home/user1** and then display the results :

```
root@solaris:~# zfs create mypool/home
root@solaris:~# zfs create mypool/home/user1
root@solaris:~# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
mypool                            184K   155M    32K  /mypool
mypool/home                        63K   155M    32K  /mypool/home
mypool/home/user1                  31K   155M    31K  /mypool/home/user1
rpool                            7.02G  12.3G  4.58M  /rpool
```

```
rpool/ROOT                          4.87G  12.3G    31K  legacy
rpool/ROOT/solaris                  4.86G  12.3G  3.92G  /
rpool/ROOT/solaris-backup-1         2.47M  12.3G  1.98G  /
rpool/ROOT/solaris-backup-1/var       46K  12.3G   758M  /var
rpool/ROOT/solaris/var               865M  12.3G   207M  /var
rpool/VARSHARE                       100K  12.3G   100K  /var/share
rpool/dump                          1.03G  12.3G  1.00G  -
rpool/export                        87.6M  12.3G    32K  /export
rpool/export/home                   87.5M  12.3G    32K  /export/home
rpool/export/home/trainee           87.5M  12.3G  87.5M  /export/home/trainee
rpool/swap                          1.03G  12.3G  1.00G  -
```

Note that the two file systems **share** the same disk space as the parent pool.

Changing the Pool Mount Point

Suppose that you want the /home file system mounted elsewhere rather than under the /mypool mount point. With ZFS, this is very simple:

```
root@solaris:~# mkdir /users
root@solaris:~# zfs set mountpoint=/users mypool/home
root@solaris:~# zfs list
NAME                               USED  AVAIL  REFER  MOUNTPOINT
mypool                             196K   155M    31K  /mypool
mypool/home                         63K   155M    32K  /users
mypool/home/user1                   31K   155M    31K  /users/user1
rpool                              7.02G  12.3G  4.58M  /rpool
rpool/ROOT                         4.87G  12.3G    31K  legacy
rpool/ROOT/solaris                 4.86G  12.3G  3.92G  /
rpool/ROOT/solaris-backup-1        2.47M  12.3G  1.98G  /
rpool/ROOT/solaris-backup-1/var      46K  12.3G   758M  /var
```

```
rpool/ROOT/solaris/var             865M  12.3G   207M  /var
rpool/VARSHARE                     100K  12.3G   100K  /var/share
rpool/dump                        1.03G  12.3G  1.00G  -
rpool/export                      87.6M  12.3G    32K  /export
rpool/export/home                 87.5M  12.3G    32K  /export/home
rpool/export/home/trainee         87.5M  12.3G  87.5M  /export/home/trainee
rpool/swap                        1.03G  12.3G  1.00G  -
```

Note that ZFS has automatically and transparently unmounted **/mypool/home** and re-mounted it at **/users**.

Adding a Hot Spare

To display all of the properties associated with **mypool**, use the **zpool** command and the **get** subcommand:

```
root@solaris:~# zpool get all mypool
NAME    PROPERTY       VALUE                  SOURCE
mypool  allocated      673K                   -
mypool  altroot        -                      default
mypool  autoexpand     off                    default
mypool  autoreplace    off                    default
mypool  bootfs         -                      default
mypool  cachefile      -                      default
mypool  capacity       0%                     -
mypool  dedupditto     0                      default
mypool  dedupratio     1.00x                  -
mypool  delegation     on                     default
mypool  failmode       wait                   default
mypool  free           186M                   -
mypool  guid           6502877439742337134    -
```

```
mypool  health         ONLINE               -
mypool  listshares     off                  default
mypool  listsnapshots  off                  default
mypool  readonly       off                  -
mypool  size           187M                 -
mypool  version        34                   default
```

Note that the **autoreplace** property is set to **off**. In order to use a hot spare, this property needs to be set to **on**.

Set the autoreplace property to on:

```
root@solaris:~# zpool set autoreplace=on mypool
root@solaris:~# zpool get autoreplace mypool
NAME    PROPERTY     VALUE  SOURCE
mypool  autoreplace  on     local
```

Add the fourth 200 Mb disk that you have created to **mypool** as a spare:

```
root@solaris:~# zpool add mypool spare c7t5d0
root@solaris:~# zpool status mypool
  pool: mypool
 state: ONLINE
  scan: none requested
config:

    NAME        STATE     READ WRITE CKSUM
    mypool      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
        c7t2d0  ONLINE       0     0     0
        c7t3d0  ONLINE       0     0     0
```

```
    spares
      c7t5d0    AVAIL

errors: No known data errors
```

Observing Pool Activity

Create a random data file in **/users/user1**:

```
root@solaris:~# cat /dev/urandom > /users/user1/randomfile &
[1] 2617
```

Write down the PID, you will need it in 2 minutes to kill the process you have just started.

Now display the writes to the pool using the **iostat** subcommand of the **zpool** command:

```
root@solaris:~# zpool iostat -v 3
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
mypool      29.6M   157M      0      5    184  50.6K
  mirror    29.6M   157M      0      5    184  50.6K
    c7t2d0      -      -      0      5  1.46K  59.8K
    c7t3d0      -      -      0      5  1.33K  56.9K
----------  -----  -----  -----  -----  -----  -----
rpool       6.96G  12.7G      0     14  27.3K   124K
  c7t0d0s1  6.96G  12.7G      0     14  27.3K   124K
----------  -----  -----  -----  -----  -----  -----

               capacity     operations    bandwidth
```

```
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
mypool      39.9M   147M      0    127      0  2.89M
  mirror    39.9M   147M      0    127      0  2.89M
    c7t2d0      -      -      0    113      0  2.78M
    c7t3d0      -      -      0    110      0  2.89M
----------  -----  -----  -----  -----  -----  -----
rpool       6.96G  12.7G      0     13      0  29.5K
  c7t0d0s1  6.96G  12.7G      0     13      0  29.5K
----------  -----  -----  -----  -----  -----  -----

              capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
mypool      53.3M   134M      0    128      0  2.78M
  mirror    53.3M   134M      0    128      0  2.78M
    c7t2d0      -      -      0    112      0  2.44M
    c7t3d0      -      -      0    113      0  2.78M
----------  -----  -----  -----  -----  -----  -----
rpool       6.96G  12.7G      0     77      0   500K
  c7t0d0s1  6.96G  12.7G      0     77      0   500K
----------  -----  -----  -----  -----  -----  -----

              capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
mypool      65.5M   121M      0    171      0  3.08M
  mirror    65.5M   121M      0    171      0  3.08M
    c7t2d0      -      -      0    153      0  3.08M
    c7t3d0      -      -      0    153      0  3.08M
----------  -----  -----  -----  -----  -----  -----
rpool       6.96G  12.7G      0     21      0  45.1K
  c7t0d0s1  6.96G  12.7G      0     21      0  45.1K
----------  -----  -----  -----  -----  -----  -----
```

```
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
mypool      75.8M   111M      0    172      0  2.88M
  mirror    75.8M   111M      0    172      0  2.88M
    c7t2d0      -      -      0    149      0  2.85M
    c7t3d0      -      -      0    149      0  2.88M
----------  -----  -----  -----  -----  -----  -----
rpool       6.96G  12.7G      0      0      0    882
  c7t0d0s1  6.96G  12.7G      0      0      0    882
----------  -----  -----  -----  -----  -----  -----

^C
```

Is your mirror functioning ?

Now kill the process creating the file **randomfile** :

```
# kill -9 PID [Entrée]
```

Delete the file **/users/user1/randomfile**:

```
root@solaris:~# rm -rf /users/user1/randomfile
[1]+  Killed                 cat /dev/urandom > /users/user1/randomfile
```

Setting a User Quota

To set a user quota, you need to use the **set** subcommand of **zpool**:

```
root@solaris:~# zfs set quota=50M mypool/home/user1
```

```
root@solaris:~# zfs get quota mypool
NAME    PROPERTY  VALUE  SOURCE
mypool  quota     none   default
root@solaris:~# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
mypool                            380K   155M    31K  /mypool
mypool/home                        63K   155M    32K  /users
mypool/home/user1                  31K  50.0M    31K  /users/user1
rpool                            7.03G  12.3G  4.58M  /rpool
rpool/ROOT                       4.87G  12.3G    31K  legacy
rpool/ROOT/solaris               4.86G  12.3G  3.92G  /
rpool/ROOT/solaris-backup-1      2.47M  12.3G  1.98G  /
rpool/ROOT/solaris-backup-1/var    46K  12.3G   758M  /var
rpool/ROOT/solaris/var             865M  12.3G   207M  /var
rpool/VARSHARE                     101K  12.3G   101K  /var/share
rpool/dump                       1.03G  12.3G  1.00G  -
rpool/export                     89.3M  12.3G    32K  /export
rpool/export/home                89.3M  12.3G    32K  /export/home
rpool/export/home/trainee        89.3M  12.3G  89.3M  /export/home/trainee
rpool/swap                       1.03G  12.3G  1.00G  -
```

Note that the quota of 50 Mb has been set on /mypool/home/user1.

Now create a random data file in /users/user1:

```
root@solaris:~# cat /dev/urandom > /users/user1/testfile
cat: output error (0/131072 characters written)
Disc quota exceeded
```

After a few minutes, you will see the **Disc quota exceeded** message.

Looking at the available disk space on /users/user1 you will notice that the value is now 0:

```
root@solaris:~# zfs list mypool/home/user1
NAME                USED  AVAIL  REFER  MOUNTPOINT
mypool/home/user1  50.1M      0  50.1M  /users/user1
```

Delete the **testfile** file:

```
root@solaris:~# rm -f /users/user1/testfile
```

Setting a User Reservation

As with setting quotas, setting a reservation is very simple:

```
root@solaris:~# zfs set reservation=25M mypool/home/user1
root@solaris:~# zfs get reservation mypool/home/user1
NAME               PROPERTY     VALUE  SOURCE
mypool/home/user1  reservation  25M    local
```

Using Snapshots

Create a file in **/users/user1**:

```
root@solaris:~# echo "This is a test file for the first snapshot" > /users/user1/snapshot1
root@solaris:~# ls /users/user1
snapshot1
```

To create a snapshot of a ZFS file system, you need to use the **snapshot** subcommand of the **zfs** command:

```
root@solaris:~# zfs snapshot mypool/home/user1@Dec13
```

The snapshot is located in a hidden directory under **/users/user1** called **.zfs/snapshot**:

```
root@solaris:~# ls -l /users/user1/.zfs/snapshot
total 3
drwxr-xr-x   2 root     root           3 Dec 13 10:24 Dec13
```

As you can see, the snapshot contains the **snapshot1** file:

```
root@solaris:~# ls -l /users/user1/.zfs/snapshot/Dec13/
total 2
-rw-r--r--   1 root     root          43 Dec 13 10:24 snapshot1
```

It is important to note here that the .zfs directory is also hidden from the **ls** command, even when using the **-a** switch:

```
root@solaris:~# ls -laR /users/user1
/users/user1:
total 8
drwxr-xr-x   2 root     root           3 Dec 13 10:24 .
drwxr-xr-x   3 root     root           3 Dec 12 16:09 ..
-rw-r--r--   1 root     root          43 Dec 13 10:24 snapshot1
```

You can also create a recursive snapshot of all file systems in a pool:

```
root@solaris:~# zfs snapshot -r mypool@Dec13-1
```

The snapshots are stored in their respective .zfs directories:

```
root@solaris:~# ls /users/.zfs/snapshot
Dec13-1
root@solaris:~# ls /users/user1/.zfs/snapshot
Dec13    Dec13-1
```

You can list all snapshots as follows:

```
root@solaris:~# zfs list -t snapshot -r mypool
NAME                        USED  AVAIL  REFER  MOUNTPOINT
mypool@Dec13-1                 0      -    31K  -
mypool/home@Dec13-1            0      -    32K  -
mypool/home/user1@Dec13        0      -  31.5K  -
mypool/home/user1@Dec13-1      0      -  31.5K  -
```

Create another file in **/users/user1**:

```
root@solaris:~# echo "This is a test file for the second snapshot" > /users/user1/snapshot2
root@solaris:~# ls -l /users/user1
total 4
-rw-r--r--   1 root     root          43 Dec 13 10:44 snapshot1
-rw-r--r--   1 root     root          44 Dec 13 10:45 snapshot2
root@solaris:~# cat /users/user1/snapshot1
This is a test file for the first snapshot
root@solaris:~# cat /users/user1/snapshot2
This is a test file for the second snapshot
```

Now take a second recursive snapshot of **mypool**:

```
root@solaris:~# zfs snapshot -r mypool@Dec13-2
root@solaris:~# zfs list -t snapshot -r mypool
NAME                        USED  AVAIL  REFER  MOUNTPOINT
mypool@Dec13-1                 0      -    31K  -
mypool@Dec13-2                 0      -    31K  -
mypool/home@Dec13-1            0      -    32K  -
mypool/home@Dec13-2            0      -    32K  -
mypool/home/user1@Dec13        0      -  31.5K  -
mypool/home/user1@Dec13-1      0      -  31.5K  -
mypool/home/user1@Dec13-2      0      -    33K  -
```

The **diff** subcommand of the **zfs** command displays the differences between two snapshots:

```
root@solaris:~# zfs diff mypool/home/user1@Dec13-1 mypool/home/user1@Dec13-2
M   /users/user1/
M   /users/user1/snapshot1
+   /users/user1/snapshot2
```

The above out put shows that **/users/user1/snapshot2** has been added and included in the **second** snapshot as it appears in the command line.

This output can contain the following characters:

Character	Description
M	**M**odification
R	**R**enamed
+	Added
-	Deleted

Note that you cannot compare the snapshots in the reverse order:

```
root@solaris:~# zfs diff mypool/home/user1@Dec13-2 mypool/home/user1@Dec13-1
Unable to obtain diffs: mypool/home/user1@Dec13-1 is not a descendant dataset of mypool/home/user1@Dec13-2
```

Rolling Back to a Snapshot

In the case that you wish to rollback to a specific snapshot, note that you can **only** roll back to the last snapshot as shown by the output of **zfs list**:

```
root@solaris:~# zfs list -t snapshot -r mypool
NAME                      USED  AVAIL  REFER  MOUNTPOINT
mypool@Dec13-1               0      -    31K  -
mypool@Dec13-2               0      -    31K  -
```

```
mypool/home@Dec13-1           0      -    32K  -
mypool/home@Dec13-2           0      -    32K  -
mypool/home/user1@Dec13       0      -  31.5K  -
mypool/home/user1@Dec13-1     0      -  31.5K  -
mypool/home/user1@Dec13-2     0      -    33K  -
root@solaris:~# zfs rollback mypool/home/user1@Dec13-1
cannot rollback to 'mypool/home/user1@Dec13-1': more recent snapshots exist
use '-r' to force deletion of the following snapshots:
mypool/home/user1@Dec13-2
```

Delete the **Dec13-2** snapshot as follows:

```
root@solaris:~# zfs destroy mypool/home/user1@Dec13-2
root@solaris:~# zfs list -t snapshot -r mypool
NAME                        USED  AVAIL  REFER  MOUNTPOINT
mypool@Dec13-1                 0      -    31K  -
mypool@Dec13-2                 0      -    31K  -
mypool/home@Dec13-1            0      -    32K  -
mypool/home@Dec13-2            0      -    32K  -
mypool/home/user1@Dec13        0      -  31.5K  -
mypool/home/user1@Dec13-1      0      -  31.5K  -
```

Now roll back to **Dec13-1**:

```
root@solaris:~# zfs rollback mypool/home/user1@Dec13-1
root@solaris:~# ls -l /users/user1
total 2
-rw-r--r--   1 root     root          43 Dec 13 10:24 snapshot1
```

Note that the **snapshot2** file has obviously disappeared since it was not in the **Dec13-1** snapshot.

Cloning a Snapshot

Snapshots are read-only. To convert a snapshot to a writable file system, you can use the **clone** subcommand of the **zfs** command:

```
root@solaris:~# zfs clone mypool/home/user1@Dec13-1 mypool/home/user3
root@solaris:~# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
mypool                           25.5M   129M    31K  /mypool
mypool/home                      25.1M   129M    34K  /users
mypool/home/user1                50.5K  50.0M  31.5K  /users/user1
mypool/home/user3                  18K   129M  31.5K  /users/user3
rpool                            7.03G  12.3G  4.58M  /rpool
rpool/ROOT                       4.87G  12.3G    31K  legacy
rpool/ROOT/solaris               4.86G  12.3G  3.92G  /
rpool/ROOT/solaris-backup-1      2.47M  12.3G  1.98G  /
rpool/ROOT/solaris-backup-1/var    46K  12.3G   758M  /var
rpool/ROOT/solaris/var            865M  12.3G   207M  /var
rpool/VARSHARE                    102K  12.3G   102K  /var/share
rpool/dump                       1.03G  12.3G  1.00G  -
rpool/export                     90.5M  12.3G    32K  /export
rpool/export/home                90.5M  12.3G    32K  /export/home
rpool/export/home/trainee        90.4M  12.3G  90.4M  /export/home/trainee
rpool/swap                       1.03G  12.3G  1.00G  -
```

Display the contents of **/users/user3**:

```
root@solaris:~# ls -l /users/user3
total 2
-rw-r--r--   1 root     root          43 Dec 13 10:24 snapshot1
```

Using Compression

In order to minimize storage space, you can make a file system use compression. Compression can be activated either at creation time or after creation. Compression only works for new data. Any existing data in the file system at the time of activating compression remains uncompressed.

To activate compression on an existing file system, you need to change the file system's **compression** property from off to on:

```
root@solaris:~# zfs set compression=on mypool/home/user1
root@solaris:~# zfs get compression mypool/home/user1
NAME               PROPERTY     VALUE  SOURCE
mypool/home/user1  compression  on     local
```

Using De-duplication

Another space saving property of ZFS file systems is **De-duplication**:

```
root@solaris:~# zfs set dedup=on mypool/home/user1
root@solaris:~# zfs get dedup mypool/home/user1
NAME               PROPERTY  VALUE  SOURCE
mypool/home/user1  dedup     on     local
```

Using Encryption

Unlike **Compression** and **De-duplication**, **Encryption** can only be set on a file system at the time of creation:

```
root@solaris:~# zfs create -o encryption=on mypool/home/user2
Enter passphrase for 'mypool/home/user2': fenestros
Enter again: fenestros
```

Note that the passphrase is not shown in the real output of the command. It is in the above example only for the purposes of this lesson.

To check if encryption is active on a file system, use the following command:

```
root@solaris:~# zfs get encryption mypool/home/user1
NAME               PROPERTY    VALUE  SOURCE
mypool/home/user1  encryption  off    -
root@solaris:~# zfs get encryption mypool/home/user2
NAME               PROPERTY    VALUE  SOURCE
mypool/home/user2  encryption  on     local
```

Replacing a Faulty Disk

In the case of a faulty disk and no hot spares, replacing the disk is a one-line operation using the **replace** subcommand of the **zpool** command:

```
root@solaris:~# zpool status mypool
  pool: mypool
 state: ONLINE
  scan: none requested
config:

    NAME        STATE     READ WRITE CKSUM
    mypool      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
        c7t2d0  ONLINE       0     0     0
        c7t3d0  ONLINE       0     0     0
    spares
      c7t5d0    AVAIL

errors: No known data errors
```

```
root@solaris:~# zpool replace mypool c7t2d0 c7t4d0
```

Use the **status** subcommand of the **zpool** command again to see what has happened:

```
root@solaris:~# zpool status mypool
  pool: mypool
 state: ONLINE
  scan: resilvered 601K in 0h0m with 0 errors on Thu Dec 13 11:45:49 2012
config:

    NAME        STATE     READ WRITE CKSUM
    mypool      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
        c7t4d0  ONLINE       0     0     0
        c7t3d0  ONLINE       0     0     0
    spares
      c7t5d0    AVAIL

errors: No known data errors
```

ZFS *Resilvering* is the equivalent of UFS re-synchronization.

Destroying a Pool

Destroying a pool is achieved by using the **destroy** subcommand of the **zpool** command:

```
root@solaris:~# zpool destroy mypool
```

As you can see by the following output, this operation has also destroyed all the associated snapshots:

```
root@solaris:~# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
rpool                            7.03G  12.3G  4.58M  /rpool
rpool/ROOT                       4.87G  12.3G    31K  legacy
rpool/ROOT/solaris               4.86G  12.3G  3.92G  /
rpool/ROOT/solaris-backup-1      2.47M  12.3G  1.98G  /
rpool/ROOT/solaris-backup-1/var    46K  12.3G   758M  /var
rpool/ROOT/solaris/var            865M  12.3G   207M  /var
rpool/VARSHARE                    102K  12.3G   102K  /var/share
rpool/dump                       1.03G  12.3G  1.00G  -
rpool/export                     90.5M  12.3G    32K  /export
rpool/export/home                90.5M  12.3G    32K  /export/home
rpool/export/home/trainee        90.4M  12.3G  90.4M  /export/home/trainee
rpool/swap                       1.03G  12.3G  1.00G  -
root@solaris:~# zfs list -t snapshot -r mypool
cannot open 'mypool': filesystem does not exist
root@solaris:~# ls -l /users
total 0
```

As you have seen above, destroying a pool, **all** the data in it and **all** the associated snapshots is disconcertingly simple. You should therefore be very careful when using the **destroy** subcommand.

Creating a RAID-5 Pool

You can create a RAID-5 pool using the RAID-Z algorithm:

```
root@solaris:~# zpool create mypool raidz c7t2d0 c7t3d0 c7t4d0 spare c7t5d0
root@solaris:~# zpool status mypool
```

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

    NAME        STATE     READ WRITE CKSUM
    mypool      ONLINE       0     0     0
      raidz1-0  ONLINE       0     0     0
        c7t2d0  ONLINE       0     0     0
        c7t3d0  ONLINE       0     0     0
        c7t4d0  ONLINE       0     0     0
    spares
      c7t5d0    AVAIL

errors: No known data errors
```

Destroy **mypool** :

```
root@solaris:~# zpool destroy mypool
```

Creating a RAID-6 Pool

You can create a RAID-6 pool using the RAID-Z2 algorithm:

```
root@solaris:~# zpool create mypool raidz2 c7t2d0 c7t3d0 c7t4d0 c7t5d0 spare c7t6d0
root@solaris:~# zpool status mypool
  pool: mypool
 state: ONLINE
  scan: none requested
config:

    NAME        STATE     READ WRITE CKSUM
    mypool      ONLINE       0     0     0
```

```
      raidz2-0  ONLINE       0     0     0
        c7t2d0  ONLINE       0     0     0
        c7t3d0  ONLINE       0     0     0
        c7t4d0  ONLINE       0     0     0
        c7t5d0  ONLINE       0     0     0
    spares
      c7t6d0    AVAIL

errors: No known data errors
```

Destroy **mypool** :

```
root@solaris:~# zpool destroy mypool
```

Create a triple parity RAID **mypool** using your five 200MB disks. Do not delete it.

Displaying the Zpool History

You can review everything that has been done to existing pools by using the **history** subcommand of the **zpool** command:

```
root@solaris:~# zpool history
History for 'mypool':
2012-12-13.14:02:17 zpool create mypool raidz3 c7t2d0 c7t3d0 c7t4d0 c7t5d0 spare c7t6d0

History for 'rpool':
2012-11-20.19:08:05 zpool create -f -B rpool c7t0d0s1
2012-11-20.19:08:05 zfs create -p -o mountpoint=/export rpool/export
2012-11-20.19:08:05 zfs set mountpoint=/export rpool/export
2012-11-20.19:08:05 zfs create -p rpool/export/home
2012-11-20.19:08:06 zfs create -p -o canmount=noauto -o mountpoint=/var/share rpool/VARSHARE
```

```
2012-11-20.19:08:06 zfs set mountpoint=/var/share rpool/VARSHARE
2012-11-20.19:08:07 zfs create -p -V 1024.0m rpool/dump
2012-11-20.19:08:12 zfs create -p -V 1024.0m rpool/swap
2012-11-20.19:08:20 zfs set primarycache=metadata rpool/swap
2012-11-20.19:25:51 zfs set primarycache=metadata rpool/swap
2012-11-20.19:26:25 zfs create rpool/export/home/trainee
2012-11-20.22:45:56 zfs set primarycache=metadata rpool/swap
2012-12-01.14:32:36 zfs set primarycache=metadata rpool/swap
2012-12-03.13:15:45 zfs set primarycache=metadata rpool/swap
2012-12-08.14:33:41 zfs /tmp/be
2012-12-11.15:33:50 zfs set primarycache=metadata rpool/swap
2012-12-12.09:57:00 zfs set primarycache=metadata rpool/swap
```

Note that the history related to destroyed pools has been deleted.

LAB #2 - Managing iSCSI Storage

Installing the COMSTAR Server

Start by installing the COMSTAR storage server software:

```
root@solaris:~# pkg install group/feature/storage-server
           Packages to install:  20
       Create boot environment:  No
Create backup boot environment: Yes
            Services to change:   1

DOWNLOAD                                  PKGS         FILES    XFER (MB)   SPEED
Completed                                 20/20     1023/1023    54.5/54.5  636k/s
```

```
PHASE                                        ITEMS
Installing new actions                   1863/1863
Updating package state database                Done
Updating image state                           Done
Creating fast lookup database                  Done
```

The **COMSTAR target mode framework** runs as the **stmf** service. Check to see if it is enabled:

```
root@solaris:~# svcs \*stmf\*
STATE          STIME    FMRI
disabled       15:43:16 svc:/system/stmf:default
```

Enable the service:

```
root@solaris:~# svcadm enable stmf
root@solaris:~# svcs \*stmf\*
STATE          STIME    FMRI
online         16:01:56 svc:/system/stmf:default
```

You can check the status of the server using the **stmfadm** command:

```
root@solaris:~# stmfadm list-state
Operational Status: online
Config Status     : initialized
ALUA Status       : disabled
ALUA Node         : 0
```

Creating SCSI Logical Units

First you need to create your **Backing Storage Device** within your **mypool** pool:

```
root@solaris:~# zfs create -V 100M mypool/iscsi
```

```
root@solaris:~# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
mypool                            103M  51.6M    31K  /mypool
mypool/iscsi                      103M   155M    16K  -
rpool                            7.40G  11.9G  4.58M  /rpool
rpool/ROOT                       5.22G  11.9G    31K  legacy
rpool/ROOT/solaris               5.22G  11.9G  4.08G  /
rpool/ROOT/solaris-backup-1      2.47M  11.9G  1.98G  /
rpool/ROOT/solaris-backup-1/var    46K  11.9G   758M  /var
rpool/ROOT/solaris-backup-2       127K  11.9G  3.92G  /
rpool/ROOT/solaris-backup-2/var    58K  11.9G   266M  /var
rpool/ROOT/solaris/var             980M  11.9G   209M  /var
rpool/VARSHARE                     102K  11.9G   102K  /var/share
rpool/dump                        1.03G  12.0G  1.00G  -
rpool/export                       108M  11.9G    32K  /export
rpool/export/home                  108M  11.9G    32K  /export/home
rpool/export/home/trainee          108M  11.9G   108M  /export/home/trainee
rpool/swap                        1.03G  12.0G  1.00G  -
```

You can see your raw device in the **/dev/zvol/rdsk/mypool/** directory:

```
root@solaris:~# ls -l /dev/zvol/rdsk/mypool
total 0
lrwxrwxrwx   1 root     root           0 Dec 14 09:42 iscsi -> ../../../..//devices/pseudo/zfs@0:6,raw
```

You can now create a logical unit using the **create-lu** subcommand of the **sbdadm** command:

```
root@solaris:~# sbdadm create-lu /dev/zvol/rdsk/mypool/iscsi
Created the following LU:

          GUID                    DATA SIZE           SOURCE
--------------------------------  -------------------  ----------------
600144f0e2a54e00000050cae6d80001  104857600            /dev/zvol/rdsk/mypool/iscsi
```

Mapping the Logical Unit

In order for the logical unit to be available to initiators, it has to be **mapped**. In order to map the logical device you need its GUID. You can use either one of the two following commands to get that information:

```
root@solaris:~# sbdadm list-lu

Found 1 LU(s)

          GUID                    DATA SIZE           SOURCE
--------------------------------  -------------------  ----------------
600144f0e2a54e00000050cae6d80001  104857600            /dev/zvol/rdsk/mypool/iscsi
```

```
root@solaris:~# stmfadm list-lu -v
LU Name: 600144F0E2A54E00000050CAE6D80001
    Operational Status     : Online
    Provider Name          : sbd
    Alias                  : /dev/zvol/rdsk/mypool/iscsi
    View Entry Count       : 0
    Data File              : /dev/zvol/rdsk/mypool/iscsi
    Meta File              : not set
    Size                   : 104857600
    Block Size             : 512
    Management URL         : not set
    Vendor ID              : SUN
    Product ID             : COMSTAR
    Serial Num             : not set
    Write Protect          : Disabled
    Write Cache Mode Select: Enabled
    Writeback Cache        : Enabled
    Access State           : Active
```

Create simple mapping for this logical unit by using the **add-view** subcommand of the **stmfadm** command:

```
root@solaris:~# stmfadm add-view 600144F0E2A54E00000050CAE6D80001
```

Creating a Target

In order to create a target the **svc:/network/iscsi/target:default** service must be online. Check if it is:

```
root@solaris:~# svcs \*scsi\*
STATE          STIME    FMRI
disabled       15:42:56 svc:/network/iscsi/target:default
online         Dec_12   svc:/network/iscsi/initiator:default
```

Start the service:

```
root@solaris:~# svcadm enable -r svc:/network/iscsi/target:default
root@solaris:~# svcs \*scsi\*
STATE          STIME    FMRI
online         Dec_12   svc:/network/iscsi/initiator:default
online         10:06:54 svc:/network/iscsi/target:default
```

Now create a target using the **create-target** subcommand of the **itadm** command:

```
root@solaris:~# itadm create-target
Target iqn.1986-03.com.sun:02:897fd011-8b3d-cf2b-fc1d-a010bd97d035 successfully created
```

To list the target(s), use the **list-target** subcommand of the **itadm** command:

```
root@solaris:~# itadm list-target
TARGET NAME                                                  STATE    SESSIONS
iqn.1986-03.com.sun:02:897fd011-8b3d-cf2b-fc1d-a010bd97d035  online   0
```

Configuring the Target for Discovery

Finally, you need to configure the target so it can be discovered by initiators:

```
root@solaris:~# devfsadm -i iscsi
```

References

- **The Oracle Technology Network**
