

## ZFS

### Storage Pools - Hot Spares (Update 3 only)

- Since Solaris 10 Update 3, ZFS supports Hot Spares.
- A Hot Spare is a component (i.e. disk, partition, etc.) which will replace a faulted component, either manually or automatically.
- Designate hot spares in the following ways:
  - \* With the *zpool create* command, when initially creating a pool.
  - \* After the pool is created with the *zpool add* command.
- Hot spare devices can also be shared between more than one pool.
- Designating devices as hot spares when the pool is created:-

```
# zpool create tarn c0t2d0s3 c0t2d0s4 spare c0t2d0s5 c0t2d0s6
```

```
# zpool status
```

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

NAME	STATE	READ	WRITE	CKSUM
lake	ONLINE	0	0	0
c0t2d0s0	ONLINE	0	0	0
c0t2d0s1	ONLINE	0	0	0

```
errors: No known data errors
```

```
PTO...
```

## ZFS

**Storage Pools - Hot Spares - Adding spares**

pool: tarn  
state: ONLINE  
scrub: none requested  
config:

NAME	STATE	READ	WRITE	CKSUM
tarn	ONLINE	0	0	0
c0t2d0s3	ONLINE	0	0	0
c0t2d0s4	ONLINE	0	0	0
<b>spares</b>				
<b>c0t2d0s5</b>	<b>AVAIL</b>			
<b>c0t2d0s6</b>	<b>AVAIL</b>			

errors: No known data errors

- Designating devices as hot spares after the pool is created:-

```
# zpool add -f lake spare c0t2d0s5 c0t2d0s6
```

```
# zpool status
```

pool: lake  
state: ONLINE  
scrub: none requested  
config:

NAME	STATE	READ	WRITE	CKSUM
lake	ONLINE	0	0	0
c0t2d0s0	ONLINE	0	0	0
c0t2d0s1	ONLINE	0	0	0
<b>spares</b>				
<b>c0t2d0s5</b>	<b>AVAIL</b>			
<b>c0t2d0s6</b>	<b>AVAIL</b>			

errors: No known data errors

## ZFS

### Storage Pools - Hot Spares - removing spares

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

```
NAME      STATE    READ WRITE CKSUM
tarn      ONLINE  0   0   0
  c0t2d0s3 ONLINE   0   0   0
  c0t2d0s4 ONLINE   0   0   0
spares
  c0t2d0s5 AVAIL
  c0t2d0s6 AVAIL
```

errors: No known data errors

- To remove a Hot Spare:-

```
# zpool remove lake c0t2d0s6
```

```
# zpool remove tarn c0t2d0s6
```

\* The above commands remove the device *c0t2d0s6* from pools *lake* and *tarn*.

## ZFS

### Storage Pools - Hot Spares in use

- In theory Hot Spares are used automatically, but this depends upon the *fmd* ZFS agent (Remember SMF, etc?) which is working by default.
- To replace a faulted component with a Hot Spare, if replacement has not been carried out automatically, use the *zpool replace* command as shown previously.
- Here is an example showing an automatic component replacement:-

```
# zpool create lake mirror c0t2d0s0 c0t2d0s1 spare c0t2d0s3
      *      Create a simple pool with a two-way mirror and
              one spare.
```

#### # zpool status

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

NAME	STATE	READ	WRITE	CKSUM
lake	ONLINE	0	0	0
mirror	ONLINE	0	0	0
c0t2d0s0	ONLINE	0	0	0
c0t2d0s1	ONLINE	0	0	0
spares				
c0t2d0s3	AVAIL			

```
errors: No known data errors
```

## ZFS

## Storage Pools - Hot Spares in use

\* Create a dataset and add some data.

```
# zfs create lake/workdir
```

```
# cp -r /platform/sun4u /lake/workdir
```

```
# zpool list
```

```
NAME SIZE USED AVAIL CAP HEALTH ALTROOT
lake 11.6G 267M 11.4G 2% ONLINE -
```

- Now corrupt one of the sides of the mirror, BEING VERY CAREFUL WITH THE DEVICE NAME!!

```
# dd if=/dev/zero of=/dev/dsk/c0t2d0s1 bs=8192 count=1000
```

```
1000+0 records in
```

```
1000+0 records out
```

```
# zpool status
```

```
pool: lake
```

```
state: ONLINE
```

```
scrub: none requested
```

```
config:
```

NAME	STATE	READ	WRITE	CKSUM
lake	ONLINE	0	0	0
mirror	ONLINE	0	0	0
c0t2d0s0	ONLINE	0	0	0
c0t2d0s1	ONLINE	0	0	0
spares				
c0t2d0s3	AVAIL			

```
errors: No known data errors
```

\* Strange, everything still appears to be OK...

\* Probably because there's been no disk activity, so:

```
# zpool scrub lake
```

## ZFS

## Storage Pools - Hot Spares automatic replacement

# *zpool status*

```
pool: lake
state: DEGRADED
status: One or more devices has experienced an unrecoverable error. An
       attempt was made to correct the error. Applications are unaffected.
action: Determine if the device needs to be replaced, and clear the errors
       using 'zpool clear' or replace the device with 'zpool replace'.
see: http://www.sun.com/msg/ZFS-8000-9P
scrub: resilver in progress for 0h0m, 0.03% done, 0h0m to go
config:
```

NAME	STATE	READ	WRITE	CKSUM
lake	DEGRADED	0	0	0
mirror	DEGRADED	0	0	0
c0t2d0s0	ONLINE	0	0	0
spare	DEGRADED	0	0	0
c0t2d0s1	DEGRADED	0	0	102 too many errors
c0t2d0s3	ONLINE	0	0	0
spares				
c0t2d0s3	INUSE	currently in use		

errors: No known data errors

- Now things look different; the *fmd* daemon's ZFS agents have correctly identified the failed device and brought the spare into use.
- *fmstat* displays the ZFS agent information:-

# *fmstat*

module	ev_recv	ev_acpt	wait	svc_t	%w	%b	open	solve	memsz	bufsz
USII-io-diagnosis	0	0		0.0	10.1	0 0	0 0	0 0	0 0	
ietc. etc.										
syslog-msgs	1	0		0.0	27.1	0 0	0 0	0 0	0 0	
<b>zfs-diagnosis</b>	<b>102</b>	<b>102</b>		<b>0.1</b>	<b>170.6</b>	<b>0 0</b>	<b>1 1</b>	<b>168b</b>	<b>144b</b>	
<b>zfs-retire</b>	<b>1</b>	<b>0</b>		<b>0.0</b>	<b>1780.4</b>	<b>0 0</b>	<b>0 0</b>	<b>0 0</b>	<b>0 0</b>	

## ZFS

**Storage Pools - Hot Spares Information**

- We can use *fmadm* to give us more details:-

```
# fmadm faulty
```

```
-----
TIME          EVENT-ID          MSG-ID          SEVERITY
-----
Oct 23 11:17:53 021e4894-2497-e300-b156-b0a11f9ef382 ZFS-8000-GH  Major
Host          : mussel
Platform      : SUNW,Sun-Blade-100   Chassis_id :
Fault class   : fault.fs.zfs.vdev.checksum
Affects       : zfs://pool=lake/vdev=aa835add3f61ccd4
               faulted but still in service
Problem in    : zfs://pool=lake/vdev=aa835add3f61ccd4
Description   : The number of checksum errors associated with a ZFS device
               exceeded acceptable levels. Refer to
               http://sun.com/msg/ZFS-8000-GH for more information.
Response      : The device has been marked as degraded. An attempt
               will be made to activate a hot spare if available.
Impact        : Fault tolerance of the pool may be compromised.
Action        : Run 'zpool status -x' and replace the bad device.
```

- When the faulted component is replaced it should automatically be brought back into use and the spare disk reinstated as a spare once again, but this depends upon the pool property *autoreplace*, which is *off* by default:-

```
# zpool get autoreplace lake
```

```
NAME          PROPERTY  VALUE      SOURCE
lake          autoreplace  off        default
```

- To set the value of *autoreplace* to *on*:-

```
# zpool set autoreplace=on lake
```

## ZFS

### Storage Pools - Hot Spares Information

- Bear in mind that this is a contrived example of failure.
- Automatic replacement is not going to work as expected, as the component was never physically removed and replaced.
- However, we can force the failed component to be used again, simply by detaching the spare:-

```
# zpool detach lake c0t2d0s3
```

- Now check with zpool status:-

```
# zpool status
```

```
pool: lake
```

```
state: ONLINE
```

```
scrub: resilver completed after 0h0m with 0 errors on Fri Oct 23  
17:12:47 2009
```

```
config:
```

NAME	STATE	READ	WRITE	CKSUM
lake	ONLINE	0	0	0
mirror	ONLINE	0	0	0
c0t2d0s0	ONLINE	0	0	0
c0t2d0s1	ONLINE	0	0	0 315M resilvered
spares				
c0t2d0s3	AVAIL			

```
errors: No known data errors
```

- Everything looks OK.
- Some experimentation and experience with the Hot Spare facility would probably be helpful to appreciate the full extent of its operation in any given configuration.

## ZFS

### Storage Pools - Upgrading Pools from an Earlier Release

- ZFS provides a facility for upgrading storage pools to the current version.
- Use the following command to examine your storage pools:-

```
# zpool upgrade
```

This system is currently running ZFS version 15.

All pools are formatted using this version.

- If you want to upgrade:-

```
# zpool upgrade lake
```

```
      *      Would upgrade the lake pool
```

```
# zpool upgrade -a
```

```
      *      Would upgrade all pools.
```

- To get more detail:-

```
# zpool upgrade -v
```

```
0# zpool upgrade -v
```

This system is currently running ZFS pool version 15.

The following versions are supported:

```
VER  DESCRIPTION
```

- ```
-----
```
- 1 Initial ZFS version
  - 2 Ditto blocks (replicated metadata)
  - 3 Hot spares and double parity RAID-Z
  - 4 zpool history
  - 5 Compression using the gzip algorithm
  - 6 bootfs pool property
  - 7 Separate intent log devices
  - 8 Delegated administration
  - 9 refquota and refreservation properties
  - 10 Cache devices

## ZFS

### Storage Pools - Upgrading Pools from an Earlier Release

- 11 Improved scrub performance
- 12 Snapshot properties
- 13 snapused property
- 14 passthrough-x aclinherit
- 15 user/group space accounting

For more information on a particular version, including supported releases, see:

<http://www.opensolaris.org/os/community/zfs/version/N>

Where 'N' is the version number.

```
-bash-3.00# zpool upgrade
```

This system is currently running ZFS pool version 15.

All pools are formatted using this version.

- Once upgraded, a pool is not accessible from older versions of the software.

## ZFS

### Storage Pools - Upgrading datasets from an Earlier Release

- Since Solaris 10 Update 6, ZFS datasets within pools can also be upgraded:-

# *zfs upgrade -v*

The following filesystem versions are supported:

VER DESCRIPTION

- 
- 1 Initial ZFS filesystem version
  - 2 Enhanced directory entries
  - 3 Case insensitive and File system unique identifier (FUID)
  - 4 userquota, groupquota properties

For more information on a particular version, including supported releases, see:

<http://www.opensolaris.org/os/community/zfs/version/zpl/N>

Where 'N' is the version number.

- Use the following commands to upgrade a specific dataset:-

# *zfs upgrade lake/users*

\* Or more likely you would:-

# *zfs upgrade -r lake/users*

\* Or even more likely you would:-

# *zfs upgrade -a*

\* *-r* also upgrades all descendent datasets.

\* *-a* updates all datasets on all imported pools, i.e. not necessarily all attached ones.

- The file system version and pool version can be upgraded independently.
- Once upgraded, a dataset is not accessible from older versions of the software.

## ZFS

### Storage Pools - Informational commands

- We have already seen some commands being used to obtain information about ZFS:-

```
# zpool status
```

```
# zpool list
```

- Each of the above commands could be given a storage pool name as an argument, for example:-

```
# zpool status lake
```

- Output fields can also be defined:-

```
# zpool list -o name,size
```

```
NAME          SIZE
lake          9.7G
```

```
# zpool list -H -o name,size
```

```
lake 9.7G
```

- The -H option suppresses the heading, and causes the output fields to be tab separated, rather than padded with spaces, making the output more suitable for processing with further commands in a shell script.

```
# zpool iostat
```

```
pool          capacity          operations          bandwidth
used    avail    read write    read write
-----
lake        92.6M  9.6G      0    0        38.4K 26.1K
```

- Use the above command to display I/O statistics.
- Bandwidth numbers are in units per second.

## ZFS

## Storage Pools - Informational commands

- The *iostat* option can also be used with an interval specified:-

```
# zpool iostat 2
```

| pool | capacity |       | operations |       | bandwidth |       |
|------|----------|-------|------------|-------|-----------|-------|
|      | used     | avail | read       | write | read      | write |
| lake | 204M     | 9.49G | 0          | 0     | 34.9K     | 53.1K |
| lake | 239M     | 9.45G | 28         | 67    | 3.56M     | 7.54M |
| lake | 239M     | 9.45G | 18         | 35    | 2.29M     | 4.46M |
| lake | 258M     | 9.44G | 0          | 48    | 57.7K     | 4.43M |
| lake | 258M     | 9.44G | 6          | 3     | 769K      | 410K  |
| lake | 287M     | 9.41G | 6          | 122   | 878K      | 13.9M |
| lake | 287M     | 9.41G | 2          | 2     | 377K      | 9.05K |
| lake | 287M     | 9.41G | 36         | 48    | 4.53M     | 5.34M |
| lake | 297M     | 9.40G | 8          | 4     | 1.10M     | 9.48K |

\* Displays stats every 2 seconds; use ^C to abort.

```
# zpool iostat -v
```

| pool     | capacity |       | operations |       | bandwidth |       |
|----------|----------|-------|------------|-------|-----------|-------|
|          | used     | avail | read       | write | read      | write |
| lake     | 305M     | 9.39G | 0          | 0     | 43.7K     | 74.7K |
| mirror   | 153M     | 4.69G | 0          | 0     | 38.6K     | 37.5K |
| c0t2d0s0 | -        | -     | 0          | 0     | 39.1K     | 38.5K |
| c0t2d0s1 | -        | -     | 0          | 1     | 81        | 114K  |
| mirror   | 152M     | 4.69G | 0          | 0     | 5.11K     | 37.5K |
| c0t2d0s3 | -        | -     | 0          | 0     | 5.13K     | 38.2K |
| c0t2d0s4 | -        | -     | 0          | 0     | 27        | 38.2K |

\* Displays stats for individual components.

- One further option provides detailed status information:-

```
# zpool status -x
```

```
all pools are healthy
```

\* The above output would show fault details if any.

## ZFS

### Storage Pools - Command History (Update 4 8/07)

- A new feature of Solaris 10 Update 4 is ZFS command history recording.
- All successful commands which affect a pool, or dataset within it, (including property changes) are recorded

#### # *zpool history lake*

History for 'lake':

```
2007-10-14.17:53:50 zpool create -f lake mirror c0t2d0s0
c0t2d0s1
```

```
2007-10-14.17:54:34 zfs create lake/users
```

```
2007-10-14.17:54:40 zfs create lake/users/joe
```

```
2007-10-14.17:54:45 zfs create lake/users/fred
```

```
2007-10-14.17:54:53 zfs create lake/users/bert
```

```
2007-10-14.17:56:04 zpool add -f lake mirror c0t2d0s3 c0t2d0s4
```

- If you do not specify a storage pool name, the command history for all pools is displayed.
- Since Solaris 10 Update 6 you can use the *-l* option to the *history* command:-

#### # *zpool history -l lake*

History for 'lake':

```
2009-10-23.11:14:07 zpool create lake mirror c0t2d0s0 c0t2d0s1 spare c0t2d0s3
[user root on mussel:global]
```

```
2009-10-23.11:14:21 zfs create lake/workdir [user root on mussel:global]
```

```
2009-10-23.11:17:51 zpool scrub lake [user root on mussel:global]
```

```
2009-10-23.11:59:22 zpool clear lake [user root on mussel:global]
```

etc..

- This produces extra information about host, user and zone.

## ZFS

## Storage Pools - Zpool Properties

- Pool properties can also be displayed:-

```
# zpool get all lake
```

| NAME | PROPERTY      | VALUE               | SOURCE  |
|------|---------------|---------------------|---------|
| lake | size          | 11.6G               | -       |
| lake | used          | 315M                | -       |
| lake | available     | 11.3G               | -       |
| lake | capacity      | 2%                  | -       |
| lake | altroot       | -                   | default |
| lake | health        | ONLINE              | -       |
| lake | guid          | 4258667727905002105 | default |
| lake | version       | 15                  | default |
| lake | bootfs        | -                   | default |
| lake | delegation    | on                  | default |
| lake | autoreplace   | off                 | default |
| lake | cachefile     | -                   | default |
| lake | failmode      | wait                | default |
| lake | listsnapshots | on                  | default |

- Generally, you will not have to frequently change properties.
- We saw an example earlier of changing the *autoreplace* property to *on* for automatically replacing a spare with a repaired component.
- Other property values can be seen in the *zpool* manual page.

## ZFS

### Storage Pools - Exporting and Importing

- Storage pools can be exported in order to physically move them to another machine, or perhaps share them with another server on a SAN device.
- This is relatively straightforward, but before disconnecting the hardware make sure that you have exported the pool as follows:-

```
# zpool export lake
```

```
cannot unmount '/lake/fa': Device busy
```

```
# zpool export -f lake
```

```
# zpool status
```

```
no pools available
```

- If the system complains “Device Busy” use the *-f* flag to *export* to force, but ensure that there is no process using the pool.
- When importing a pool, you may first have to perform a device reconfiguration (*boot -r* or *devfsadm*), then you can run the *zpool* command as shown on the next page.

## ZFS

### Storage Pools - Exporting and Importing

#### # *zpool import*

pool: lake

id: 9934542050057062740

state: ONLINE

action: The pool can be imported using its name or numeric identifier.

config:

|          |        |
|----------|--------|
| lake     | ONLINE |
| mirror   | ONLINE |
| c0t2d0s2 | ONLINE |
| c0t2d0s1 | ONLINE |
| mirror   | ONLINE |
| c0t2d0s3 | ONLINE |
| c0t2d0s4 | ONLINE |

- The above command detects any available pools for import.
- Now import the pool:-

#### # *zpool import lake*

#### # *df -h*

- \* The *df* output should show that the pool, and any contained datasets, have been mounted.

## ZFS

### Storage Pools - Solaris 10 Update 3 (11/06):-

- You can now import destroyed pools:-

```
# zpool import -D
```

```
pool: lake
```

```
id: 10128464646006909075
```

```
state: ONLINE (DESTROYED)
```

```
action: The pool can be imported using its name or numeric identifier.
```

```
The pool was destroyed, but can be imported using the '-Df' flags.
```

```
config:
```

```
lake ONLINE
```

```
c0t2d0s0 ONLINE
```

```
c0t2d0s1 ONLINE
```

```
# zpool import -Df lake
```

- The above feature does not seem to work with experimental storage pools created from files.
- If you use components of the destroyed pool as part of another pool or filesystem, then the destroyed pool cannot be imported.

## ZFS

### *Exercise*

- Take one of your mirror components offline.
- Copy some more data to the *class* dataset within your storage pool.
- Now put the mirror component back online. Did it resilver OK?
- Replace a mirror component with a spare partition.
- Now reverse the operation to reinstate the original partition.
- Use a command to check the status of the storage pool.
- Display the performance statistics for the pool.

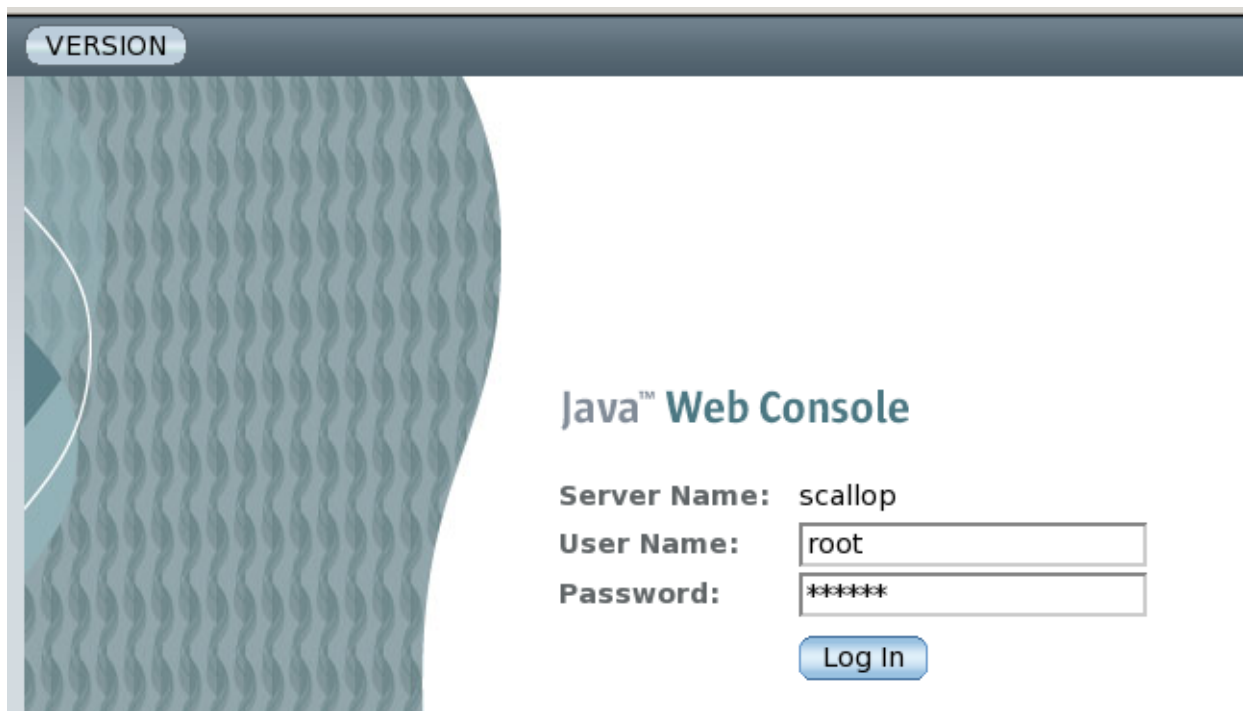
## ZFS

### ZFS GUI

- ZFS has a web browser-based GUI interface.
- You can access the following URL to get a login screen (Note: *https* not *http*):-

<https://localhost:6789>

- You should now see:-



- Login as *root* as shown.
- Only *localhost* is permitted by default; to allow remote access:-

```
# svccfg -s svc:/system/webconsole  
    setprop options/tcp_listen=true
```

```
# svcadm refresh svc:/system/webconsole
```

- If you have problems accessing a remote machine in the classroom, use the IP address rather than the hostname.

# ZFS

## ZFS GUI

- Having logged in, you can choose the ZFS link as shown below:-



- Start Each Application in a New Window

### Systems

No applications available

### Desktop Applica

No applications avail

### Storage

ZFS Administration



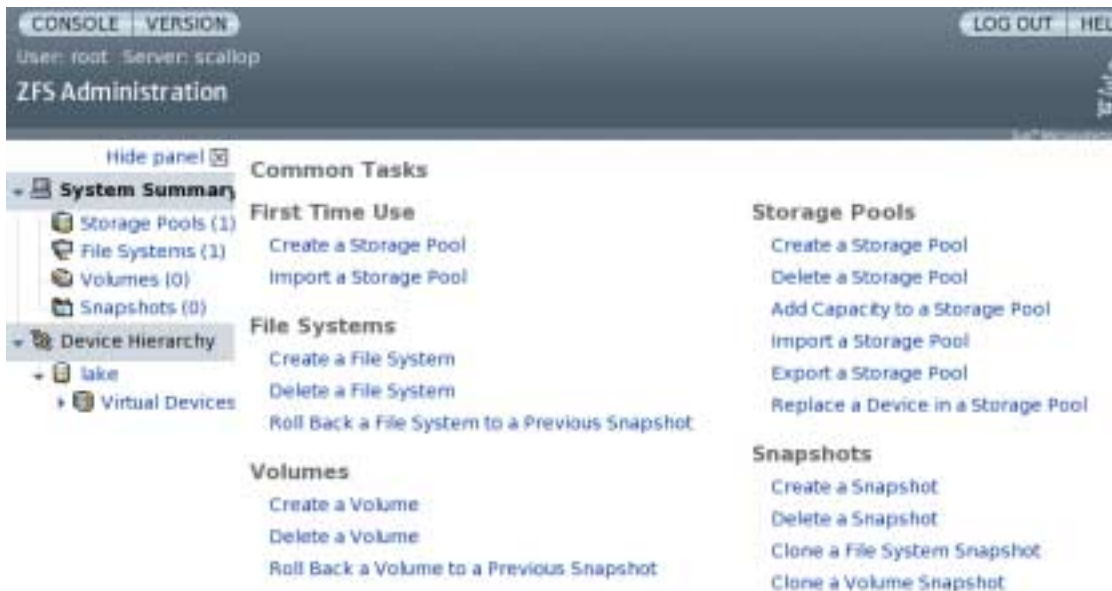
### Other

No applications avail

### Services

No applications available

- Now you'll see:-



## ZFS

### ZFS GUI

- You can select a storage pool and perform admin functions as we have seen from the command line.



- When creating a new pool, or adding capacity to a pool using the wizard, click on *Show Slices* to see all the slices, or you'll see just the whole disks.
- Try using it to carry out some of the functions we have already used from the command line:-
  - \* Destroy the current storage pool, and create a new one from two slices as a mirror.
  - \* Add extra storage using another mirror (two slices) - you may need to do this from the command line.
  - \* Take a mirror component offline. (PTO)
  - \* Create a dataset within the pool.
  - \* Place the offlined component back online.