# Oracle Solaris Cluster 4.x Administration

**Student Guide - Volume II**



D74942GC10
Edition 1.0
February 2012
D75862

ORACLE

**Authors**

Raghavendra JS
Zeeshan Nofil
Venu Poddar

**Technical Contributors and Reviewers**

Thorsten Fruauf
Harish Mallya
Hemachandran Namachivayam
Pramod Rao
Tim Read
Sambit Nayak
Venugopal Navilugon
Shreedhar
Lisa Shepherd
Mahesh Subramanya
Venkateswarlu Tella

**Graphic Designer**

Satish Bettegowda

**Editors**

Raj Kumar
Richard Wallis

**Publishers**

Michael Sebastian
Giri Venugopal

# Contents

7

# Using ZFS with Oracle Solaris Cluster Software

# Objectives

After completing this lesson, you should be able to:

- Build ZFS storage pools and file systems
- Use ZFS for Oracle Solaris Cluster failover data

# Agenda

- Building ZFS storage pools and file systems
- Using ZFS for Oracle Solaris Cluster failover data

# Typical ZFS Configuration

```
# zpool status
 pool: rpool
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        rpool       ONLINE       0     0     0
          c3t0d0s0  ONLINE       0     0     0

errors: No known data errors
# zfs list -r rpool
NAME                              USED  AVAIL  REFER  MOUNTPOINT
rpool                            5.00G  24.3G  39.5K  /rpool
rpool/ROOT                       2.41G  24.3G    31K  legacy
rpool/ROOT/solaris               2.41G  24.3G  1.88G  /
rpool/ROOT/solaris-backup-1       114K  24.3G  1.58G  /
rpool/ROOT/solaris-backup-1/var    61K  24.3G   143M  /var
rpool/ROOT/solaris/var            320M  24.3G  96.2M  /var
rpool/dump                       1.56G  24.3G  1.51G  -
rpool/export                      100K  24.3G    32K  /export
rpool/export/home                  68K  24.3G    32K  /export/home
rpool/export/home/oracle           36K  24.3G    36K  /export/home/oracle
rpool/swap                       1.03G  24.3G  1.00G  -
```



Provisioning of Zeta File System (ZFS) is automated by the Oracle Solaris installer. There are no special requirements for Oracle Solaris Cluster, besides the recommendation for a 750 MB minimum swap space. Note that swap space and dump space are automatically provisioned on ZFS volumes (`zvol`s).

# Typical ZFS Configuration

```
# swap -l
swapfile             dev    swaplo   blocks     free
/dev/zvol/dsk/rpool/swap 124,2         8  2097144  2097144

# dumpadm
       Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash
  Savecore enabled: yes
   Save compressed: on
```

# ZFS in Oracle Solaris Cluster

- ZFS is the default file system in Oracle Solaris 11.
- Leave a slice outside of ZFS reserved for Solaris Volume Manager `metadb` if needed.
- You should plan a dedicated slice for `metadb` on the root disk.



ZFS has little bearing on proper cluster operations. ZFS's advantages in simplicity and flexibility, and the ability to manage and roll back snapshots, make ZFS a preferred choice even if you require other volume managers to manage data.

Recall that the `/global/.devices/node@#` file system must be a UFS. When you have ZFS root, it will likely reside on a loopback file interface (`lofi`) device using `/.globaldevices` as the backing store.

**ZFS Root Pool and Oracle Solaris Cluster**

ZFS typically consists of entire disks. The root pool is different and requires that it be built on top of Solaris partitions.

In typical non-cluster scenarios, the Solaris installer provisions slice 0 of the boot disk to map the entire disk, and creates the root pool using slice 0.

If Solaris Volume Manager is used for shared storage volume management, you should plan a dedicated slice for `metadb` on the root disk.

The only variation that you need to consider in Oracle Solaris Cluster is the need for Solaris Volume Manager dedicated `metadb` partitions if you require Solaris Volume Manager for any or all of your cluster data.

Currently, the interactive installation does not have the ability to create a partition that is not part of the root pool. In this case, you would need separate local disks if you required `metadb`**s**.

# Using ZFS and Snapshots

```
# zfs snapshot rpool/ROOT/solaris@test
...
ok boot -F failsafe
Rebooting with command: boot -F failsafe
Boot device: /pci@1f,700000/scsi@2/disk@0,0:a File and args: -F failsafe
.
.
Hardware watchdog enabled
Configuring devices.
Searching for installed OS instances...
ROOT/s10s_u9wos_14b was found on rpool.
Do you wish to have it mounted read-write on /a? [y,n,?] y
mounting rpool on /a
Starting shell.
```



You can take snapshots of the ZFS file system or of any other file system within the root pool. The operation looks like any other snapshot operation, and you can give your snapshot any name you like:

```
# zfs snapshot rpool/ROOT/solaris@test
```

There is a special procedure for rolling back snapshots of the root file system:

1. Boot or reboot the system by using the `-F` failsafe option:

   ```
   ok boot -F failsafe
   The ok prompt will be displayed only in a SPARC system.
   ```

2. Confirm that the system mounts the root pool for you under `/a` when it prompts you to do so.
3. Roll back your snapshot.
4. Reboot the system.

# Using ZFS and Snapshots

```
# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
rpool                            5.00G  24.3G  39.5K  /rpool
rpool/ROOT                       2.41G  24.3G    31K  legacy
rpool/ROOT/solaris               2.41G  24.3G  1.88G  /
rpool/ROOT/solaris-backup-1       114K  24.3G  1.58G  /
rpool/ROOT/solaris-backup-1/var    61K  24.3G   143M  /var
rpool/ROOT/solaris/var            320M  24.3G  96.2M  /var
rpool/dump                       1.56G  24.3G  1.51G  -
rpool/export                      100K  24.3G    32K  /export
rpool/export/home                  68K  24.3G    32K  /export/home
rpool/export/home/oracle           36K  24.3G    36K
   /export/home/oracle
rpool/swap                       1.03G  24.3G  1.00G  -

# zfs rollback rpool/ROOT/solaris@test
# reboot
```



```
...
Searching for installed operating system (OS) instances...
ROOT/solaris was found on rpool.
Do you wish to have it mounted read-write on /a? [y,n,?] y
mounting rpool on /a

Starting shell.
...
```

# Building ZFS Pools and File Systems

```
# zpool create marcpool mirror /dev/rdsk/
c0t600144F0B3CAAC8300004EF9E6300008d0 /dev/rdsk/
c0t600144F0B3CAAC8300004EF9E6450009d0

# zpool status
 pool: marcpool
 state: ONLINE
  scan: none requested
config:

        NAME                                                STATE     READ WRITE CKSUM
        marcpool                                            ONLINE       0     0     0
          mirror-0                                          ONLINE       0     0     0
            /dev/rdsk/c0t600144F0B3CAAC8300004EF9E6300008d0s0  ONLINE       0     0     0
            /dev/rdsk/c0t600144F0B3CAAC8300004EF9E6450009d0s0  ONLINE       0     0     0

errors: No known data errors
```



In the cluster, you use traditional disk paths (not Device ID [DID]) as components of pools. The pools can still fail over even if the device paths have different names on different nodes.

# Building ZFS Pools and File Systems

```
# zfs create marcpool/test1
# zfs create marcpool/test2


# zfs list -r marcpool
NAME            USED  AVAIL  REFER  MOUNTPOINT
marcpool        162K  9.78G    33K  /marcpool
marcpool/test1   31K  9.78G    31K  /marcpool/test1
marcpool/test2   31K  9.78G    31K  /marcpool/test2
```



ZFS is the default file system. The default mount point for that file system is */poolname*. Now you can create additional file systems within the pool. ZFS automatically creates mount points and mounts the file systems. You do not need to add `/etc/vfstab` entries.

# Building ZFS Pools and File Systems

```
# zfs set mountpoint=/oracle marcpool/test1
# zfs set mountpoint=/shmoracle marcpool/test2
# df -h |grep marcpool
marcpool               9.8G    32K       9.8G     1%    /marcpool
marcpool/test1         9.8G    31K       9.8G     1%    /oracle
marcpool/test2         9.8G    31K       9.8G     1%    /shmoracle
```



The mount points default to */poolname/fsname*, but you can change them.

# Growing a ZFS Storage Pool

You can add new devices at any time by using:

```
# zpool add marcpool mirror c1t3d0 c2t3d0
```



You can grow a ZFS storage pool by adding new devices at any time. There is no such thing as growing or shrinking individual ZFS file systems, and you get much more flexibility by being able to set quotas and reservations on the file systems, as shown in the next section.

When you grow a ZFS storage pool, the system will check that you are using the same kind of underlying RAID as devices already in the pool. Though it is not recommended, you can specify pools with mixed RAID behavior by using the `-f` option to the `zpool` command.

The example shown in the slide adds new mirrored disks into a mirrored pool.

# Quotas and Reservations

- A quota defines the maximum space for a file system.
- A reservation defines guaranteed space.

```
# zfs set quota=20g marcpool/myfs2
# zfs set reservation=6g marcpool/myfs2
```



You control the amount of space associated with each individual file system within the pool by using quotas and reservations.

The default (without quotas and reservations) is that all file systems share all the space in the pool. Any file system can use up all the space in the pool, thereby depriving another file system of space.

A quota sets a maximum amount of space for a file system (to prevent it from using up too much space).

A reservation sets a guaranteed amount of space for the file system. This is just a guaranteed amount and does not translate to any particular physical file blocks.

Quotas and reservations can be grown at any time, within the bounds of the total size of the pool. Quotas and reservations can be shrunk at any time, as long as the current usage of a file system is below the new lowered values. This scheme gives much more flexibility than traditional resizing of legacy types of file systems.

The example shown in the slide sets a quota and a reservation on a file system.

# Quiz

Which of the following statements are true about quotas and reservations?

a. A quota sets a maximum amount of space for a file system.
b. A reservation sets a guaranteed amount of space for a file system.
c. Reservations can be grown after they are set.
d. Quotas cannot be grown after they are set.



**Answer: a, b, c**

# ZFS Snapshots

```
# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
marcpool                          180K  9.78G    31K  /marcpool
marcpool/test1                     31K  9.78G    31K  /oracle
marcpool/test2                     31K  9.78G    31K  /shmoracle

# zfs snapshot marcpool/test1@retest
clnode2:/# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
marcpool                          183K  9.78G    31K  /marcpool
marcpool/test1                     31K  9.78G    31K  /oracle
marcpool/test2                     31K  9.78G    31K  /shmoracle
rpool                             5.00G  24.3G  39.5K  /rpool
# zfs rollback marcpool/test1@retest
```



ZFS has an instantaneous point-in-time snapshot feature.

Initially, snapshots do not consume any additional space pool. If there are no changes to the parent file system, the parent and the snapshot point to the same disk blocks. ZFS implements a copy-on-write policy for the parent; in this way, a ZFS snapshot appears to grow over time because it occupies the different blocks (the original blocks) from the parent after the parent has made modifications.

The parent file system can be rolled back. If you want to roll back to a snapshot that is not the most recent one, you must specify a `-r` option to the `rollback` subcommand which will automatically destroy the snapshots taken after the one you are rolling back to:

```
clnode2:/# zfs rollback marcpool/test1@retest
```

# Zpool Ownership

```
# zpool export marcpool
# zpool import
 pool: nfspool
    id: 9136242729351964297
 state: ONLINE
status: The pool was last accessed by another system.
action: The pool can be imported using its name or numeric identifier and
        the '-f' flag.
   see: http://www.sun.com/msg/ZFS-8000-EY
config:

        nfspool                                    ONLINE
          mirror-0                                 ONLINE
            c0t600144F0B3CAAC8300004EF9E776000Ad0  ONLINE
            c0t600144F0B3CAAC8300004EF9E7E1000Bd0  ONLINE

  pool: marcpool
    id: 9312545695319373888
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:
…
# zpool import marcpool
```



The commands that ZFS uses to transfer ownership of a pool from one host to another are `zpool export` and `zpool import`. The pool is always the unit of ownership. A new node importing a pool does not need to be aware of the pool's presence; the `zpool import` command automatically scans the connected storage looking for pools.

# Using ZFS for Oracle Solaris Cluster Failover Data

- ZFS has a built-in volume management layer.
- ZFS does not require `/etc/vfstab` entries.



ZFS is available as a failover file system for application data. You can store data only for failover applications (not scalable applications); as such, you can run the applications only on nodes that are physically connected to the storage.

The configuration database that ZFS automatically maintains within the pool contains all the mount information for file systems within the pool. You never need to create associated `vfstab` entries.

# Agenda

- Building ZFS storage pools and file systems
- Using ZFS for Oracle Solaris Cluster failover data

# ZFS Pool Automatic Failover in the Cluster

- ZFS is not managed by a device group layer.
- ZFS is managed by application-layer resources.



The Oracle Solaris Cluster device group management layer does not manage ZFS storage pools in any way.

Automating the failover of ZFS storage in the cluster is achieved using application-level resources.

# SUNW.HAStoragePlus for ZFS

SUNW.HAStoragePlus:

- Operates differently for ZFS
- Operates at the granularity of the zpool
- Manages the location of the zpool



SUNW.HAStoragePlus is a resource type that enables application resources that depend on global devices, cluster file systems, failover file systems, or zpool to synchronize their startup with the availability of the storage resources on which they depend.

The zpool is the ZFS volume manager component. SUNW.HAStoragePlus works differently for ZFS. It operates at the granularity of the zpool. That is, SUNW.HAStoragePlus manages the location of the zpool.

# `Failmode` Property

```
zpool set failmode=wait <pool name>
zpool set failmode=continue <pool name>
zpool set failmode=panic <pool name>
zpool set failmode=hard <pool name>
zpool clear
```

**Example:**

```
# zpool set failmode=continue tank
# zpool get failmode tank
NAME   PROPERTY  VALUE     SOURCE
tank   failmode  continue  local
```

ORACLE



The `failmode` property determines the behavior of a catastrophic pool failure due to a loss of device connectivity or the failure of all devices in the pool. The property can be set to these values: `wait`, `continue`, or `panic`. The default value is `wait`, which means you must reconnect the device or replace a failed device, and then clear the error with the `zpool clear` command. `zpool set failmode=wait <pool name>` is the setting intended when HA-ZFS is configured. The `continue` mode returns EIO to any new write request but attempts to satisfy reads. Any write I/Os that were already in-flight at the time of the failure are queued and may be resumed using `zpool clear`.

For failover data services or HA, the resource property may also be reset to `hard`.

The `failmode` property is set like other settable ZFS properties, which can be set either before or after the pool is created.

# Summary

In this lesson, you should have learned how to:

- Build ZFS storage pools and file systems
- Use ZFS for Oracle Solaris Cluster failover data

# Practice 7 Overview: Configuring Volume Management by Using ZFS

This practice covers the following topics:

- Task 1: Creating a ZFS Pool and file system for cluster data
- Task 2: Creating a snapshot and then modifying your data
- Task 3: Manually migrating your ZFS pool to another node

8

# Using Solaris Volume Manager with Oracle Solaris Cluster Software

# Objectives

After completing this lesson, you should be able to:

- Provide an overview of Solaris Volume Manager
- Provide an overview of shared disk sets
- Describe Solaris Volume Manager multiowner disk sets
- Describe creating and managing shared disks
- Build volumes in shared disk sets with soft partitions of mirrors
- Use Solaris Volume Manager status commands
- Perform Oracle Solaris Cluster software-level device group management

# Agenda

- Provide an overview of Solaris Volume Manager
- Describe shared disk sets
- Describe Solaris Volume Manager multiowner disk sets
- Describe creating and managing shared disks
- Build volumes in shared disk sets with soft partitions of mirrors
- Use Solaris Volume Manager status commands
- Perform Oracle Solaris Cluster software-level device group management

# Solaris Volume Manager

- Solaris Volume Manager (SVM) is used to create and manage disk space.
- Solaris Volume Manager supports Redundant Array of Independent/Inexpensive Disks (RAID).
- It also supports dynamic hot spares.



Solaris Volume Manager is a software product that enables you to manage large numbers of disks and the data on those disks. You can use Solaris Volume Manager to increase storage capacity, increase data availability, and ease administration of large storage devices. Solaris Volume Manager can increase the reliability and availability of data by using Redundant Array of Independent Disks or RAID-1 (mirror) volumes and RAID-5 volumes. Solaris Volume Manager hot spares can provide another level of data availability for mirrors and RAID-5 volumes.

A disk set is a set of physical storage volumes that contain logical volumes and hot spares. Volumes and hot spare pools must be built on drives from within that disk set. The disk set provides data availability in a cluster environment.

# Exploring Solaris Volume Manager Disk Space Management

To explore disk space management, you can:

- Use Solaris Volume Manager with and without soft partitions
- Identify the purpose of Solaris Volume Manager disk sets in the cluster
- Manage disk sets
  - Local disk set database replicas (`metadb`)
  - Shared disk set `metadb` replicas
- Initialize Solaris Volume Manager to build cluster disk sets
- Build cluster disk sets
- Mirror disks with soft partitions in cluster disk sets

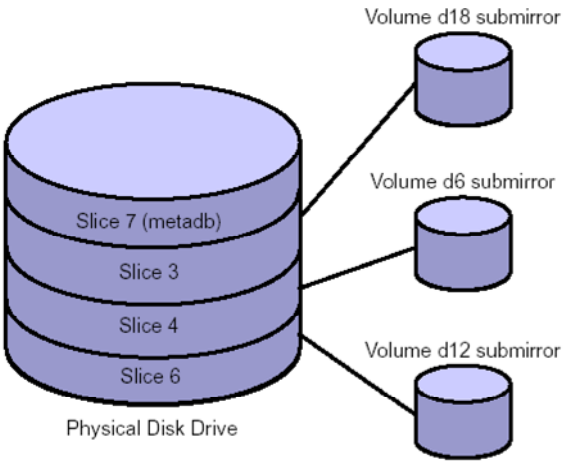
ORACLE



Solaris Volume Manager has two distinct ways of managing disk space. The original Solstice DiskSuite software (Solaris Volume Manager's precursor product) had the restriction that Oracle Solaris Operating System (OS) partitions were the smallest-granularity building blocks for volumes. The current Solaris Volume Manager supports both the traditional way of managing space and the new method, called *soft partitioning*.

# Solaris Volume Manager Partition-Based Disk Space Management

## Traditional disk space management






The figure in the slide illustrates Solaris Volume Manager disk-space management, which equates standard disk partitions with building blocks for virtual volumes.

The following are the limitations of using only partitions as Solaris Volume Manager building blocks:

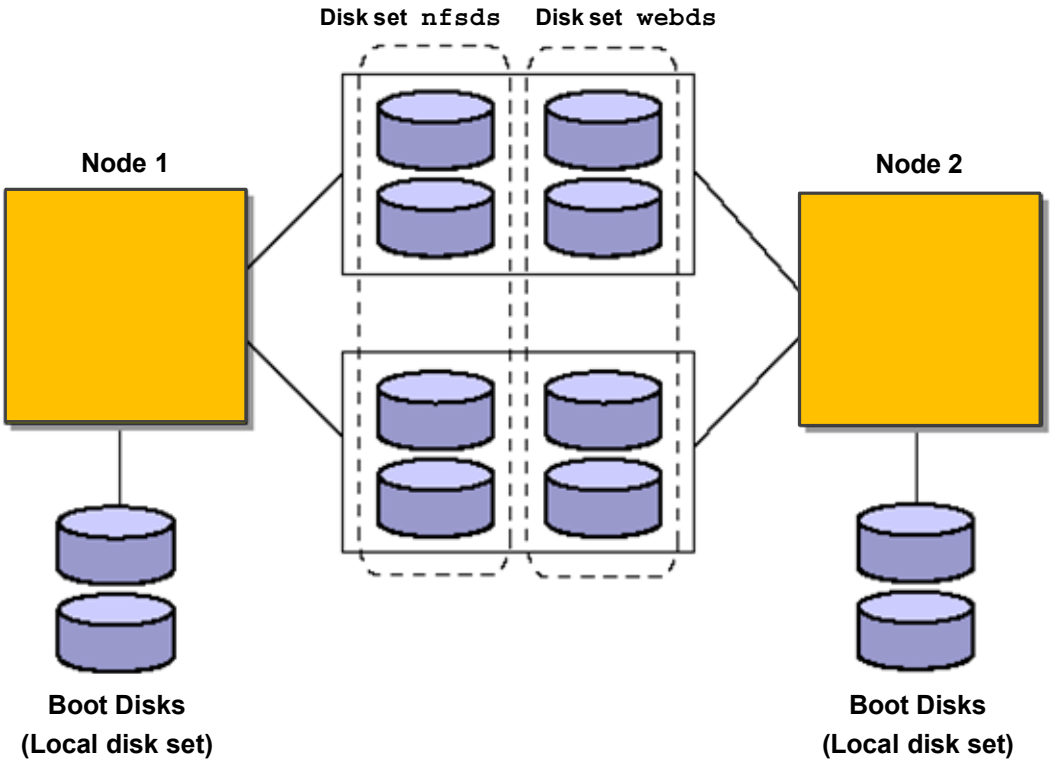
- The number of building blocks per disk or LUN is limited to the traditional seven (or eight, at the very most) partitions. This is particularly restrictive for large LUNs in hardware RAID arrays.
- Disk repair is harder to manage because an old partition table on a replacement disk must be re-created or replicated manually.

**Note:** Traditional VTOC partitions have a slice 7 that is used for the `metadb`, whereas ZFS uses EFI, where slice 6 is the partition to use.

# Agenda

- Provide an overview of Solaris Volume Manager
- Describe shared disk sets
- Describe Solaris Volume Manager multiowner disk sets
- Describe creating and managing shared disks
- Build volumes in shared disk sets with soft partitions of mirrors
- Use Solaris Volume Manager status commands
- Perform Oracle Solaris Cluster software-level device group management

# Exploring Solaris Volume Manager Disk Sets






When you use Solaris Volume Manager to manage data in the Oracle Solaris Cluster environment, all disks that hold the data for cluster data services must be members of Solaris Volume Manager shared disk sets.

Only disks that are physically located in the shared storage are members of the shared disk sets. Only disks that are in the same disk set operate as a unit. They can be used together to build mirrored volumes, and primary ownership of the disk set transfers as a whole from node to node. You should always have disks from different controllers (arrays) in the same disk set so that you can mirror disks across controllers.

Shared disk sets are given a name that often reflects the intended usage of the disk set (for example, `nfsds`).

To create shared disk sets, Solaris Volume Manager requires that each host (node) must have a local disk set on nonshared disks. The only requirement for these local disks is that they have local disk set `metadb`s (which are described later). The figure shows the contents of two shared disk sets in a typical cluster, along with the local disk sets on each node.

# Agenda

- Provide an overview of Solaris Volume Manager
- Describe shared disk sets
- Describe Solaris Volume Manager multiowner disk sets
- Describe creating and managing shared disks
- Build volumes in shared disk sets with soft partitions of mirrors
- Use Solaris Volume Manager status commands
- Perform Oracle Solaris Cluster software-level device group management

# Solaris Volume Manager Multiowner Disk Sets (for Oracle RAC)

A multiowner disk set:

- Allows simultaneous access to the disk set from multiple nodes
- Is currently used only with Oracle RAC
- Requires RAC framework currently



Solaris Volume Manager has a multiowner disk set feature. That is, a multiowner disk set allows more than one node to physically access the storage simultaneously.

In the current implementation, multiowner disk sets are only for use with Oracle RAC software. In fact, you cannot create a multiowner disk set until you have enabled a layer of software underlying Oracle RAC (known as the *RAC framework*). This will be described in more detail, and you will get a chance to work with it, if you choose, in one of the optional practice exercises. In a multiowner disk set, no ownership changes.

Managing multiowner disk sets is identical to managing other disk sets, except that you use the `-M` option when you create the disk set.

*Solaris Volume Manager for Sun Cluster* is the name of the software feature for multiowner disk sets.

# Using Solaris Volume Manager Database Replicas (`metadb` Replicas)

- Small partitions contain Solaris Volume Manager configuration and state information.
  - A separate set for the local disk set
  - A separate set for each shared disk set
- Multiple copies on the same partition are allowed (you might do this to balance the numbers of `metadb` replicas across disks or partitions).



Solaris Volume Manager requires that certain partitions serve as volume database replicas, storing the Solaris Volume Manager configuration and state information in a raw format (nonfile system). These are normally small, dedicated Oracle Solaris OS partitions. In the local disk sets, Solaris Volume Manager allows the creation of `metadb`s on a large partition and then the use of the same partition as a component in a volume. The reasons why you should never do this on the boot disks are described later in this lesson.

There are separate sets of `metadb` replicas for the local disk sets and for each shared disk set. In the previous example (the slide titled “Exploring Solaris Volume Manager Disk Sets,” which shows two nodes and two shared disk sets), there are four distinct collections of `metadb` replicas.

You can put several copies of the `metadb` on the same partition. You might do this to balance the numbers of `metadb` replicas across disks or partitions, for reasons described in the following subsections.

# Local Replica Management

- Add local replicas manually.
- Put them on any dedicated local partition.
- Spread them evenly across local disks and controllers.
- Add at least three (if you have fewer, Solaris Volume Manager logs warnings) replicas.
- Mathematics:
  - If there are fewer than 50% of the replicas remaining, Solaris Volume Manager ceases to operate.
  - If there are exactly 50% of the replicas remaining, the node cannot reboot.
  - If it cannot reboot, delete the broken replicas.



When managing local replicas, note the following:

- You must add local replicas manually.
- You can put local replicas on any dedicated partitions on the local disks. You might prefer to use slice 6 (`s6`) as a convention because the shared disk-set replicas have to be on that partition.
- You must spread local replicas evenly across disks and controllers.
- If you have less than three local replicas, Solaris Volume Manager logs warnings. You can put more than one copy in the same partition to satisfy this requirement. For example, with two local disks, set up a dedicated partition on each disk and put three copies on each disk.

The mathematics for local replicas is as follows:

- If fewer than 50% of the defined `metadb` replicas are available, Solaris Volume Manager ceases to operate.
- If exactly 50% (or fewer) of the defined `metadb` replicas are available at boot time, you cannot boot the node. However, you can boot to single-user mode and just use the `metadb` command to delete ones that are not available.

**Note:** Override the behavior of the last bullet item by inserting the following line in the `/etc/system` file:

```
set md:mirrored_root_flag=1
```

This flag is a misnomer because it has nothing to do with using SVM to mirror root. It simply does let you boot fully with exactly 50% (but not less than 50% ) of local `metadb`s available.

# Agenda

- Provide an overview of Solaris Volume Manager
- Describe shared disk sets
- Describe Solaris Volume Manager multiowner disk sets
- Describe creating and managing shared disks
- Build volumes in shared disk sets with soft partitions of mirrors
- Use Solaris Volume Manager status commands
- Perform Oracle Solaris Cluster software-level device group management

# Shared Disk Set Replica Management

- There are separate collections for each disk set.
- They are automatically added to slice 6 (beginning of disk).
  - Slice 6 if ZFS is used and slice 7 for traditional VTOC partitions
- You must fix manually if they break.
- Mathematics:
  - If there are fewer than 50% of the replicas remaining, the disk set ceases to operate.
  - If there are exactly 50% of the replicas remaining, failovers are not possible.



Consider the following issues when managing shared disk set replicas:

- There are separate `metadb` replicas for each disk set.
- They are automatically added to disks as you add disks to disk sets. They will be, and must remain, on slice 6.
- You must use the `metadb` command to remove and add replicas if you replace a disk containing replicas.

**Shared Disk Set Replica Quorum Mathematics**

The mathematics for shared disk-set replica quorums is as follows:

- If fewer than 50% of the defined replicas for a disk set are available, the disk set ceases to operate.
- If exactly 50% of the defined replicas for a disk set are available, the disk set still operates but it cannot be taken or switched over.

# Initializing the Local `metadb` Replicas on Local Disks

- Use `c#t#d#` for local `metadb` and all local configuration.
- Use DIDs for shared disk sets.
- Add the first `metadb`:

```
# metadb -a -f -c 3 c0t0d0s6
```



Solaris Volume Manager management cannot be performed until you initialize the local `metadb` replicas on each node.

These instructions assume that you have a small partition to use for the `metadb` replicas on your boot disk. In this example, the partition is `s6`.

Make sure you initialize local `metadb` replicas correctly and separately on each node.

**Using DIDs Compared to Using Traditional `c#t#d#`**

In the Oracle Solaris Cluster environment, you can add any `metadb` or partition component by using either its cluster DID (`/dev/did/rdsk/d#s#`) or the traditional `c#t#d#`.

Use the DID naming for all shared disk sets. Without it, you are restricted to having identical controller numbers on each node. If you assume this will always be true, you will be surprised when you add a new node or repair a node.

Use the traditional `c#t#d#` naming scheme for local `metadb` replicas and devices. This makes recovery easier if you need to access these structures when booted in non-clustered mode. Omit `/dev/rdsk` to abbreviate traditional names in all of the Solaris Volume Manager commands.

### Leaving Dedicated Partitions Even When Using ZFS Root

You must have local `metadb` replicas on dedicated local disk partitions even when you use ZFS root. Dedicate a small partition for `metadb`s outside of your ZFS configuration, if you need to use Solaris Volume Manager for your data.

### Adding the Local `metadb` Replicas to the Local Disks

Use the `metadb -a` command to add local `metadb` replicas. As you add the first ones, you must use the `-f` (force) option. The example shown in the slide creates three copies on each of two local disks. If these disks were mirrored using ZFS, you would have to make sure you had a slice 7 of sufficient size dedicated for `metadb`s on each one.

# Initializing the Local `metadb` Replicas on Local Disks

```
# metadb -i
 flags           first blk       block count
     a m  pc luo        16              8192            /dev/dsk/c3t0d0s6
     a    pc luo        8208            8192            /dev/dsk/c3t0d0s6
     a    pc luo        16400           8192            /dev/dsk/c3t0d0s6
 r - replica does not have device relocation information
 o - replica active prior to last mddb configuration change
 u - replica is up to date
 l - locator for this replica was read successfully
 c - replica's location was in /etc/lvm/mddb.cf
 p - replica's location was patched in kernel
 m - replica is master, this is replica selected as input
 t - tagged data is associated with the replica
 W - replica has device write errors
 a - replica is active, commits are occurring to this replica
 M - replica had problem with master blocks
 D - replica had problem with data blocks
 F - replica had format problems
 S - replica is too small to hold current data base
 R - replica had device read errors
 B - tagged data associated with the replica is not valid
```

# Shared Disk Set Mediators

- Mediators are a way around the second rule of shared disk-set replica quorum mathematics.
- Mediators let nodes count as extra votes if there are exactly 50% remaining.
- You can still do SVM disk set failovers if there are exactly 50% of the `metadb` replicas remaining.



When you have nodes connected to exactly two storage arrays, the implication of the replica mathematics described in the previous section is that if one storage array fails, your disk set can keep operating but cannot transfer primary control from node to node.

This is unacceptable in the Oracle Solaris Cluster environment, because it can take a while to fix a broken array or controller, and you still want to be able to gracefully survive a node failure during that period.

The Oracle Solaris Cluster software environment includes special Solaris Volume Manager add-ons called *mediators*. Mediators enable you to identify the nodes themselves as tie-breaking votes in the case of failure of exactly 50% of the `metadb` replicas of a shared disk set. The mediator data is stored in the memory of a running Oracle Solaris OS process on each node. If you lose an array, the node mediators will change to golden status, indicating that they count as extra votes for the shared disk-set quorum mathematics (one from each node). This enables you to maintain normal disk-set operations with exactly 50% of the `metadb` replicas surviving. You can also lose a node at this point (you would still have at least one golden mediator).

Currently, there is added support for a third mediator host in Oracle Solaris Cluster.

# Creating Shared Disk Sets and Mediators

Add hosts, mediators, and disks (use DIDs).

```
# cldev list -v c0t600144F0B3CAAC8300004EF9E2520002d0
DID Device          Full Device Path
----------          ----------------
d1                  clnode1:/dev/rdsk/c0t600144F0B3CAAC8300004EF9E2520002d0

# cldev list -v d6 d7
DID Device          Full Device Path
----------          ----------------
d6                  clnode1:/dev/rdsk/c0t600144F0B3CAAC8300004EF9E2AB0007d0
d6                  clnode2:/dev/rdsk/c0t600144F0B3CAAC8300004EF9E2AB0007d0
d7                  clnode1:/dev/rdsk/c0t600144F0B3CAAC8300004EF9E6300008d0
d7                  clnode2:/dev/rdsk/c0t600144F0B3CAAC8300004EF9E6300008d0

# metaset -s nfsds -a -h clnode1 clnode2
# metaset -s nfsds -a -m clnode1 clnode2
# metaset -s nfsds -a /dev/did/rdsk/d6 /dev/did/rdsk/d7
```



Use the `metaset` command to create new empty disk sets and to add disk drives into the disk set. You must use the `-a -h` options of the metaset command first to create an empty disk set. Then you can add disks. The first host listed as you create a new disk set is the first one to be the owner of the disk set. You can add mediators with the `-a -m` options.

All the examples of disk-set operations shown in this lesson use the DID names for disks instead of the `c#t#d#` names. You might need to run the `cldev list` command often to map between the two.

# Creating Shared Disk Sets and Mediators

```
# metaset
Set name = nfsds, Set number = 1

Host                Owner
  clnode1            Yes
  clnode2

Mediator Host(s)    Aliases
  clnode1
  clnode2

Driv Dbase

d6   Yes
d7   Yes
```

# Creating Shared Disk Sets and Mediators

```
# metadb -s nfsds
 flags           first blk       block count
     a m     luo         16              8192
  /dev/did/dsk/d3s7
     a       luo         16              8192
  /dev/did/dsk/d4s7

# medstat -s nfsds
Mediator                Status  Golden
clnode1                 Ok      No
clnode2                 Ok      No
```



Mediators become *golden* if exactly 50% of the `metadb` replicas fail.

# Quiz

`metadb`s are small partitions containing Solaris Volume Manager configuration and state information, which is separate for local and shared disk sets.

a. True
b. False



**Answer: a**

# Installing Solaris Volume Manager

- Solaris Volume Manager is part of the base Oracle Solaris OS.
- Cluster framework contains `ha-cluster/storage/svm-mediator` (mediator support).

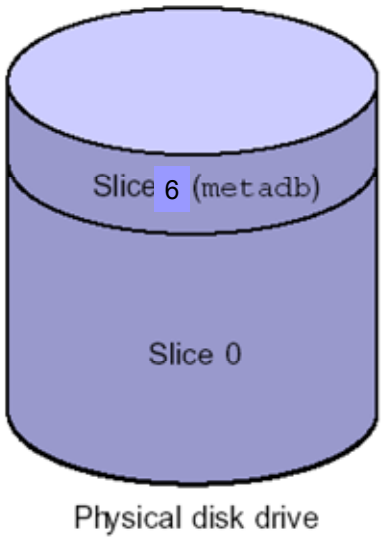


The packages associated with the standard Solaris Volume Manager functionality are part of the base operating system.

Support for shared disk-set mediators is in the `ha-cluster/storage/svm-mediator` package. These packages are automatically installed as part of the cluster framework.

# Automatic Repartitioning and `metadb` Placement on Shared Disk Sets

Uses partition 6 for EFI disks






When a disk is added to a disk set, it is automatically repartitioned as follows:

- A small portion of the drive (starting at cylinder 0) is mapped to slice 7 or slice 6 to be used for state database replicas (at least 4 MB in size).

**Note:** There is full support for disks with Extensible Firmware Interface (EFI) labels, which are required for disks of size greater than 1 terabyte (TB). These disks have no slice 7. Solaris Volume Manager automatically detects an EFI disk and uses slice 6 for the `metadb` partition.

- One `metadb` is added to slice 7 or 6 as appropriate.
- Slice 7 or 6 is marked with the flag bits `01`. (This shows up as `wu` when you view the disk by using the `format` command. `wu` means “writable but unmountable.”)
- The rest of the drive is mapped to slice 0 (on a standard VTOC disk, even slice 2 is deleted).

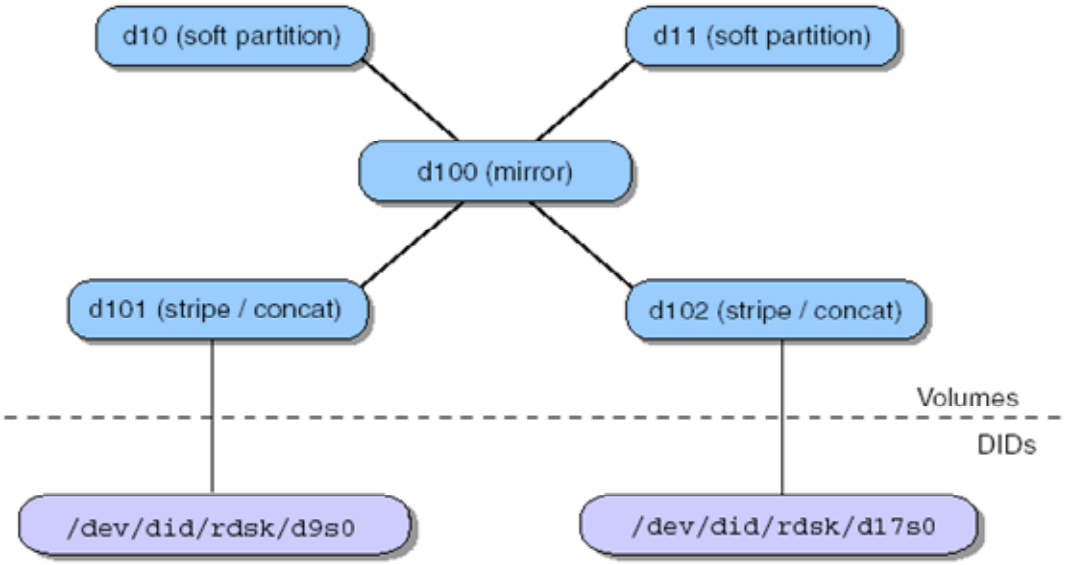
The drive is not repartitioned if the disk already has no slice 2 and if slice 7 already has the following characteristics:

- It starts at cylinder 0.
- It has at least 4 MB (large enough to hold a state database).
- The flag bits are already `01` (`wu` in the `format` command).

Regardless of whether the disk is repartitioned, disk set `metadb` replicas are added automatically to slice 7 or slice 6 as appropriate. If you have exactly two disk controllers, you should always add an equivalent numbers of disks or LUNs from each controller to each disk set to maintain the balance of `metadb` replicas in the disk set across controllers.

**Note:** If it is impossible to add the same number of disks from each controller to a disk set, you should manually run the command `metadb -s setname -b` to balance the number of `metadb` replicas across controllers.

# Using Shared Disk-Set Disk Space




ORACLE



This section shows the commands that implement the strategy described earlier in the lesson.

- Always use slice 0 as in the disk set (almost the entire drive or LUN).
- Build submirrors out of slice `0` of disks across two different controllers.
- Build a mirror out of those submirrors.
- Use soft partitioning of the large mirrors to size the volumes according to your needs.

The diagram in the slide demonstrates the strategy again in terms of volume (`d#`) and DID (`d#s#`). Although Solaris Volume Manager allows you to use `c#t#d#`, always use DID numbers in the cluster to guarantee a unique, agreed-upon device name from the point of view of all nodes.

# Agenda

- Provide an overview of Solaris Volume Manager
- Describe shared disk sets
- Describe Solaris Volume Manager multiowner disk sets
- Describe creating and managing shared disks
- Build volumes in shared disk sets with soft partitions of mirrors
- Use Solaris Volume Manager status commands
- Perform Oracle Solaris Cluster software-level device group management

# Building Volumes in Shared Disk Sets with Soft Partitions of Mirrors

```
# metainit -s nfsds d101 1 1 /dev/did/rdsk/d9s0
nfsds/d101: Concat/Stripe is setup
# metainit -s nfsds d102 1 1 /dev/did/rdsk/d17s0
nfsds/d102: Concat/Stripe is setup
# metainit -s nfsds d100 -m d101
nfsds/d100: Mirror is setup
# metattach -s nfsds d100 d102
nfsds/d100: submirror nfsds/d102 is attached
# metainit -s nfsds d10 -p d100 200m
d10: Soft Partition is setup
# metainit -s nfsds d11 -p d100 200m
d11: Soft Partition is setup
# metattach -s nfsds d10  400m
nfsds/d10: Soft Partition has been grown
# metattach -s nfsds d11  400m
nfsds/d11: Soft Partition has been grown
```



The following are two ways to indicate which disk set you are referring to for each `metainit` command:

- Use `-s disksetname` with the command.
- Use *`disksetname`*`/d#` for volume operands in the command.

This lesson uses the former model for all examples.

The example shown in the slide lists the commands that are used to build the configuration described earlier. The example shows how concat/stripe, mirror, soft partition, and so on are set up.

The following commands show how a soft partition can be grown if there is space (even noncontiguous space) in the parent volume. You will see in the output of `metastat` on the page after next how both soft partitions contain noncontiguous space, because of the order in which they are created and grown.

```
# metattach -s nfsds d10  400m
nfsds/d10: Soft Partition has been grown
# metattach -s nfsds d11  400m
nfsds/d11: Soft Partition has been grown
```

**Note:** The volumes are being increased by 400 MB to a total size of 600 MB.

# Agenda

- Provide an overview of Solaris Volume Manager
- Describe shared disk sets
- Describe Solaris Volume Manager multiowner disk sets
- Describe creating and managing shared disks
- Build volumes in shared disk sets with soft partitions of mirrors
- Use Solaris Volume Manager status commands
- Perform Oracle Solaris Cluster software-level device group management

# Using Solaris Volume Manager Status Commands

```
# metastat -s nfsds
nfsds/d100: Soft Partition
    Device: nfsds/d99
    State: Okay
    Size: 1024000 blocks (500 MB)
        Extent              Start Block              Block count
             0                     1024                  1024000

nfsds/d99: Mirror
    Submirror 0: nfsds/d0
      State: Okay
    Submirror 1: nfsds/d1
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 4173824 blocks (2.0 GB)

nfsds/d0: Submirror of nfsds/d99
    State: Okay
    Size: 4173824 blocks (2.0 GB)
    Stripe 0:
        Device    Start Block  Dbase        State Reloc Hot Spare
        d3s0            0     No            Okay   Yes
```



These commands are applicable to shared disk sets as well. The `metastat` command is the only command that is used.

No commands display information about volumes or `metadb` replicas in multiple disk sets at the same time. If you do not specify a disk set name (with `-s`), you get output only about the local disk set on the node on which you entered the command.

**Checking Volume Status**

The `metastat` command output shown in the slide (and on the following page) is for the mirrored volume and soft partitions built in the previous example.

```
# metastat -s nfsds
nfsds/d11: Soft Partition
    Device: nfsds/d100
    State: Okay
    Size: 1228800 blocks (600 MB)
        Extent        Start Block              Block count
             0             409664                   409600
             1            1638528                   819200
```

```
nfsds/d100: Mirror
    Submirror 0: nfsds/d101
      State: Okay
    Submirror 1: nfsds/d102
      State: Resyncing
    Resync in progress: 0 % done
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 71118513 blocks (33 GB)

nfsds/d101: Submirror of nfsds/d100
    State: Okay
    Size: 71118513 blocks (33 GB)
    Stripe 0:
        Device    Start Block  Dbase        State Reloc Hot Spare
        d9s0            0     No            Okay   No


nfsds/d102: Submirror of nfsds/d100
    State: Resyncing
    Size: 71118513 blocks (33 GB)
    Stripe 0:
        Device    Start Block  Dbase        State Reloc Hot Spare
        d17s0           0     No            Okay   No


nfsds/d10: Soft Partition
    Device: nfsds/d100
    State: Okay
    Size: 1228800 blocks (600 MB)
        Extent              Start Block              Block count
             0                        32                   409600
             1                    819296                   819200

Device Relocation Information:
Device   Reloc  Device ID
d17   No        -
d9    No        -
```

# Agenda

- Provide an overview of Solaris Volume Manager
- Describe shared disk sets
- Describe Solaris Volume Manager multiowner disk sets
- Describe creating and managing shared disks
- Build volumes in shared disk sets with soft partitions of mirrors
- Use Solaris Volume Manager status commands
- Perform Oracle Solaris Cluster software-level device group management

# Managing Solaris Volume Manager Disk Sets and Oracle Solaris Cluster Device Groups

Disk sets are automatically registered as device groups.

```
# cldg status
=== Cluster Device Groups ===

--- Device Group Status ---

Device Group Name     Primary     Secondary     Status
-----------------     -------     ---------     ------
nfsds                 clnode1     clnode2       Online
webds                 clnode1     clnode2       Online
# cldg show nfsds
=== Device Groups ===

Device Group Name:                              nfsds
  Type:                                            SVM
  failback:                                        false
  Node List:                                       clnode1, clnode2
  preferenced:                                     true
  numsecondaries:                                  1
  diskset name:                                    nfsds
```



Configuration changes can be done only by using `meta` commands. The `cldg` command is used to change the current ownership of the disk set and to change certain properties, such as `failback`.

When Solaris Volume Manager is used in the cluster environment, the commands are tightly integrated into the cluster. Creation of a shared disk set automatically registers that disk set as a cluster-managed device group.

Addition or removal of a directly connected node to the disk set using `metaset -s nfsds -a -h` *`newnode`* automatically updates the cluster to add or remove the node from its list. There is no need to use `cluster` commands to resynchronize the device group if volumes are added and deleted.

# Managing Solaris Volume Manager Disk Sets and Oracle Solaris Cluster Device Groups

To switch a device group, you can use:

```
# cldg switch -n clnode2 nfsds
Jan 3 14:06:03 clnode1 Cluster.Framework: stderr: metaset: clnode1: Device
      busy

[a mirror was still synching, will restart on other node]
# cldg status
=== Cluster Device Groups ===

--- Device Group Status ---

Device Group Name     Primary     Secondary     Status
-----------------     -------     ---------     ------
nfsds                 clnode1     clnode2       Online
webds                 clnode1     clnode2       Online
```



In the Oracle Solaris Cluster environment, use only the `cldg switch` command (rather than the `metaset -[rt]` commands) to change physical ownership of the disk set. As demonstrated in this example, if a mirror is in the middle of synching, this forces a switch anyway and restarts the synch of the mirror all over again.

# Managing Solaris Volume Manager Device Groups

- Changes you can make:

```
# cldg set -p failback=true nfsds
```

- Offline:

```
# cldg offline nfsds
```

- Back online (choose one):

```
– # cldg online nfsds
– # cldg switch -n node_to_switch_to nfsds
```



Solaris Volume Manager is cluster aware. As such, there is no need to register Solaris Volume Manager disk sets with the `cldg create` command. The `cldg set` command can be used to perform cluster-specific changes to Solaris Volume Manager device groups.

**Other Changes to Device Groups**

The properties of existing Solaris Volume Manager device groups can be changed. For example, the `failback` property of a group can be modified with the following command:

```
# cldg set -p failback=true nfsds
```

### Putting a Device Group Offline

You can take a Solaris Volume Manager device group out of service, as far as the cluster is concerned, for emergency repairs.

To put the device group offline, all of the Solaris Volume Manager volumes must be unused (unmounted, or otherwise not open). Then you can issue the following command:

```
# cldg offline nfsds
```

You rarely need to take the device group offline because almost all repairs can be done while the device group is in service.

To place the device group back online, type one of the following commands (you do not need both):

- `# cldg online nfsds`
- `# cldg online -n` *`node_to_switch_to`* `nfsds`

# Quiz

The first step when using Solaris Volume Manager in a cluster environment is to register Solaris Volume Manager disk sets with the `cldg create` command.

a. True
b. False



**Answer: b**

# Summary

In this lesson, you should have learned how to:

- Provide an overview of Solaris Volume Manager
- Describe shared disk sets
- Describe Solaris Volume Manager multiowner disk sets
- Describe creating and managing shared disks
- Build volumes in shared disk sets with soft partitions of mirrors
- Use Solaris Volume Manager status commands
- Perform Oracle Solaris Cluster software-level device group management

# Practice 8 Overview: Configuring Volume Management by Using Solaris Volume Manager

This practice covers the following topics:

- Task 1: Initializing the SVM local `metadb` replicas
- Task 2: Selecting the SVM demo volume disk drives
- Task 3: Configuring SVM disk sets and volumes for a web server
- Task 4: Creating a cluster file system
- Task 5: Testing cluster file systems
- Task 6: Managing disk sets

# Practice 8 Overview: Configuring Volume Management by Using Solaris Volume Manager

- Task 7: Creating a global `web` file system
- Task 8: Testing cluster file systems
- Task 9: Managing disk device groups
- Task 10: Viewing and managing Solaris Volume Manager device groups by using Oracle Solaris Cluster Manager

9

# Managing the Public Network with IPMP

# Objectives

After completing this lesson, you should be able to:

- Define the purpose of IPMP
- Define the concepts of an IPMP group
- List examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describe the operation of the `in.mpathd` daemon
- List the options available for use with the `ipadm` command that support IPMP
- Configure IPMP using `ipadm` commands
- Perform a forced failover of an adapter in an IPMP group
- Describe the integration of IPMP into the Oracle Solaris Cluster software environment

ORACLE

# Agenda

- Defining the purpose of IPMP
- Defining the concepts of an IPMP group
- Listing examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describing the operation of the `in.mpathd` daemon
- Listing the options available for use with the `ipadm` command that support IPMP
- Configuring IPMP manually with `ipadm` commands
- Performing a forced failover of an adapter in an IPMP group
- Describing the integration of IPMP into the Oracle Solaris Cluster software environment

# IPMP: Introduction

Internet Protocol network multipathing (IPMP):

- Is the standard Oracle Solaris Operating System (OS) feature to configure redundant network adapters on the same segment
- Detects failures and repairs of adapters
- Is required for Oracle Solaris Cluster software public networks
- Is enforced by “application IP” implementations



Internet Protocol network multipathing (IPMP) has been a standard part of the base Oracle Solaris OS since Solaris 8 OS Update 3 (01/01).

IPMP enables you to configure redundant network adapters, on the same server (node) and on the same subnet, as an IPMP failover group.

The IPMP daemon detects failures and repairs of network connectivity for adapters, and provides failover and failback of IP addresses among members of the same group. Existing TCP connections to the IP addresses that fail over as part of an IPMP group are interrupted for short amounts of time without data loss and without being disconnected.

In the Oracle Solaris Cluster software environment, you must use IPMP to manage any public network adapters on which you will be placing the IP addresses associated with applications running in the cluster.

These application IP addresses are implemented by mechanisms known as `LogicalHostname` and `SharedAddress` resources. The configuration of these resources requires that the public network adapters being used are under the control of IPMP.

# Agenda

- Defining the purpose of IPMP
- Defining the concepts of an IPMP group
- Listing examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describing the operation of the `in.mpathd` daemon
- Listing the options available for use with the `ipadm` command that support IPMP
- Configuring IPMP manually with `ipadm` commands
- Performing a forced failover of an adapter in an IPMP group
- Describing the integration of IPMP into the Oracle Solaris Cluster software environment

ORACLE

# Describing General IPMP Concepts

IPMP group requirements:

- An adapter can be a member of only one group.
- All members of a group must be on the same subnet.
- Adapters on different subnets must be in different groups.
- You can have multiple groups on the same subnet.
- You can have a group with only one member (no failover).
- `local-mac-address?=true` is required.
- Each adapter must have a dedicated test address.
  - Oracle Solaris OS has option for no test address (link state testing), but IPMP is still more robust with test addresses.
- IPV4 and IPV6 (if used on same adapters) must have the same group name.
  - You can have test addresses for neither, one, or both.

ORACLE





IPMP allows you to group together network adapters for redundancy. Members of the same IPMP group are identified by a common group name. This can be any alphanumeric name. The group name is meaningful only inside a single Oracle Solaris OS server.

**Defining IPMP Group Requirements**

You must observe the following rules when configuring an IPMP group:

- A network adapter can be a member of only one group.
- When both IPv4 and IPv6 are configured on a physical adapter, the group names are always the same (and thus need not be specified explicitly for IPv6).
- All members of a group must be on the same subnet. The members, if possible, should be connected to physically separate switches on the same subnet.
- Adapters on different subnets must be in different groups.
- You can have multiple groups on the same subnet.
- You can have a group with only one member. However, there is no redundancy and there is no automatic failover.

- Each Ethernet adapter must have a unique MAC address. This is achieved by setting the `local-mac-address?` variable to `true` in the OpenBoot PROM on a SPARC system. This is automatically set correctly by `scinstall`.
- Network adapters in the same group must be of the same type. For example, you cannot combine Ethernet adapters and Asynchronous Transfer Mode (ATM) adapters in the same group.
- When more than one adapter is in an IPMP group, each adapter requires a dedicated test IP address, or test interface. This is an extra static IP for each group member specifically configured for the purposes of testing the health of the adapter using ping traffic.
- The test interface enables test traffic on all the members of the IPMP group. This is the reason that `local-mac-address?=true` is required for IPMP.
- Oracle Solaris 11 OS does not require test addresses, even with multiple adapters in a group. If you configure adapters in Oracle Solaris 11 OS IPMP groups without test addresses, the health of the adapter is determined solely by the link state of the adapter.

**Note:** Using IPMP without test addresses reduces network traffic and reduces the administrative strain of allocating the addresses. However, the testing is less robust. For example, you may experience adapters with a valid link state but broken receive logic. Such an adapter would be properly faulted using test addresses. The remaining examples in this lesson focus on creating IPMP configurations with test addresses.

- If both IPv4 and IPv6 are configured on the same adapters, it is possible to have test addresses for both IPv4 and IPv6, but it is not necessary. If a failure is detected (even if you have only an IPv4 test address), all IPv4 and IPv6 addresses (except the test address) fail over to the other physical adapter.
- If you do choose to have an IPv6 test address, that test address will always be the link-local IPv6 address (the address automatically assigned to the adapter, which can only be used on the local subnet).

# Agenda

- Defining the purpose of IPMP
- Defining the concepts of an IPMP group
- Listing examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describing the operation of the `in.mpathd` daemon
- Listing the options available for use with the `ipadm` command that support IPMP
- Configuring IPMP manually with `ipadm` commands
- Performing a forced failover of an adapter in an IPMP group
- Describing the integration of IPMP into the Oracle Solaris Cluster software environment

# Configuring Standby Adapters in a Group

- The standby adapter can have only the test interface configured manually.
- Additional IPs are added only as result of failover.
- A standby adapter is usually not used for a two-member group in Oracle Solaris Cluster software.
- It a good idea to use a standby adapter if there are more than two members in a group.





In an IPMP group with two or more members, you can configure an adapter as a standby adapter for the group. Standby adapters have the following properties:

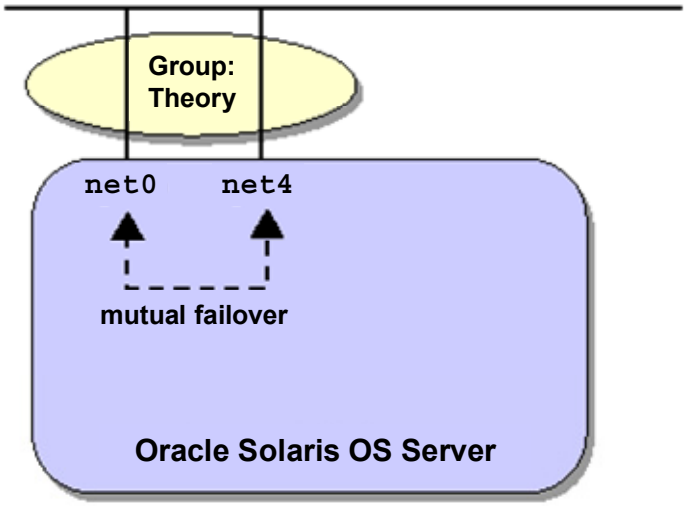
- You are allowed (and are required to) configure only the test interface on the adapter. Any attempt to manually configure any other addresses fails.
- Additional IP addresses are added to the adapter only as a result of a failure of another member of the group.
- The standby adapter is preferred as a failover target if another member of the group fails.
- You must have at least one member of the group that is not a standby adapter.

**Note:** The examples of two-member IPMP groups in the Oracle Solaris Cluster software environment do not use any standby adapters. This allows the Oracle Solaris Cluster software to balance additional IP addresses associated with applications across both members of the group.

If you had a three-member IPMP group in the Oracle Solaris Cluster software environment, setting up one of the members as a standby would still be a valid option.

# IPMP Group Example

Single IPMP group with two members





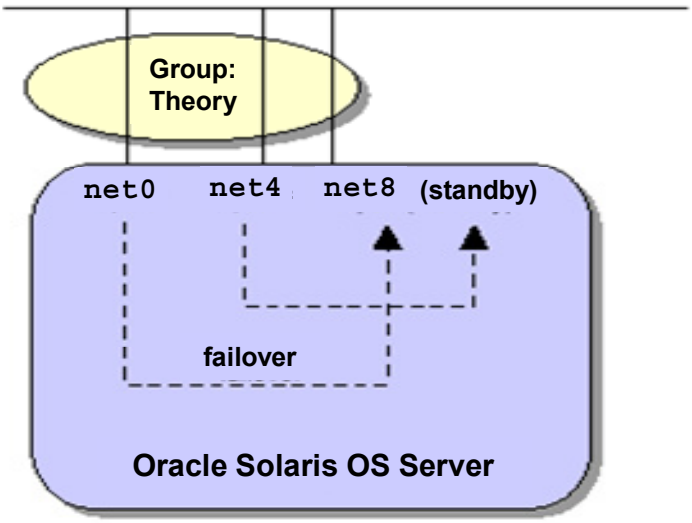

The following sections describe and illustrate examples of network adapters in IPMP groups on a single Oracle Solaris OS server.

**Single IPMP Group with Two Members and No Standby**

The figure in the slide shows a server with two member adapters in a single IPMP group. These two adapters must be on the same subnet and provide failover for each other.

# IPMP Group Example

Single IPMP group with three members including a standby

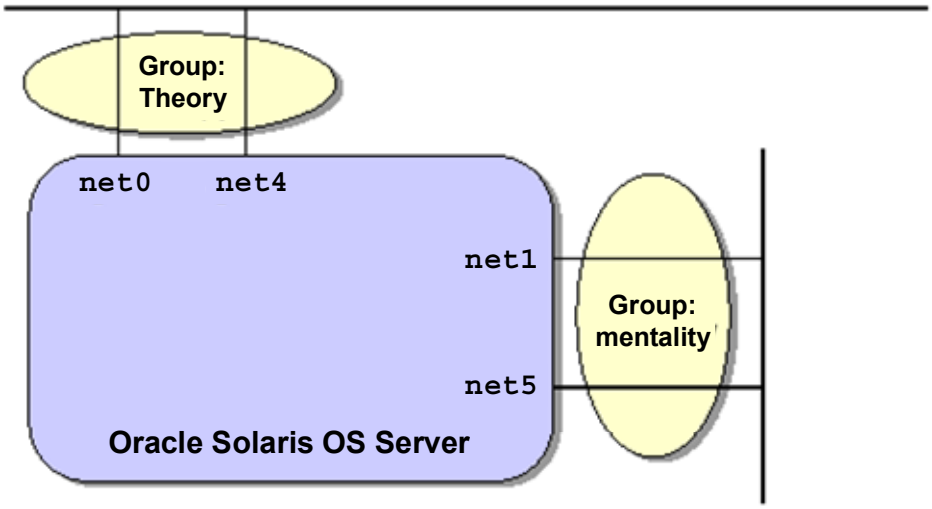





The figure in the slide shows a server with three members of a single IPMP group. In the example, one of the adapters in the group is configured as a standby.

# IPMP Group Example

Two IPMP groups on different subnets

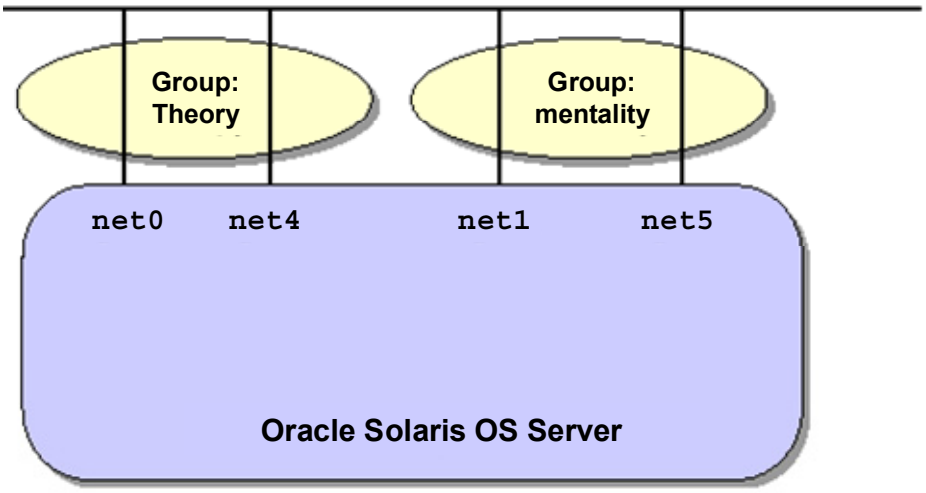





The figure in the slide shows how different IPMP groups must be used for adapters on different subnets.

# IPMP Group Example

Two IPMP groups on the same subnet






The figure in the slide shows two different IPMP groups configured on the same subnet. Failover still occurs only within each particular group.

# Agenda

- Defining the purpose of IPMP
- Defining the concepts of an IPMP group
- Listing examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describing the operation of the `in.mpathd` daemon
- Listing the options available for use with the `ipadm` command that support IPMP
- Configuring IPMP manually with `ipadm` commands
- Performing a forced failover of an adapter in an IPMP group
- Describing the integration of IPMP into the Oracle Solaris Cluster software environment

# Describing the `in.mpathd` Daemon

- The `in.mpathd` daemon controls IPMP.
- When used with test addresses, the daemon does the following:
  - Failure detection: Pings on test interfaces for each adapter
  - Router (host, network, default) is preferred target: Selects five targets if there are no routers
  - Adds host routes to influence selection route
    Example: `route add -host 192.168.1.5 192.168.1.5 -static`
- Failover: All failover IPs are failed over to another member.
- Failback: `in.mpathd` keeps testing to detect the repair.

ORACLE





The `in.mpathd` daemon controls the behavior of IPMP. This behavior can be summarized as a three-part scheme:

- Network path failure detection
- Network path failover
- Network path failback

The `in.mpathd` daemon starts automatically when an adapter is made a member of an IPMP group.

**Network Path Failure Detection**

The following paragraphs describe the functionality of the `in.mpathd` daemon in a configuration using test addresses. Without test addresses, adapter failure detection and repair are based solely on the link state of the adapter.

When test addresses are used, the `in.mpathd` daemon sends Internet Control Message Protocol (ICMP) echo probes (pings) to the targets connected to the link on all adapters that belong to a group to detect failures and repair. The test address is used as the source address of these pings.

**Note:** Using test addresses in Oracle Solaris 11 is more robust. You can have broken adapters, ports, or cables and still have a valid link state.

The `in.mpathd` daemon automatically chooses targets in the following order:

1. Targets are chosen from the routing table in memory in the following order:
   a. Host
   b. Network
   c. Default
2. If no targets are discovered through the routing table, the targets are discovered by a ping command to the 224.0.0.1 (all-hosts) multicast IP address.

To ensure that each adapter in the group functions properly, the `in.mpathd` daemon probes all the targets separately through all the adapters in the multipathing group, using each adapter's test address. If there are no replies to five consecutive probes, the `in.mpathd` daemon considers the adapter as having failed. The probing rate depends on the failure detection time (FDT). The default value for failure detection time is 10 seconds. For a failure detection time of 10 seconds, the probing rate is approximately one probe every two seconds.

You might need to manipulate the choice of ping targets. For example, you might have default routers that are explicitly configured not to answer pings.

One strategy is to enter specific static host routes to IP addresses that are on the same subnet as your adapters. These are then chosen first as the targets. For example, if you wanted to use 192.168.1.39 and 192.168.1.5 as the targets, you could run these commands:

```
# route add -host 192.168.1.39 192.168.1.39 -static
# route add -host 192.168.1.5 192.168.1.5 -static
```

You can put these commands in a boot script.

**Network Path Failover**

After a failure is detected, failover of all network access occurs from the failed adapter to another functional adapter in the group. If you have configured a standby adapter, the `in.mpathd` daemon chooses that for failover of IP addresses and multicast memberships. If you have not configured a standby adapter, `in.mpathd` chooses the adapter with the smallest number of IP addresses. The new adapter accepts all of the IP addresses of the failed adapter, except for the test address.

**Network Path Failback**

The `in.mpathd` daemon detects whether the failed path has been repaired. At this time, the IP addresses that were moved during the failover are returned to their original path, assuming that failback is enabled.

# Agenda

- Defining the purpose of IPMP
- Defining the concepts of an IPMP group
- Listing examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describing the operation of the `in.mpathd` daemon
- Listing the options available for use with the `ipadm` command that support IPMP
- Configuring IPMP manually with `ipadm` commands
- Performing a forced failover of an adapter in an IPMP group
- Describing the integration of IPMP into the Oracle Solaris Cluster software environment

# Configuring IPMP

| ifconfig | ipadm |
|---|---|
| group groupname | Equivalent does not exist. |
| -failover | Equivalent does not exist. |
| deprecated | ipadm set-addprop<br>ipadm show-addprop |
| standby | Equivalent does not exist. |
| addif | ipadm create-addr -T static<br>ipadm create-addr -T dhcp<br>ipadm create-addr -T addrconf |
| removeif | ipadm delete-addr |



IPMP can now be configured using the `ipadm` command instead of using options available for use with the `ifconfig` command. The `ipadm` command is introduced to eventually replace the `ifconfig` command for interface configuration. `ipadm` is an upgrade over `ifconfig`. Unlike `ifconfig`, changes made with `ipadm` persist across reboots.

**Examining `ipadm` Equivalents for `ifconfig` Options for IPMP**

Options that have been added for use with IPMP are:

- `group groupname`: Adapters on the same subnet are placed in a failover group by configuring them with the same group name. The group name can be any name and must be unique only within a single Oracle Solaris OS (node). It is irrelevant whether you have the same or different group names across the different nodes of a cluster. At the moment, there is no equivalent `ipadm` command.
- `-failover`: Use this option to demarcate the test address for each adapter. The test interface is the only interface on an adapter that does not fail over to another adapter when a failure occurs. At the moment, there is no equivalent `ipadm` command.
- `deprecated`: This option, though not required, is typically used on the test interfaces. Any IP address marked with this option is not used as the source IP address for any client connections initiated from this machine.

- `standby`: This option, when used with the physical adapter (`-failover deprecated standby`), turns that adapter into a standby-only adapter. No other virtual interfaces can be configured on that adapter until a failure occurs on another adapter within the group. At the moment, there is no equivalent `ipadm` command.
- `addif`: Use this option to create the next available virtual interface for the specified adapter. The maximum number of virtual interfaces per physical interface is 8192.
- `removeif`**:** Use this option to remove a virtual interface by just giving the IP address assigned to the interface. You do not need to know the virtual interface number.

# Putting Test Addresses on Physical or Virtual Interfaces

- When given the choice, place the test interface on a virtual interface.
- This is because of an obscure bug for physical adapters with deprecated flags.



IPMP was designed so that the test address for each member of an IPMP group can be either on the physical interface (for example, `qfe0`) or on a virtual interface (for example, `qfe0:1`, created with the `ipadm create-addr -T` command).

There were at one time many existing IPMP documents written with the test IP on the physical interface because it looks cleanest to have only the virtual interfaces failing over to other adapters and the IP on the physical interface always staying where it is.

The convention changed due to a bug relating to having a deprecated flag on the physical adapter. Refer to bug #4710499 for more information.

**Note:** The bug concerns failure of certain RPC applications, specifically when the deprecated flag is on the physical adapter. You should always use the deprecated flag with test addresses, and, therefore, you should ensure that the test addresses are not on the physical adapter. The examples in this lesson follow the convention of placing the test interface on a virtual interface (when you have that choice).

# Quiz

Which of the following three functions does the `in.mpathd` daemon control when managing IPMP?

a. Network path failure detection
b. Network path failover
c. Network path failback



**Answer: a, b, c**

# Agenda

- Defining the purpose of IPMP
- Defining the concepts of an IPMP group
- Listing examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describing the operation of the `in.mpathd` daemon
- Listing the options available for use with the `ipadm` command that support IPMP
- Configuring IPMP manually with `ipadm` commands
- Performing a forced failover of an adapter in an IPMP group
- Describing the integration of IPMP into the Oracle Solaris Cluster software environment

# Using the `ipadm` Command to Configure IPMP

- You can configure an interface using:

```
# ipadm create-addr -T static -a 10.0.2.15/24 e1000g0/v4static
```

- Note that there are two flags (`-T` and `-a`) in this example.
- The IP address is now being assigned the name `e1000g1/v4static`.



You can configure a static IP address by using the following command syntax:

```
# ipadm create-addr -T static -a 10.0.2.15/24 net1/v4static
```

There are additional tags such as `-T` and `-a` when you use the `ipadm` command. In addition, a name is assigned.

# Using the `ipadm` Command to Configure IPMP

Use the following command to view IP addresses that are configured on the network interfaces:

```
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
sc_ipmp0/static1  static   ok           192.168.1.101/24
sc_ipmp0/v4add2   static   ok           192.168.1.150/24
net0/test         static   ok           192.168.1.151/24
net1/?            static   ok           192.16.0.17/29
net2/test         static   ok           192.168.1.152/24
net3/?            static   ok           192.16.0.9/29
clprivnet0/?      static   ok           192.16.0.65/27
lo0/v6            static   ok           ::1/128
net0/_a           static   ok           fe80::a00:27ff:fe80:4c01/10
net2/_a           static   ok           fe80::a00:27ff:fe74:f5a1/10
```

# in.mpathd Configuration File

```
#
# Time taken by mpathd to detect a NIC failure in ms.
# The minimum time that can be specified is 100 ms.
#
FAILURE_DETECTION_TIME=10000
#
# Failback is enabled by default. To disable failback
# turn off this option
#
FAILBACK=yes
#
# By default only interfaces configured as part of
# multipathing groups are tracked. Turn off this
# option to track all network interfaces on the system
#
TRACK_INTERFACES_ONLY_WITH_GROUPS=yes
```



The `in.mpathd` daemon uses the settings in the `/etc/default/mpathd` configuration file to invoke multipathing. Changes to this file are read by the `in.mpathd` daemon at startup and on a SIGHUP. This file contains the default settings and information (as shown in the slide).

**Note:** In general, you do not need to edit the default `/etc/default/mpathd` configuration file.

The three settings you can change in the `/etc/default/mpathd` file are:

- `FAILURE_DETECTION_TIME`: You can adjust the value of this parameter. If the load on the network is too great, the system cannot meet the failure detection time value. Then the `in.mpathd` daemon prints a message on the console, indicating that the time cannot be met. It also prints the time that it can meet currently. If the response comes back correctly, the `in.mpathd` daemon meets the failure detection time provided in this file.
- `FAILBACK`: After a failover, failbacks take place when the failed adapter is repaired. However, the `in.mpathd` daemon does not fail back the addresses if the `FAILBACK` option is set to `no`.
- `TRACK_INTERFACES_ONLY_WITH_GROUPS`: In stand-alone servers, you can set this to `no` so that `in.mpathd` monitors traffic even on adapters not in groups. In the cluster environment, make sure that you leave the value as `yes` so that private transport adapters are not probed.

# Quiz

Is the following statement true or false?

If you want an IPV6 test address, create empty file names for the IPV6 interfaces.

a. True
b. False



**Answer: b**

# Agenda

- Defining the purpose of IPMP
- Defining the concepts of an IPMP group
- Listing examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describing the operation of the `in.mpathd` daemon
- Listing the options available for use with the `ipadm` command that support IPMP
- Configuring IPMP manually with `ipadm` commands
- Performing a forced failover of an adapter in an IPMP group
- Describing the integration of IPMP into the Oracle Solaris Cluster software environment

# Performing Failover and Failback Manually

- Failover

```
# if_mpadm -d net1
```

- Failback

```
# if_mpadm -r net1
```



**Failover:** You can do a forced failover of an adapter in an IPMP group, as shown in the slide. The command causes IPMP to behave as if `net1` had failed. All IP addresses, except the test interface, are failed over to another member of the group. The adapter is marked down and is not accessed by `in.mpathd` until re-enabled.

**Failback:** You can re-enable an adapter after this operation by using the command shown in the slide. This command allows the adapter to take back the IP addresses to which it was originally assigned, assuming that `FAILBACK` is set to `yes` in the `/etc/default/mpathd` file.

# Agenda

- Defining the purpose of IPMP
- Defining the concepts of an IPMP group
- Listing examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describing the operation of the `in.mpathd` daemon
- Listing the options available for use with the `ipadm` command that support IPMP
- Configuring IPMP manually with `ipadm` commands
- Performing a forced failover of an adapter in an IPMP group
- Describing the integration of IPMP into the Oracle Solaris Cluster software environment

ORACLE

# Configuring IPMP in the Oracle Solaris Cluster Environment

- Configure IPMP before or after cluster installation.
- Using the same group names for the same subnet on different nodes might be a good convention.
- Do not use a standby adapter if a group has only two members.
- Ensure `FAILBACK=yes`.



There are no special cluster-related tools for configuring IPMP in the Oracle Solaris Cluster environment. IPMP is configured on each node exactly as in a non-clustered environment.

**Using the Same Group Names on Different Nodes**

It makes no difference to IPMP whether you use the same group names or different group names for IPMP across nodes. IPMP itself is aware only of what is going on within the local Solaris OS.

It is a helpful convention to use the same group names on all nodes that are connected to the same subnet.

**Understanding Standby and Failback**

It is unlikely that you want to use a standby adapter in a two-member IPMP group in the cluster environment. If you do, the Oracle Solaris Cluster software will be unable to load-balance additional application-related IP addresses across the members of the group.

Retain the setting `FAILBACK=yes` in the `/etc/default/mpathd` file. Oracle Solaris Cluster software is automatically trying to load-balance additional IP addresses across the members of the group; when a repair is detected, they are rebalanced.

# Integrating IPMP into the Oracle Solaris Cluster Software Environment

The `pnmd` daemon:

- Is a wrapper around IPMP
- Stores IPMP status in CCR
- Facilitates application failover if one node has total network outage
- Consults the `/var/cluster/run/pnm_callbacks` file

```
# cat /var/cluster/run/pnm_callbacks
sc_ipmp0 orangecat-nfs.mon /usr/cluster/lib/rgm/rt/hafoip/
hafoip_ipmp_callback mon nfs-rg orangecat-nfs
```



There is nothing special about IPMP itself within the cluster. In the cluster environment, as in a non-cluster environment, `in.mpathd` is concerned with probing network adapters only on a single Solaris OS, and manages failovers and failbacks between the adapters in a group.

The Oracle Solaris Cluster software environment, however, needs additional capability wrapped around IPMP to do the following:

- Make the status of IPMP groups on each node available throughout the cluster.
- Facilitate application failover in the case where all members of an IPMP group on one node have failed, but the corresponding group on the same subnet on another node has a healthy adapter.

Clearly, IPMP itself is unaware of any of these cluster requirements. Instead, the Oracle Solaris Cluster software uses a cluster-specific public network management daemon (`pnmd`) to perform this cluster integration.

### Capabilities of the `pnmd` Daemon in Oracle Solaris Cluster Software

In the cluster environment, the `pnmd` daemon has the following capabilities:

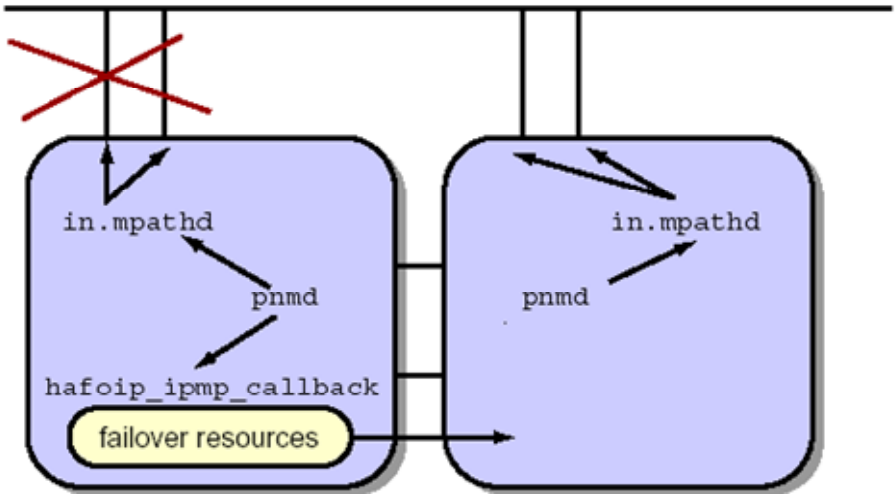
- Communicates with `pnmd` on other nodes to transmit the status of IPMP groups
- Facilitates application failover

When `pnmd` detects that all members of a local IPMP group have failed, it consults a file named `/var/cluster/run/pnm_callbacks`. This file contains entries that would have been created by the activation of `LogicalHostname` and `SharedAddress` resources.

It is the job of `hafoip_ipmp_callback` (in the example in the slide) to decide whether to migrate resources to another node.

# Integrating IPMP into the Oracle Solaris Cluster Software Environment

## Summary of IPMP cluster integration




ORACLE



The figure in the slide summarizes the IPMP and `pnmd` elements of public network management in the cluster.

# Integrating IPMP into the Oracle Solaris Cluster Software Environment

Viewing cluster-wide IPMP information with the `clnode status -m` command:

```
# clnode status -m

--- Node IPMP Group Status ---

Node Name     Group Name     Status     Adapter     Status
---------     ----------     ------     -------     ------
clnode1       sc_ipmp0       Online     net2        Online
clnode1       sc_ipmp0       Online     net0        Online
clnode2       sc_ipmp0       Online     net2        Online
clnode2       sc_ipmp0       Online     net0        Online
```



In the Oracle Solaris Cluster software environment, running the `clnode status -m` command from any node shows the status of IPMP group members on all the nodes.

# Summary

In this lesson, you should have learned how to:

- Define the purpose of IPMP
- Define the concepts of an IPMP group
- List examples of network adapters in IPMP groups on a single Oracle Solaris OS server
- Describe the operation of the `in.mpathd` daemon
- List the options available for use with the `ipadm` command that support IPMP
- Configure IPMP manually with `ipadm` commands
- Perform a forced failover of an adapter in an IPMP group
- Describe the integration of IPMP into the Oracle Solaris Cluster software environment

# Practice 9 Overview: Configuring and Testing IPMP

This practice covers the following topics:

- Task 1: Verifying the `local-mac-address?` variable
- Task 2: Verifying the adapters for the IPMP group
- Task 3: Verifying or entering test addresses in the `/etc/inet/hosts` file
- Task 4: Configuring IPMP with public network adapters and test addresses
- Task 5: Verifying that IPMP is configured
- Task 6: Verifying IPMP failover and failback

ORACLE

10

# Managing Data Services, Resource Groups, and HA-NFS

# Objectives

After completing this lesson, you should be able to:

- Describe how data service agents enable a data service in a cluster to operate properly
- List the components of a data service agent
- Describe data service packaging, installation, and registration
- Describe the primary purpose of resource groups
- Differentiate between failover and scalable data services
- Describe how to use special resource types
- List the guidelines for using global and highly available local file systems
- Differentiate between standard, extension, and resource group properties
- Register resource types
- Configure resource groups and resources
- Control switching of resources and resource groups manually

ORACLE

# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

ORACLE

# Data Services in the Cluster

- Highly available and scalable applications
- Minimal down time for applications
- Applications run across nodes or on branded zones.
- Off-the-shelf applications
  - Where to put binaries? Local copies or shared storage?





The Oracle Solaris Cluster software framework makes applications highly available (HA), minimizing application interruptions after any single failure in the cluster.

In addition, some applications, such as Apache web server software, are supported not only with HA features, but also in a scalable configuration. This configuration allows the service to run on multiple nodes of the cluster simultaneously while providing a single IP address to the client.

**`solaris10` Branded Zones**

HA zones can host clustered failover and scalable applications. Note that `solaris10` zones must be configured with HA zones; they cannot be used without it. HA zones also support `solaris` brand zones.

### Off-the-Shelf Application

For most applications supported in the Oracle Solaris Cluster software environment, software customization is not required to enable the application to run correctly in the cluster. An Oracle Solaris Cluster software agent is provided to enable the data service to run correctly in the cluster environment.

### Application Requirements

You should identify requirements for all of the data services before you begin the Oracle Solaris OS and Oracle Solaris Cluster software installation. Failure to do so might result in installation errors that require you to completely reinstall the Oracle Solaris OS and Oracle Solaris Cluster software.

### Determining the Location of the Application Binaries

You can install the application software and application configuration files on one of the following locations:

- **Local disks of each cluster node:** Placing the software and configuration files on the individual cluster nodes enables you to upgrade application software later without shutting down the service. The disadvantage is that you have several copies of the software and configuration files to maintain and administer.
- **A cluster file system or highly available local file system using shared storage:** If you put the application binaries on a shared file system, you have only one copy to maintain and manage. However, you must shut down the data service in the entire cluster to upgrade the application software. If you can spare a small amount of down time for upgrades, place a single copy of the application and configuration files on a shared global or highly available local file system.

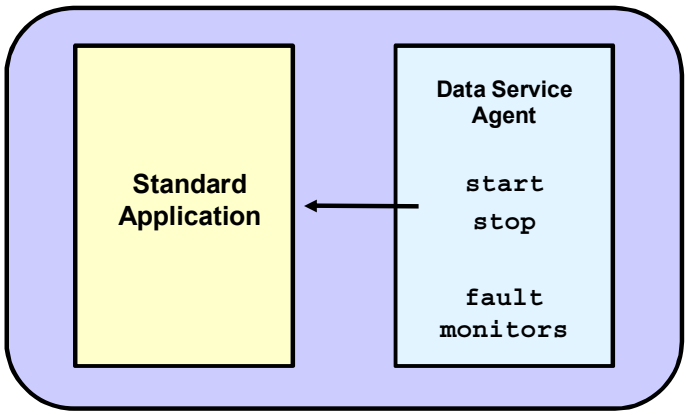
# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

ORACLE

# Oracle Solaris Cluster Software Data Service Agents

- Start and stop “methods”
- Fault monitors
- Start and stop methods for fault monitors






The word *agent* is an informal name for a set of components, written specifically for Oracle Solaris Cluster software, that enable a data service in a cluster to operate properly.

# Components of a Data Service Agent

- Methods to start and stop the service in the cluster
- Fault monitors for the service
- Methods to start and stop the fault monitoring
- Methods to validate configuration of the service in the cluster
- Registration information file (resource type definition)
- Data Service Agent Fault Monitoring
  - Monitor the health of the daemons
  - Monitor the health of the service



Typical components of a data service agent include the following:

- Methods to start and stop the service in the cluster
- Fault monitors for the service
- Methods to start and stop the fault monitoring
- Methods to validate configuration of the service in the cluster
- A registration information file that allows the Oracle Solaris Cluster software to store all the information about the methods into the Cluster Configuration Repository (CCR)
- You only need to reference a resource type to refer to all the components of the agent.

Oracle Solaris Cluster software provides an API that can be called from shell programs, and an API that can be called from C or C++ programs. Most of the program components of data service methods that are supported by Solaris products are actually compiled C and C++ programs.

### Fault Monitor Components

Fault monitoring components that are specific to data services in Oracle Solaris Cluster software are run on the local node only. This is the same node that is running the data service. Fault monitoring components are intended to detect application failure and can suggest either application restarts or failovers in the cluster.

The actual capabilities of these fault monitors are application specific and often poorly documented. The general strategy for fault monitors in the Oracle Solaris Cluster environment is to monitor the health of the following:

- The daemons, by placing them under control of the process monitoring facility (`rpc.pmfd`). This facility calls action scripts if data service daemons stop unexpectedly
- The service, by using client commands

**Note:** Data service fault monitors do not need to monitor the health of the public network itself because this is already done by the combination of `in.mpathd` and `pnmd`.

# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

# Data Service Packaging, Installation, and Registration

- Packages for Oracle Solaris–supported agents are part of Oracle Solaris Cluster installation.
- Some packages come with the software.
- Adding a package gives you a usable “resource type.”
- Package name is not the same as “resource type.”
  - Add the `ha-cluster/data-service/nfs` package, and the `SUNW.nfs` resource type becomes available.
- Add an agent in a global Oracle Solaris zone, and it will be propagated to HA zones.
  - `pkg install` is the command used in Oracle Solaris 11.



Agent packages released along with Oracle Solaris Cluster are part of the Oracle Solaris Cluster installation medium. You can install them using the Oracle Solaris Cluster installation at the same time that you install the Oracle Solaris Cluster framework. You can invoke the Oracle Solaris Cluster installation at a later time to install them, or you can just use the `pkgadd` command.

Some agents are supplied with the application software, rather than with the cluster software.

If you intend to run certain clustered applications only in `solaris10` branded zones, you might install the agents only in these HA zones. Alternatively, you might choose to install the agents always in the global Oracle Solaris zone, without the `-G` option, and have them automatically propagate to all existing and future HA zones.

**Data Service Packages and Resource Types**

Each data service agent encapsulates all the information about the agent as a resource type.

When this resource type is registered with the cluster software, you do not need to know the location or names of the components of the agent. You only need to reference the application's resource type to determine all the correct information about methods and fault monitors for that component.

**Note:** The package that you add by using the `pkgadd` command to install an agent and the corresponding resource type might have different names. For example, when you install the `SUNWschtt` package, a resource type called `SUNW.iws` becomes available.

# Quiz

Which of the following are components of a data service agent?

a. Methods to start and stop the service in the cluster
b. Fault monitors for the service
c. Methods to start and stop the fault monitoring
d. Methods to monitor the public interface
e. Methods to validate configuration of the service in the cluster



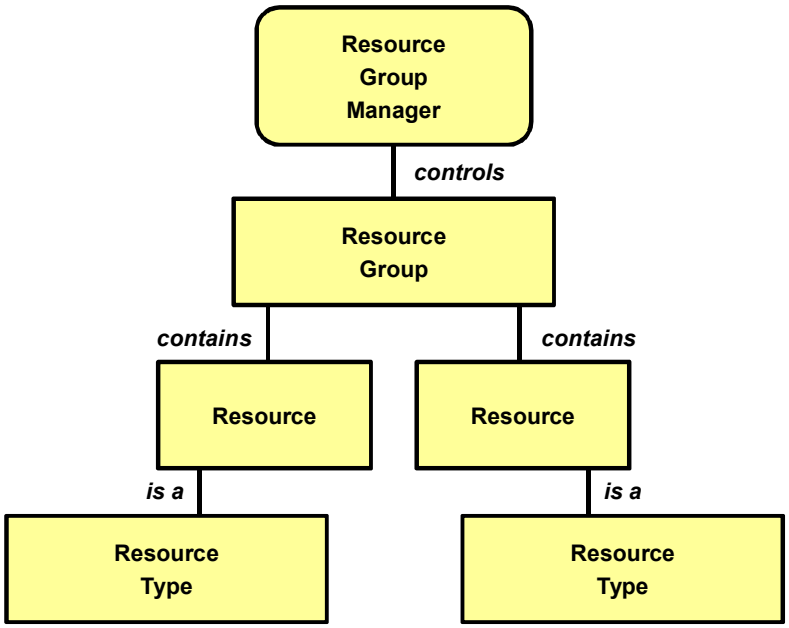
**Answer: a, b, c, e**

# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

# Resources, Resource Groups, and the Resource Group Manager

## Resource Group Manager






Data services are placed under the control of the cluster by configuring the services as resources within resource groups. In the figure in the slide, the `rgmd` daemon is the Resource Group Manager, which controls all activity having to do with resources and resource groups. That is, the `rgmd` daemon controls all data service activity within the cluster.

# Resources

- A resource is a configured instance.
  - Example: An instance of an Apache web server
- A resource has:
  - A type (for example, `SUNW.apache`)
  - A unique name (just used with cluster utilities)
  - A set of properties that define it (and distinguish it from other resources of same type)
- A resource always belongs in a resource group (RG).
  - An RG is a collection of resources (failover or scalable).



In the context of clusters, the word *resource* refers to any element above the layer of the cluster framework that can be turned on or off and can be monitored within the cluster.

An example of a resource would be an instance of a running data service. For example, an Apache web server with a single `httpd.conf` file counts as one resource.

Each resource is an instance of a specific resource type. For example, each Apache web server is an instance of the `SUNW.apache` resource type.

Other types of resources represent IP addresses and data storage that are required by the application service.

A particular resource is identified by the following:

- Its resource type
- A unique name, which is used as identification in Oracle Solaris Cluster utilities
- A set of properties, which are parameters that define the configuration of a particular resource

You can configure multiple resources of the same type in the cluster, either in the same resource group or in different resource groups.

For example, you might want to run two failover Apache web server application services that reside, by default, on different nodes in the cluster, but could fail over to the same node if there were only one node available. In this case, you have two resources, both of type `SUNW.apache`, in two different resource groups.

# Resource Groups

Failover and scalable resource groups

- Failover resource groups
  - An RG is the unit of failover.
  - All resources in the group run on a node together.
  - All fail or switch over to a different node together.
- Scalable groups
  - Services run simultaneously on multiple nodes.
  - Scalable applications specifically use load balancing.



Resource groups are collections of resources. Resource groups are either failover or scalable.

**Failover Resource Groups**

For failover applications in the cluster, the resource group becomes the unit of failover. That is, the resource group is the collection of services that always run together on one node of the cluster at a given time, and simultaneously fail over or are switched over to another node.

**Scalable Resource Groups**

Scalable resource groups describe the collection of services that run simultaneously on one or more nodes or zones.

A scalable application in a scalable resource group also refers to an application making use of the load-balancing services built into Oracle Solaris Cluster software. Applications that are in scalable groups but do not make use of the built-in load-balancing service are referred to as *multimaster* applications.

# Resource Groups

- How many data services are in a resource group?
  - A separate group for each application
  - Or many applications in the same group (fail over together)
- Advantage of a separate group for each application
  - More flexibility (separate failover)
- Advantage of having all applications in a single group
  - Knowing that if they all run on one node, they will probably run OK (resource-wise) on another node.





**Number of Data Services in a Resource Group**

Multiple data services can be configured to run in the same resource group.

For example, you can run two web server application services, a directory server software identity management service, and a database as four separate resources in the same failover resource group. These applications always run on the same node at the same time, and always simultaneously migrate to another node. Or you can put all four of these services in separate resource groups. The services could still run on the same node, but could fail over and switch over independently.

**Benefits of Putting All Services in a Single Resource Group**

If you put all data services in the same failover resource group of a two-node cluster, the node that is not currently hosting the resource group is in a pure backup mode. Some customers prefer to deploy services this way, in a single resource group, because this configuration provides more predictable behavior. If the node currently running all the data services is performing optimally, and there is a failover, you can predict that the new node will have the same performance, assuming equivalent servers.

# Resource Group Manager

The Resource Group Manager (RGM) (the `rgmd` daemon):

- Maintains all statuses of resources and resources groups
- Launches methods for resources on the proper nodes



The `rgmd` daemon maintains all the information about data services that are known to the cluster as resources in resource groups. The `rgmd` daemon launches all methods that start resources in resource groups and performs all failovers.

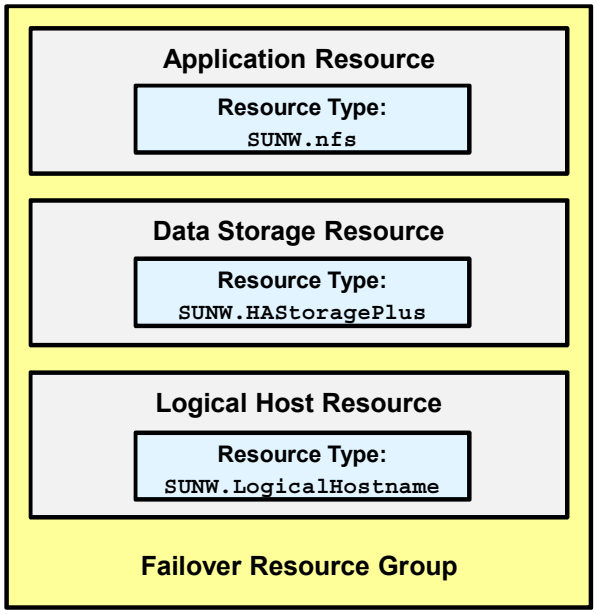
# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

ORACLE

# Describing Failover Resource Groups

- RG and resource names must both be globally unique.
- There are RG properties for the whole RG.




ORACLE



A resource name must be globally unique, not merely unique within a particular resource group.

This failover resource group is defined with a unique name in the cluster. Configured resources are added to the resource group. When these resources are placed into the same failover resource group, they move together.

# Resources and Resource Types

- Each instance of a data service is a resource.
- The resource type describes the kind of resource.
- Some applications, such as Oracle databases, require multiple resources to form one logical application.
- There are a few special resource types that are not data service types.



Each instance of a data service under cluster control is represented by a resource within a resource group.

Resources exist only inside resource groups. There is no such thing as a disembodied resource that is not a member of a resource group.

Each resource has a resource type that describes the type of resource it is (for example, `SUNW.nfs` for a Network File System [NFS] resource).

At least one defined resource type exists for each supported service in the cluster. Some applications that are typically considered to be a single entity, such as an instance of the Oracle database, actually require two different resources with different types: the Oracle server and the Oracle listener.

In addition to resource types that relate to data services, a few other special resource types relate to IP addresses and storage. These resource types are described later in this lesson.

# Resource Type Versioning

- Type versioning is a framework to allow the co-existence of different resources of old and new types.
- The resource type version is actually official part of resource type names:
  - `SUNW.nfs:3.3`
  - `SUNW.nfs`
  - `SUNW.SharedAddress:2`
  - `SUNW.HAStoragePlus:9`



Oracle Solaris Cluster software gives you the ability to use different versions of resource types. For example, old and new versions of data service agents can co-exist as separate types. Individual resources of an original resource type can be upgraded to a new type on a resource-by-resource basis.

Officially, the version number is appended to the resource type name. For example, the official name of the NFS resource type is `SUNW.nfs:3.3`.

When you use a resource type name, the version suffix can be dropped if there is no ambiguity. For that matter, the vendor prefix can also be dropped. For example, when you initially install Oracle Solaris Cluster software, all of the following names can be used to refer to the `NFS` resource type:

- `SUNW.nfs:3.3`
- `SUNW.nfs`
- `SUNW.SharedAddress:2`
- `SUNW.HAStoragePlus:9`

# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

ORACLE

# Using Special Resource Types

`SUNW.LogicalHostname`

- IP addresses that must run on the same node as the application and fail over with the application

`SUNW.SharedAddress`

- Special type of IP address with load balancing for scalable services





This section describes special resource types that you include in resource groups to complete the configuration.

**`SUNW.LogicalHostname` Resource Type**

Resources of type `SUNW.LogicalHostname` represent one or more IP addresses on a particular subnet that will become the logical IP addresses used to access the services in that resource group.

That is, each IP address described by a `SUNW.LogicalHostname` resource migrates from node to node, along with the services for that group. If the group is mastered by an HA zone rather than by a physical node, the IP address is automatically configured in the appropriate zone. The clients use these IP addresses to access the services in the cluster.

A large part of what makes cluster failover relatively transparent to the client is that IP addresses migrate along with services of the group. The clients always use the same logical IP address to contact a particular instance of a data service, regardless of which physical node or zone is actually running the service.

### `SUNW.SharedAddress` Resource Type

Resources of type `SUNW.SharedAddress` represent a special type of IP address that is required by scalable services. This IP address is configured on the public network on only one node or zone with failover capability, but provides a load-balanced IP address that supports scalable applications that run on multiple nodes or zones simultaneously.

# Using Special Resource Types

`SUNW.HAStoragePlus`

- Manages:
  - Global raw devices
  - Cluster file system
  - Highly available local file system (traditional, non-ZFS)
  - ZFS `zpool`(s), including all file systems within
- Is optional, but it:
  - Checks to see whether the global devices or file systems in question are offline
  - Allows dependency from application resources
  - Allows you to set `AffinityOn=true` (affinity switching with RG)
- Is required to manage failover





### `SUNW.HAStoragePlus` Resource Type

The `SUNW.HAStoragePlus` resource type manages the following types of storage:

- Global raw devices
- Cluster file systems
- Traditional (non-ZFS) highly available local file systems
- ZFS zpools, including all the file systems within

### Global Raw Devices and Cluster File Systems

Global raw devices and cluster file system management, which include failing over from a failed node, are part of the Oracle Solaris Cluster software framework. So it might seem redundant to provide a resource type that also manages these global devices and file systems.

In fact, managing global storage with a `SUNW.HAStoragePlus` resource is optional, but serves the following useful purposes:

- The `START` method of `SUNW.HAStoragePlus` is used to check whether the global devices or file systems in question are accessible from the node where the resource group is going online.
- Usually, you use the `Resource_dependencies` standard property to configure a dependency so that the application resources depend on the `SUNW.HAStoragePlus` resource. In this way, the Resource Group Manager does not try to start services if the storage the services depend on is not available.
- The `SUNW.HAStoragePlus` resource type can also be used to attempt colocation of resource groups and device groups onto the same node, thus enhancing the performance of disk-intensive data services.

### Highly Available Local File Systems

You must configure a `SUNW.HAStoragePlus` to manage a highly available local file system. It is the resource methods that provide the failover of the file system.

A highly available local file system must fail over together as the service fails over, but can fail over only to nodes that are physically connected to the storage.

The configurations of global and highly available local file systems look very similar. Both use the `FilesystemMountPoints` property to identify the file system (or systems) in question. The `SUNW.HAStoragePlus` methods distinguish between global and highly available local file systems by using the `/etc/vfstab` entry.

### Highly Available Local File System Monitoring

The monitoring functionality of the `HAStoragePlus` resource type monitors the health of the file systems, ZFS storage pools, and global devices managed by the `HAStoragePlus`. Unavailability of the managed entities causes the resource to be restarted or failed over to another node. The fault monitor probes the status of all entities managed by the resource. When there is more than one entity to be monitored, the probing is done in parallel.

# Quiz

Which of the following does `SUNW.LogicalHostname` manage?

a. IP addresses that must run on the same node as the application and fail over with the application
b. Global raw devices and cluster file system
c. A special type of IP address with load balancing for scalable services



**Answer: a**

# Quiz

Which of the following does `SUNW.SharedAddress` manage?

a. IP addresses that must run on the same node as the application and fail over with the application

b. Global raw devices and cluster file system

c. A special type of IP address with load balancing for scalable services



**Answer: c**

# Quiz

Which of the following does `SUNW.HAStoragePlus` manage?

a. IP addresses that must run on the same node as the application and fail over with the application
b. Global raw devices and cluster file system
c. A special type of IP address with load balancing for scalable services



**Answer: b**

# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

ORACLE

# Guidelines for Using Cluster and Highly Available Local File Systems

Use a cluster file system for any of the following:

- A scalable service
- A failover service that must fail over to a node that is not physically connected to the storage
- Data within the same file system for different failover services that may be running on different nodes
- Access to data from a node that is not currently running the service



This section provides some general guidelines for choosing between global and highly available local file systems.

**When to Use a Cluster File System**

Use a cluster file system if you want to support any of the following:

- A scalable service
- A failover service that must fail over to a node that is not physically connected to the storage
- A single file system that contains data for different failover services that may be running on different nodes
- Access to the data from a node that is not currently running the service

If your storage is physically connected to all possible nodes for the resource group containing the `HAStoragePlus` resource, you can migrate the underlying ownership of the storage along with the service, as a performance benefit.

# Guidelines for Using Cluster and Highly Available Local File Systems

A highly available local file system provides better performance, assuming all of the following are true:

- The file system is for a failover service only.
- The `Nodelist` for the resource group contains only nodes that are physically connected to the storage. That is, the `Nodelist` for the resource groups is the same as the `Nodelist` for the device group.
- Only the services in a single resource group use the file system.



**When to Use a Highly Available Local File System**

If the conditions listed in the slide are true, a highly available local file system provides a higher level of performance than a cluster file system, especially for services that are file-system intensive.

# Guidelines for Using Cluster and Highly Available Local File Systems

`HAStoragePlus` and ZFS

- For ZFS, an `HAStoragePlus` resource represents one or more zpools.
- Just configure the resource with the `Zpools` property referring to one or more pools.
  - Automatically includes all file systems
  - No `vfstab` entries required (everything is automated by ZFS itself)
- Each `HAStoragePlus` resource represents traditional FS(s) or ZFS zpool(s), but not both.
- When configured in an Oracle Solaris zone, zpools are exposed only in that zone.



The same `HAStoragePlus` resource that provides failover of traditional file systems can also provide ZFS failover. For ZFS, an `HAStoragePlus` instance represents one or more zpools and all the file systems within.

To manage failover of ZFS file systems within the cluster, all you have to do is configure an `HAStoragePlus` instance with the value of the `Zpools` property indicating one or more pools that should fail over. You do not need to configure any `/etc/vfstab` entries whatsoever. All of the ZFS mount information is self-contained in the zpool configuration database.

**Note:** A single `HAStoragePlus` instance can refer to multiple traditional (non-ZFS) file systems, or multiple ZFS zpools, but not both. When you use the `Zpools` property, the values of the properties for traditional file systems (`FilesystemMountPoints` and `AffinityOn`) are ignored.

ZFS technology is aware of Oracle Solaris zones, and ZFS zpools that are configured into an HA zone via `HAStoragePlus` are exposed only in that zone. They are not visible in the global Oracle Solaris zone.

# Guidelines for Using Cluster and Highly Available Local File Systems

Generic Data Service

- The `SUNW.gds` resource type is a single, precompiled data service that requires minimal modifications.
- You just set the following properties differently for each instance of `SUNW.gds`:
  - What application is being launched?
  - How will this application be probed?
  - How will this application be stopped?



The Generic Data Service (GDS) is a resource type for making simple applications highly available or scalable by using the resource type `SUNW.gds` to plug them into the Oracle Solaris Cluster RGM framework. The GDS is a fully functional Oracle Solaris Cluster resource type complete with callback methods and a Resource Type Registration file.

Basically, the concept is that many different applications can share the same resource type. All you have to do is have a resource type in which what is being launched is not implied by the resource type itself. Instead, what is being launched, how to probe this application, and how to stop the application are just properties that can be set differently for each instance of `SUNW.gds`.

Many Solaris-supported data services (for example, DHCP, Samba, and many more) do not actually supply new resource types. Rather, they supply configuration scripts that configure resources to represent the applications as instances of `SUNW.gds`.

In the practice for this lesson, you use an instance of the `SUNW.gds` type to put your own customized application under control of the Oracle Solaris Cluster software.

# Understanding Resource Dependencies and Resource Group Dependencies

- "A depends on B"
  - A (dependent) and B (dependee) are different resources.
  - They can be in the same resource group or in different resource groups.
- Regular resource dependencies:
  - Are strongly enforced when the RGM is starting resources
  - Have the requirement that the dependee is running before starting the dependent
  - Enforce ordering
- Weak dependencies
  - Ordering only, but no requirement

ORACLE





You can declare dependency relationships between resources. Relationships between individual resources can be between resources in the same resource group or in different resource groups. You can also configure group dependencies that take into account only the state of a group rather than the state of its resources.

There are four levels of resource dependencies: regular, weaker, restart, offline-restart.

**Regular Resource Dependencies**

If resource A depends on resource B, there is one hard-and-fast rule (known henceforth as the "prime directive of regular dependencies"):

Resource A cannot be brought online unless resource B is already online.

Note that this directive has some implications:

- The cluster must serialize starting A and B (waiting to start A until B is online) in cases where otherwise they could be started in parallel.
- The cluster must serialize stopping A and B (waiting to stop B until A is stopped) in cases where otherwise they could be stopped in parallel.

You are allowed to specifically and manually disable resource B, even if resource A is still running. This is somewhat counterintuitive, but because you are explicitly issuing the command (`clrs disable`) to disable the dependee, you can do it. Note that the prime directive (the first bullet item above) does not explicitly forbid you from stopping the resources in the “wrong order.”

**Weak Dependencies (Weaker Than Regular Dependencies)**

If resource A has a weak dependency on resource B, `rgmd` preserves the order of the stops and starts as if it were a regular dependency, but does not enforce the dependency in other ways. If B fails to start, the `rgmd` will still try to start A. You can manually disable either resource or its group freely.

# Understanding Resource Dependencies and Resource Group Dependencies

- Restart dependencies
  - Restart dependencies are stronger than regular dependencies.
  - The RGM restarts the dependent if the dependee restarts.
  - Dependencies could be missed if a fault monitor restarts the dependee without communicating to the RGM.
- Offline-restart dependencies
  - When the dependee goes offline, the RGM restarts the dependent (but the restart is blocked until the dependee goes back online).
- Resource group dependencies (fairly obsolete)

ORACLE





**Restart Dependencies (Stronger Than Regular Dependencies)**

Restart dependencies have all the attributes of regular dependencies, with the additional attribute that if the RGM is told to restart resource B, or is informed that some agent did the restart of resource B itself, the RGM will automatically restart resource A.

It is possible for a fault monitor, if it does not follow the standard conventions, to restart an application itself without ever informing RGM. In this case, the restart part of a restart dependency cannot be enforced.

Similar to the case of regular dependencies, you are allowed to manually and specifically disable the dependee, resource B, while leaving resource A running. However, when you manually re-enable resource B, the restart dependency begins and the RGM restarts resource A.

**Offline-Restart Dependencies (Slight Variation of Restart Dependencies)**

In this slight variation, when RGM is informed that the dependee, resource B, is offline, or when it is put explicitly offline, the RGM issues a restart on resource A. Resource A explicitly depends on resource B; therefore, resource A's start will be blocked until B actually restarts.

**Resource Group Dependencies**

Resource group dependencies imply an ordering relationship between two groups: If resource group G has a group dependency on resource group H, group G cannot be brought online unless group H is online. Group H cannot be brought offline unless group G is offline. This type of dependency considers the state of the group only, rather than the resources inside it.

Resource group dependencies are somewhat limited. They are enforced between groups running on different nodes when you manually try to start or stop resource groups in the wrong dependency order, regardless of which nodes the groups are mastered.

They consider only the group state and not individual resource state. Therefore, resource group dependencies are made obsolete by the various flavors of resource dependencies that are now possible between resources in different groups.

# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

# Configuring Resource and Resource Groups Through Properties

- Resources and resource groups are configured through sets of *name=value* properties.
- Standard resource properties
  - Certain property names apply to any resource.
  - Many of these (timeouts for methods, for example) have default values, and many administrators never know that they exist.
  - Standard resource properties include dependencies and some other interesting properties.
  - Standard resource properties can be accessed by using:

```
# man rg_properties
```

ORACLE



You configure specific resources within the cluster by defining values for resource properties and resource group properties. Properties consist of a set of *name=value* pairs.

These properties are used by the data service agent and by `rgmd`. There is no way they can be accessed directly by the application software itself, because it is cluster-unaware software.

Some properties are essential for running the service in the cluster. Scripts specifically used to run the service in the cluster can read these property values and use them to locate configuration files, or can use them to pass command-line parameters to the actual services.

Other properties are essential only for data service fault monitoring. Misconfiguring these properties might leave the service running fine, but cause fault monitoring to fail.

Each resource can have dozens of properties. Fortunately, many important properties for particular types of resources are automatically provided with reasonable default values. Therefore, most administrators of Oracle Solaris Cluster software environments never need to deal with the properties.

**Standard Resource Properties**

The names of standard resource properties can be used with any type of resource.

You can access a full list of standard resource properties and their general meanings by typing `# man rg_properties`.

Of the dozens of properties listed, only a handful are critical for a particular resource type. Other properties can be ignored, or can have default values that are unlikely to be changed.

# Configuring Resource and Resource Groups Through Properties

`Resource_dependencies`

- For example, an NFS application resource, might have dependency on an `HAStoragePlus` resource:

```
Resource_dependencies=nfs-stor
Resource_dependencies_weak
Resource_dependencies_restart
Resource_dependencies_offline_restart
Failover_mode
```



The following properties are standard resource properties. That is, they can have meaningful values for many different types of resources.

**`Resource_dependencies` Property**

Resource dependencies, which imply ordering, are configured using standard properties.

`Resource_dependencies=nfs-stor`

**`Resource_dependencies_weak` Property**

To set up a weak dependency so that resource A has a weak dependency on resource B, set the `Resource_dependencies_weak` property on resource A. (Recall that this implies an ordering but not a real dependency.)

`Resource_dependencies_weak=resB`

### `Resource_dependencies_restart` and `Resource_dependencies_offline_restart` Properties

To set the two kinds of restart dependencies, set the `Resource_dependencies_restart` or `Resource_dependencies_offline_restart` property on resource A.

```
Resource_dependencies_restart=resB

    Resource_dependencies_offline_restart=resB
```

### `Failover_mode` Property

The `Failover_mode` property describes what should happen to a resource if the resource fails to start up or shut down properly.

# Configuring Resource and Resource Groups Through Properties

Extension resource properties

- Are specific to resources of a particular type
- For example, a `SUNW.apache` instance has:

```
Bin_dir=directory_containing_apachectl
```

- For example, a `SUNW.HAStoragePlus` instance has:

```
AffinityOn=[true|false]
FilesystemMountpoints=mount-points
```

- Reference hint:

```
# man SUNW.apache
# man SUNW.HAStoragePlus
```



Names of extension properties are specific to resources of a particular type. You can get information about extension properties from the `man` page for a specific resource type, as in these examples:

- `# man SUNW.apache`
- `# man SUNW.HAStoragePlus`

For example, if you are setting up the Apache web server, you must create a value for the `Bin_dir` property, which points to the directory containing the `apachectl` script that you want to use to start the web server.

The `HAStoragePlus` type has the following important extension properties regarding file systems:

```
FilesystemMountPoints=list_of_storage_mount_points
AffinityOn=True/False
--------------------
Zpools=list_of_zpools
```

Use the first of these extension properties to identify which storage resource is being described. The second extension property is a parameter that tells the cluster framework to switch physical control of the storage group to the node running the service. Switching control to the node that is running the service optimizes performance when services in a single failover resource group are the only services accessing the storage. The third property (`Zpools`) is used only for specifying pool names for a failover ZFS resource, in which case the first two properties are not used.

Many resource types (including LDAP and DNS) have an extension property called `Confdir_list` that points to the configuration.

Many have other ways of identifying their configuration and data. There is no hard-and-fast rule about extension properties.

# Configuring Resource and Resource Groups Through Properties

Resource group properties

- Properties concerning an entire group (only)
  - For example, `Mode=[failover|scalable]`
  - Reference hint:

```
# man rg_properties
```

# Configuring Resource and Resource Groups Through Properties

Some significant resource group properties:

- `RG_dependencies` (somewhat obsolete)
- `Nodelist` (required)

```
Nodelist=clnode1,clnode2
```

- `Failback` (default value `False`)
- `Implicit_network_dependencies`
  - Default value is `true`.
- `Pingpong_interval`
- `Pathprefix` (group, but currently only used with NFS)





### Some Significant Resource Group Properties

The following properties are associated with a resource group rather than with an individual resource.

#### `RG_dependencies` Property

The `RG_dependencies` property will be set on a resource group to describe its dependency on another group. If resource group A has the property `RG_dependencies=rgB`, then resource group A cannot be brought online unless resource group B is online. Resource group B must be brought online somewhere, but not necessarily on the same node.

#### `Nodelist` Property

The `Nodelist` property for a resource group describes the nodes or zones on which the resource group can run.

The value of this property is always an ordered list, from most preferred node to least preferred node. As you learned earlier, it is possible for the same group node list to include multiple zones on the same physical node, although it is not a core function of the cluster zones integration feature.

### `Failback` Property

If the `Failback` property is `TRUE` (not the default), the resource group automatically switches back when a preferred node or zone (earlier in the node list) joins the cluster.

### `Implicit_network_dependencies` Property

The `Implicit_network_dependencies` property is `TRUE` by default. This property makes all the services in the resource group dependent on all the `LogicalHostname` and `SharedAddress` resources. That is, no services can start if the logical IP addresses cannot be brought online. You can set the `Implicit_Network_Dependencies` property to `FALSE` if you have a service that does not depend on logical IP addresses.

### `Pingpong_interval` Property

This property controls resource group behavior in two ways:

- If a resource group fails to start twice on the same particular node or zone (failure of `START` methods of same or different resources in the group) within the interval (expressed in seconds), the RGM does not consider that node or zone to be a candidate for the group failover.
- If the fault monitor for one particular resource requests that a group be failed off a particular node or zone, and if the fault monitor for the same resource then requests another failover that would bring the group back to the original node or zone, the RGM rejects the second failover if it is within the interval.

**Note:** The `Pingpong_interval` property is meant to prohibit faulty start scripts or properties and faulty fault monitors, or problem applications, from causing endless ping-ponging between nodes or Oracle Solaris zones.

### `Pathprefix` Property

The `Pathprefix` property points to a directory in a shared-storage file system that is used for the administrative purposes of services in the resource group.

Currently, the only data service that must have the `Pathprefix` property set is NFS, which uses the property to find its `dfstab` file. NFS needs the `dfstab` file so that it knows what to share. NFS also uses the same area to store file-lock state information. File-lock state information is typically stored in the `/etc` directory on a stand-alone NFS server, but it must be in the shared storage in the cluster.

An NFS resource must have its `dfstab` file named:

```
value_of_Pathprefix_for_RG/SUNW.nfs/dfstab.resource_name
```

When you create an NFS resource, the resource checks that the `Pathprefix` property is set on its resource group, and that the `dfstab` file exists.

# Flexible Load-Based Distribution of Resource Groups into Nodes

Load-based RG node selection criteria enhance the current behavior of the RGM.

Properties:

- `Loadlimits`
  - Assigned to each node
  - Hard/Soft
- `Loadfactors`
  - Assigned to resource groups
  - `Loadfactors` of resource groups on each node are summed up to provide a total load.
- `Priority`: Sets a priority for an RG
- `Preemption_mode`

ORACLE



Until the previous release, Oracle Solaris Cluster used a static, ordered node list, along with failure history and configured RG affinities, as the criteria for the distribution of resource groups onto cluster nodes. Starting with Oracle Solaris Cluster 3.3, load-based RG node selection criteria enhance the behavior of the RGM.

You can now configure a set of zero or more `loadlimits` for each node. Each resource group may be assigned a set of `loadfactors`. As resource groups are brought online on nodes by the RGM, the `loadfactors` of resource groups on each node are summed up to provide a total load that can be compared to that node's corresponding `loadlimits`.

Node capacities may be defined in terms of hard and/or soft `loadlimits`. Hard `loadlimits` are strictly enforced. The RGM will not allow a hard `loadlimit` to be exceeded by the RGs mastered by that node. Soft `loadlimits` may be exceeded if there is not enough free capacity in the cluster.

A node on which a soft limit is exceeded will be flagged with an `overloaded` status.

Two properties determine how RGM will fail over an RG:

- **RG `Priority` property:** An RG with higher priority is more likely to be mastered by its most preferred node—and less likely to be displaced off of that node—than an RG of lower priority.
- **RG `Preemption_mode` property:** By adjusting the `Preemption_mode`, you can make it more or less likely that a resource group will be preempted from a node by a higher-priority resource group due to node overload.

# Quiz

Which of the following statements is true?

a. `Loadlimits` are assigned to resource groups.

b. `Loadfactors` are assigned to nodes to specify the total load for a node.

c. `Priority` is set for a resource group.



**Answer: c**

# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

ORACLE

# Using the `clresourcetype (clrt)` Command

- If you specify a node list for a type:
  - You restrict that type to those nodes
  - The RGM will not assume methods for the type on other nodes
- To register resource types:

```
# clrt register SUNW.nfs
# clrt register -n clnode1,clnode2 SUNW.apache
```



You use `clresourcetype` (`clrt`) to register or unregister a resource type. Registering a resource type makes it known to the CCR and available to be used for new resources.

If you register a type and do not specify a node list, you are allowing that type to be used on all nodes and HA zones, even ones that are added and created in the future. That is, the default is to place no restriction on where resources of this registered type could run. The assumption is that you will have the data service agent installed on all nodes.

If you do specify a node list, you are restricting the resource type to certain nodes or zones. The cluster then allows you to specify only those nodes or Oracle Solaris zones for the resource groups that contain resources of that type. The only property of a registered resource type that you can modify with the `clrt` command is the node list.

### Registering Resource Types

The following examples show registration of resource types. The first example specifies a node list of all current and future nodes. The second example limits the node list to only the nodes mentioned.

```
# clrt register SUNW.nfs
# clrt register -n clnode1,clnode2 SUNW.apache
```

**Note:** You can specify a zone with the syntax *nodename*:*zonename*.

# Viewing Registered Resource Types

```
# clrt list -v
Resource Type            Node List
-------------            ---------
SUNW.LogicalHostname:4   <All>
SUNW.SharedAddress:2     <All>
SUNW.nfs:3.3             <All>
SUNW.HAStoragePlus:9     <All>
SUNW.apache:4.1           clnode1 clnode2
# clrt show apache
Registered Resource Types ===
Resource Type:                        SUNW.apache:4.1
  RT_description:                     Apache Web Server on Oracle Solaris Cluster
  RT_version:                         4.1
  API_version:                        2
  RT_basedir:                         /opt/SUNWscapc/bin
  Single_instance:                    False
  Proxy:                              False
  Init_nodes:                         All potential masters
  Installed_nodes:                    clnode1 clnode2
  Failover:                           False
  Pkglist:                            SUNWscapc
  RT_system:                          False
  Global_zone:                        False
```



`clrt list -v` shows the list of registered resource types and their node lists.

`clrt show` can display more about a specific registered type or all registered types (as shown in the slide).

`Failover` (`False`) means that the resource type is not limited to being failover only (that is, it can be failover or scalable). `Global_zone` (`False`) means that methods for the resource type will run in an HA zone when the type is configured in the HA zone.

**Unregistering Types**

You can unregister only types for which there are no remaining resources:

```
# clrt unregister nfs
```

# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

# Configuring Resource Groups by Using the `clresourcegroup (clrg)` Command

- You must have an RG before creating resources in it.
- You can delete only an empty RG.

```
clrg create [-n nodelist] [-p property […...]] RG_name
clrg delete RG_name
clrg set [-p property […...]] RG_name
```

Examples:

```
# clrg create -n clnode1,clnode2 -p Pathprefix=/global/nfs/admin nfs-rg
```



You must create a resource group before creating resources in it. In addition, you can only delete a resource group that contains no resources. The command syntax used is shown in the slide.

For example, the following command creates a resource group `nfs-rg` that runs on two physical nodes (not on HA zones). The `Pathprefix` property is set in anticipation of placing an NFS resource in the group:

```
# clrg create -n clnode1,clnode2 -p Pathprefix=/global/nfs/admin nfs-rg
```

The two commands are identical. Both specify a new resource group that specifies two HA zones on two different nodes as the potential masters. You can use the second syntax because the HA zones happen to have the same name.

**Note:** If you omit the node list when creating a group, it defaults to all physical nodes of the cluster and none of the HA zones.

# Displaying Group Configuration Information

```
# clrg show nfs-rg
=== Resource Groups and Resources ===

Resource Group:                                 nfs-rg
  RG_description:                                  <NULL>
  RG_mode:                                         Failover
  RG_state:                                        Unmanaged
  Failback:                                        False
  Nodelist:                                        clnode1 clnode2

# clrg show ora-rg
Resource Groups and Resources ===
Resource Group:                                 ora-rg
  RG_description:                                  <NULL>
  RG_mode:                                         Failover
  RG_state:                                        Unmanaged
  Failback:                                        False

# clrg show -p pathprefix nfs-rg
=== Resource Groups and Resources ===

Resource Group:                                 nfs-rg
  Pathprefix:                                      /global/nfs
```

ORACLE



You can display resource group information with `clrg list` or `clrg show`. Some examples are shown in the slide.

# Displaying Group Configuration Information

```
# clrg show -v nfs-rg
=== Resource Groups and Resources ===

Resource Group:                                 nfs-rg
  RG_description:                                  <NULL>
  RG_mode:                                         Failover
  RG_state:                                        Unmanaged
  RG_project_name:                                 default
  RG_affinities:                                   <NULL>
  RG_SLM_type:                                     manual
  Auto_start_on_new_cluster:                       True
  Failback:                                        False
  Nodelist:                                        clnode1 clnode2
  Maximum_primaries:                               1
  Desired_primaries:                               1
  RG_dependencies:                                 <NULL>
  Implicit_network_dependencies:                   True
  Global_resources_used:                           <All>
  Pingpong_interval:                               3600
  Pathprefix:                                      /global/nfs
  RG_System:                                       False
  Suspend_automatic_recovery:                      False
  Priority:                                        500
  Preemption_mode:                                 Has_Cost
  Load_factors:                                    <NULL>
```

# Configuring a `LogicalHostname` or a `SharedAddress` Resource

The following special commands are for creating and modifying `LogicalHostname` or `SharedAddress` IP resources:

- `clreslogicalhostname` (`clrslh`)
- `clressharedaddress` (`clrssa`)

These commands deal with two properties:

- `Hostnamelist`: The actual IPs (same subnet)
  - Defaults to a single IP name that is the same as the resource name
  - Abbreviation: `-h ipname1,[ipname2]...`
- `NetIflist`: Tells which adapters to use
  - Format: `ipmp_grp@node1,ipmp_grp@node2,...`
  - Assigns value automatically if there is only one per node on the subnet

ORACLE



The commands `clreslogicalhostname` (`clrslh`) and `clressharedaddress` (`clrssa`) are special commands for creating and modifying `LogicalHostname` and `SharedAddress` resources, respectively.

Specifically, these commands deal with two special properties of these IP resources and can simplify the administration of these properties.

- **`Hostnamelist`:** List of IPs (on a single subnet) corresponding to this resource
  - Multiple logical IP resources on different subnets must be separate resources, although they can still be in the same resource group.
  - Each name in `Hostnamelist` can be an IPV4 address, an IPV6 address, or both.
  - If you do not specify `Hostnamelist`, these commands assume that the resource name is also the value of `Hostnamelist`.

- **`NetIfList`:** Indication of which adapters to use on each node, in the following format: `ipmp_grp@node_id,ipmp_grp@node_id,..`

  If you do not specify `NetIfList`, the commands try to figure out whether there is only one IPMP group per node on the subnet indicated by `Hostnamelist`. If so, the value of `NetIfList` is automatically derived. When you use the `-N` shorthand of the `clrslh` command, you can use the node names, but the actual value of the property will contain the node IDs.

# Configuring a `LogicalHostname` or a `SharedAddress` Resource

- Use defaults for the special properties:

```
# clrslh create -g nfs-rg orangecat-nfs
```

- Provide values by using abbreviations:

```
# clrslh create -g nfs-rg -h orangecat-nfs,myother-ip \
-N therapy@clnode1,hug@clnode2 orangecat-nfs
```



**Examples of Using `clrslh` to Add a `LogicalHostname`**

In the first example in the slide, all the defaults are used. The `HostnameList` value is the same as the name of the resource.

The second example provides values for the two special properties. The command has shorthand options (`-h` and `-N`) for supplying the values.

# Configuring Other Resources by Using the `clresource` (`clrs`) Command

Create, delete, or modify any other resource:

```
clrs create -t resource_type_name -g RG_name \
[-p property […]] res-name
clrs set [-p property […]] res-name
clrs delete res-name
```

Examples:

```
# clrs create -t HAStoragePlus -g nfs-rg -p  AffinityOn=true \
-p FilesystemMountpoints=/global/nfs nfs-stor
# clrs create -t nfs -g nfs-rg \
-p  Resource_dependencies=nfs-stor nfs-res
```



Use the following command syntax to create, delete, and change properties of resources:

```
clrs create -t resource_type_name -g RG_name [-p property […]] res-name
clrs set [-p property […]] res-name
clrs delete res-name
```

For example, the following adds `HAStoragePlus` and an `nfs` resource:

```
# clrs create -t HAStoragePlus -g nfs-rg -p AffinityOn=true \
-p FilesystemMountPoints=/global/nfs nfs-stor
# clrs create -t nfs -g nfs-rg -p Resource_dependencies=nfs-stor nfs-res
```

# Configuring Other Resources by Using the `clresource` (`clrs`) Command

```
# clrs list -v
Resource Name       Resource Type            Resource Group
-------------       -------------            --------------
nfs-stor            SUNW.HAStoragePlus:8     nfs-rg
orangecat-nfs       SUNW.LogicalHostname:4   nfs-rg
nfs-res             SUNW.nfs:3.3             nfs-rg

# clrs show nfs-res

Resources ===

Resource:                                       nfs-res
  Type:                                            SUNW.nfs:3.3
  Type_version:                                    3.3
  Group:                                           nfs-rg
  R_description:
  Resource_project_name:                           default
  Enabled{clnode1}:                                True
  Enabled{clnode2}:                                True
  Monitored{clnode1}:                              True
  Monitored{clnode2}:                              True
```



### Displaying Resource Configuration Information

You can use `clrs list` and `clrs show` to display the configuration of any resource (including the IP resources), as shown in the slide.

# Configuring Other Resources by Using the `clresource` (`clrs`) Command

```
# clrs show -p resource_dependencies nfs-res

Resources ===

Resource:                                       nfs-res
  Resource_dependencies:                        nfs-stor

  --- Standard and extension properties ---
```



The slide displays resource configuration information.

# Complete Resource Group Example for NFS

1. Add (register) the resource types:

```
# clrt register SUNW.nfs
# clrt register SUNW.HAStoragePlus
```

2. Create the resource group:

```
# clrg create -n clnode1,clnode2 \
-p Pathprefix=/global/nfs/admin nfs-rg
```

3. Add the logical host name resource:

```
# clrslh create -g nfs-rg orangecat-nfs
```

4. Add the `SUNW.HAStoragePlus` resource:

```
# clrs create -t HAStoragePlus -g nfs-rg \
-p AffinityOn=true \
-p FilesystemMountPoints=/global/nfs nfs-stor
```



This example is based on the assumption that the NFS agent has been added. You should also assume that the `dfstab` file has been configured in the `SUNW.nfs` subdirectory of the directory referenced by the group's `Pathprefix` property.

# Complete Resource Group Example for NFS

5. Add the NFS service.

- Most application resources have their own properties to point to configuration files.
  - NFS does not (it uses the group's `Pathprefix`).
- Put the dependency on storage as a property.
- Dependency on `LogicalHostname` is implicit.

```
# clrs create -t nfs -g nfs-rg \
-p Resource_dependencies=nfs-stor nfs-res
```

6. Put the group online:

```
# clrg online -M nfs-rg
```

ORACLE



Although many services have properties that point to configuration files or binaries, NFS does not because it uses the `Pathprefix` property of the resource group. Dependency on the storage is expressed by using the standard property. Dependency on the `LogicalHostname` resource is implied.

State manipulations are covered in the next section.

# Modifying Properties with `clrs set -p` ...

Each property has a tunability characteristic that tells when it can be changed:

- `at_creation`: Property cannot be changed.
- `when_disabled`: Change with the `clrs set` command only if the resource is disabled.
- `anytime`: Change anytime with the `clrs set` command.



Oracle Solaris Cluster software allows the developer of a resource type to place restrictions on when particular properties can be set or changed. Each property for a particular type has a tunability characteristic that can be one of the following:

- `at_creation`: You can only set the property as you execute the `clrs create` command to add the resource.
- `when_disabled`: You can change the property with the `clrs set` command if the resource is disabled.
- `anytime`: You can change the property anytime with the `clrs set` command.

# Agenda

- Describing how data service agents enable a data service in a cluster to operate properly
- Listing the components of a data service agent
- Describing data service packaging, installation, and registration
- Describing the primary purpose of resource groups
- Differentiating between failover and scalable data services
- Describing how to use special resource types
- Listing the guidelines for using global and highly available local file systems
- Differentiating between standard, extension, and resource group properties
- Registering resource types
- Configuring resource groups and resources
- Controlling switching of resources and resource groups manually

# Controlling the State of Resources and Resource Groups

- Resource group state:
  - A resource group is created in an unmanaged state.
  - After it is managed, it can be controlled by the cluster.
  - When it is managed but offline, it is still subject to automatic recovery.
- Resource state
  - Resources have `enabled/disabled` flags.
    - One persistent flag per resource
    - Settable even if group is offline or unmanaged
    - Affects the resource when group is/goes online
    - Normally affects both resource and its fault monitor
  - If a resource is enabled, another flag enables/disables fault monitoring.





The following examples demonstrate how to use the `clrg` and `clrs` commands to control the state of resource groups, resources, and data service fault monitors.

**Resource Group State**

When a resource group is first created, it is in an unmanaged state. After it is brought into a managed state, the cluster provides automatic control and failover for its resources. The managed/unmanaged state is a “persistent” state that survives cluster reboots.

A resource group that is managed but offline can be thought of as “currently not online on any nodes, but still subject to automatic control and recovery.” That is, the cluster will still manage it and bring it online if the entire cluster reboots, or if the cluster reconfigures because any node fails or joins.

**Resource State**

Individual resources have a persistent disabled/enabled state flag that survives transitions of its groups state and survives cluster reboots. A resource that is disabled when its group is unmanaged or offline will still be disabled (and will not run) when the group goes online. If you change the `enabled`/`disabled` state of a resource while the group is online, it turns the resource on and off. If you do so while the group is not online, it affects what will happen to the resource when the group goes online.

If a resource is enabled, another similar state flag can disable and enable just the fault monitoring for the resource.

When a resource is created using `clrs create`, `clrslh create`, or `clrssa create`, it is automatically put into the `enabled` state, regardless of the state of its resource group.

# Controlling the State of Resources and Resource Groups

Online an offline group onto a preferred node:

- With `-M`, manages group first if unmanaged
- With `-e`, changes all resource flags to enabled

```
# clrg online [-M] [-e] nfs-rg
# clrg online [-M] [-e] -n node nfs-rg
```

Switch a group to a specific node:

```
# clrg switch [-M] [-e] -n node nfs-rg
```

Remaster a resource group:

```
# clrg remaster nfs-rg
```

Restart a resource group (bounce):

```
# clrg restart nfs-rg
```

Offline a resource group:

```
# clrg offline nfs-rg
```

ORACLE





**Resource Group Operations**

Use these commands to do the following:

- **Online an offline resource group onto its preferred node or a specific node:** With `-M`, it will bring the group into the managed state first if it is unmanaged. Without `-M`, the group must already be in the managed state. With `-e`, all disabled resources are enabled. Without `-e`, the enabled/disabled state of each resource is preserved.
- **Switch a failover resource group to a specific node:** (It might be online already on another node, or offline.) With `-M`, it will bring the group into the managed state first if it is unmanaged. Without `-M`, the group must already be in the managed state. With `-e`, all disabled resources are enabled. Without `-e`, the enabled/disabled state of each resource is preserved.

- **Remaster a failover resource group:** The group switches to its preferred node, even if it is already running on another node.
- **Offline a resource group:** The group remains in the managed state, so it is still subject to automatic recovery.
- **Restart a resource group.**

# Controlling the State of Resources and Resource Groups

Disable a resource and its fault monitor.

- A flag that persists even when the group is offline:

```
# clrs disable nfs-res
```

- Enable resource and its fault monitor:

```
# clrs enable nfs-res
```

- Clear the `STOP_FAILED` flag:

```
# clrs clear -f STOP_FAILED -n node nfs-res
```



**Resource Operations**

Use the `clrs` commands to do the following:

- Disable a resource and its fault monitor.
- Enable a resource and its fault monitor.
- Clear the `STOP_FAILED` flag.

# Controlling the State of Resources and Resource Groups

Disable fault monitor (but leave the resource enabled).

- A flag that persists even when the group is offline:

```
# clrs unmonitor nfs-res
```

- Enable fault monitor:

```
# clrs monitor nfs-res
```

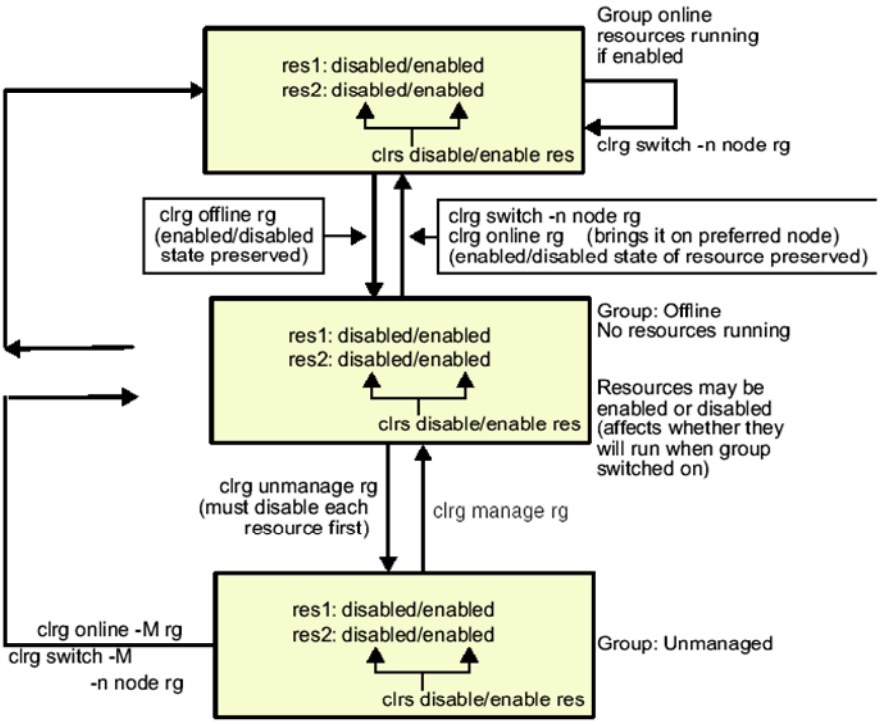


**Fault Monitor Operations**

Use the `clrs` commands to do the following:

- Disable the fault monitor for a resource.
- Enable the fault monitor for a resource.

# Summary of Resource Group and Resource Transitions






The diagram summarizes the resource and resource group transitions. Note the following:

- The `-M` switch for `clrg switch` or `clrg online` manages a group and then brings it online.
- By default, all transitions preserve the state of the `enabled/disabled` flags (for the service and its fault monitoring). You can force all resources to be enabled by adding the `-e` option to `clrg switch/online`.

# Suspended Resource Groups

When you suspend a group:

- You can still transition the group manually (as shown in the preceding slides)
- You can still enable and disable resources
- Fault monitors are still started
- Fault monitors do not cause a restart of resources
- Group does not automatically fail over

```
# clrg suspend nfs-rg
# clrs resume nfs-rg
```

ORACLE



When you suspend a resource group, you disable any automatic recovery, failover, or restarting of the resource group or the resources within it.

- You can still transition the resource group any way you like by using the commands presented in the previous slides.
- You can still enable/disable any individual resources by using the commands presented in the previous slides. If the resource group is online, the resources will go on and off accordingly.
- The fault monitors for resources will still be running.
- Resources will not automatically be restarted by fault monitors, nor will resource groups automatically fail over, even if an entire node fails.

The reason that you might want to suspend an online resource group is to perform maintenance on it—that is, start and stop some applications manually, but preserve the online status of the group and other components so that dependencies can still be honored correctly.

The reason that you might suspend an offline resource group is so that it does not go online automatically when you did not intend it to do so.

For example, when you put a resource group offline (but it is still managed and not suspended), a node failure still causes the group to go online.

To suspend a group, enter:

```
# clrg suspend grpname
```

To remove the suspension of a group, enter:

```
# clrg resume grpname
```

To see whether a group is currently suspended, use `clrg status` , as demonstrated in the next slide.

# Displaying Resource and Resource Group Status by Using the `clrg status` and `clrs status` Commands

Example of status commands for a single failover application:

```
# clrg status

Cluster Resource Groups ===

Group Name     Node Name           Suspended    Status
----------     ---------           ---------    ------
nfs-rg         clnode1             No           Offline
               clnode2             No           Online

# clrs status -g nfs-rg

Cluster Resources ===

Resource Name   Node Name   State        Status Message
-------------   ---------   -----        --------------
nfs-stor        clnode1     Offline      Offline
                clnode2     Online       Online

orangecat-nfs   clnode1     Offline      Offline
                clnode2     Online       Online - LogicalHostname online.

nfs-res         clnode1     Offline      Offline
                clnode2     Online       Online - Service is online.
```



There are separate commands to show the status of resource groups and resources. When viewing resources, you can still use `-g grpname` to limit the output to a single group.

An example of status commands for a single failover application is shown in the slide.

# Using the `clsetup` Utility for Resource and Resource Group Operations

```
*** Resource Group Menu ***
    Please select from one of the following options:

        1) Create a resource group
        2) Add a network resource to a resource group
        3) Add a data service resource to a resource group
        4) Resource type registration
        5) Online/Offline or Switchover a resource group
        6) Suspend/Resume recovery for a resource group
        7) Enable/Disable a resource
        8) Change properties of a resource group
        9) Change properties of a resource
       10) Remove a resource from a resource group
       11) Remove a resource group
       12) Clear the stop_failed error flag from a resource

        ?) Help
        s) Show current status
        q) Return to the main menu

    Option:
```



The `clsetup` utility has an extensive set of menus pertaining to resource and resource group management. These menus are accessed by selecting option 2 (Resource Group) from the main menu of the `clsetup` utility.

The `clsetup` utility is an intuitive, menu-driven interface that guides you through the options without having to remember the exact command-line syntax. The `clsetup` utility calls the `clrg`, `clrs`, `clrslh`, and `clrssa` commands, which were described in previous sections of this lesson.

The Resource Group menu for the `clsetup` utility looks similar to the output shown in the slide.

# Summary

In this lesson, you should have learned how to:

- Describe how data service agents enable a data service in a cluster to operate properly
- List the components of a data service agent
- Describe data service packaging, installation, and registration
- Describe the primary purpose of resource groups
- Differentiate between failover and scalable data services
- Describe how to use special resource types
- List the guidelines for using global and highly available local file systems
- Differentiate between standard, extension, and resource group properties
- Register resource types
- Configure resource groups and resources
- Control switching of resources and resource groups manually

ORACLE

# Practice 10 Overview: Installing and Configuring HA for NFS

This practice covers the following topics:

- Task 1: Installing and configuring the HA for NFS agent and server
- Task 2: Registering and configuring the Oracle Solaris Cluster HA for NFS data service
- Task 3: Verifying access by NFS clients
- Task 4: Observing Oracle Solaris Cluster HA for NFS failover behavior
- Task 5: Generating cluster failures and observing behavior of the NFS failover
- Task 6: Making a customized application fail over with a generic data service resource

ORACLE

11

# Configuring Scalable Services and Advanced Resource Group Relationships

# Objectives

After completing this lesson, you should be able to:

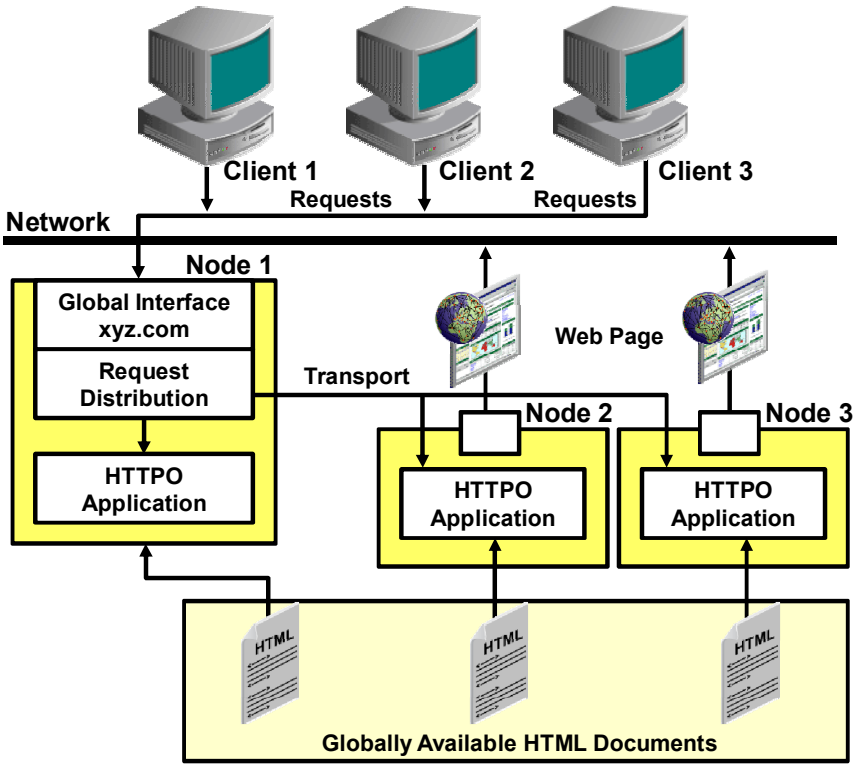
- Identify scalable services and shared addresses
- Describe the characteristics of scalable services
- Describe a `SharedAddress` resource
- Describe the properties of resource groups and scalable groups
- Describe how the `SharedAddress` resource works with scalable services
- Add auxiliary nodes for a `SharedAddress` property
- Review command examples for a scalable service
- Control scalable resources and resource groups
- View scalable resources and group status
- Describe advanced resource group relationships

ORACLE

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

# Using Scalable Services and Shared Addresses






The scalable service architecture allows some services, such as Apache web server, to run simultaneously on multiple nodes or HA zones, while appearing to the client to be running on a single server.

Clients connect to such a service by using a single IP address called a shared address. Typically, clients do not know which node or Oracle Solaris Zone they connect to.

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

# Exploring the Characteristics of Scalable Services

- Off-the-shelf applications are used as scalable services.
- File and data access is through global devices and a global file system.
- File locking should already be present.
- Thread synchronization does not translate properly into a scalable service.





Similar to the failover services described in previous lessons, the applications used as scalable services in the Oracle Solaris Cluster environment are generally off-the-shelf applications that are not specifically compiled or released to run in the cluster. The application binaries running on the various nodes are unaware of each other. This works because of the Oracle Solaris Cluster `SharedAddress` mechanism.

**File and Data Access**

Similar to the failover services, all the data for the scalable service must be in the shared storage.

Unlike the failover service, a file system–oriented scalable service must use the global file system. A `SUNW.HAStoragePlus` resource can still be configured to manage dependencies between the service and the storage.

### File Locking for Writing Data

One of the big barriers to taking just any application and turning it into a scalable service is that a service that is cluster-unaware may have been written in such a way as to ignore file-locking issues.

In the Oracle Solaris Cluster software, an application that performs data modification without any type of locking or file synchronization mechanism generally cannot be used as a scalable service.

Although the Oracle Solaris Cluster software provides the global data access methods, it does not automatically call any file-locking primitives.

Web servers that do file writing in common gateway interface (CGI) scripts, although not written specifically for a scalable platform, are usually written in such a way that multiple instances can be launched simultaneously even on a stand-alone server. Therefore, they already have the locking in place to make them ideal to work as scalable services in Oracle Solaris Cluster software.

Web servers that perform file writing using Java servlets must be examined much more closely. A servlet might use thread synchronization rather than file locking to enforce serial write access to a critical resource. This does not translate properly into a scalable service.

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

# Using the `SharedAddress` Resource

The `SharedAddress` resource provides:

- A single IP address for client usage
- Load balancing to the nodes on which the service is running
- Client affinity (per client as opposed to per connection) load balancing
- Load-balancing weights (per scalable service, not per `SharedAddress` resource)





The entity that holds a scalable service together in the Oracle Solaris Cluster software is the `SharedAddress` resource.

This resource not only provides a single IP address that makes the scalable service look similar to a single server from the point of view of the client, but also provides the load balancing of requests to all the nodes or Oracle Solaris Zones on which a scalable service is active.

**Client Affinity**

Certain types of applications require that load balancing in a scalable service be on a per-client basis, rather than a per-connection basis. That is, they require that the same client IP always have its requests forwarded to the same node or Oracle Solaris Zone. The prototypical example of an application requiring such client affinity is a shopping cart application, where the state of the client's shopping cart is recorded only in the memory of the particular node where the cart was created.

A single `SharedAddress` resource can provide standard (per-connection) load balancing and the type of sticky load balancing required by shopping carts. A scalable service's agent registers which type of load balancing is required, based on a property of the data service.

### Load-Balancing Weights

The Oracle Solaris Cluster software also lets you control the weighting that should be applied for load balancing. The default weighting is equal (connections or clients, depending on the stickiness) per node or Oracle Solaris Zone, but through the use of properties described in the following sections, a better node or Oracle Solaris Zone can be made to handle a greater percentage of requests.

**Note:** You could have a single `SharedAddress` resource that provides load balancing for the same application using multiple IP addresses on the same subnet. You could have multiple `SharedAddress` resources that provide load balancing for the same application using IP addresses on different subnets. The load balancing properties discussed here are set per scalable application and not per address.

# Quiz

From the point of view of a client, the `SharedAddress` resource provides a single IP address that makes the scalable service look similar to a single server.

a. True
b. False



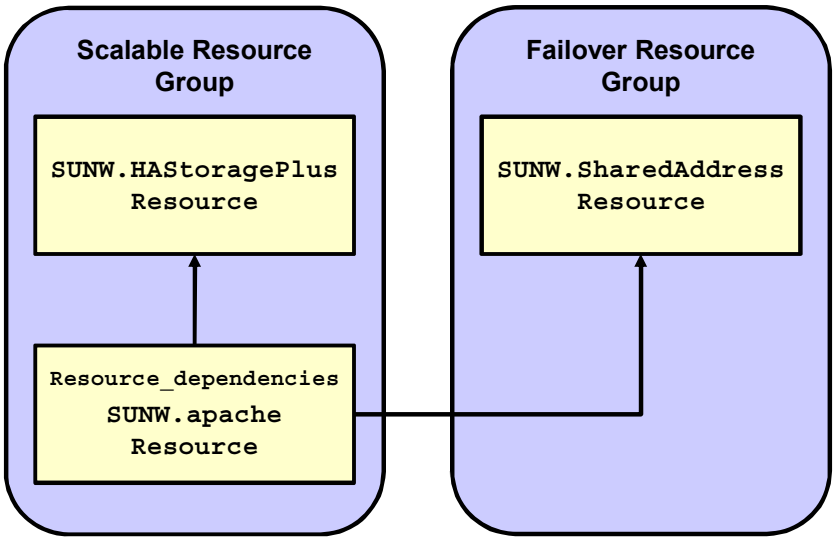
**Answer: a**

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

# Exploring Resource Groups for Scalable Services

- Failover group for the `SharedAddress` resource
- Scalable group for service and the `HAStoragePlus` resource




ORACLE



A scalable service requires the creation of two resource groups.

- **Failover resource group:** Contains the `SharedAddress` resource. It is online, or mastered, by only one node at a time. The node that masters the `SharedAddress` resource is the only one that receives incoming packets for this address from the public network.
- **Scalable resource group:** Contains an `HAStoragePlus` resource and the actual service resource

**Relationship Between Scalable and Failover Resource Groups**

The `HAStoragePlus` resource is added to guarantee that the storage is accessible on each node before the service is started on that node.

# Resources and Their Properties in the Resource Groups

The failover group for the `SharedAddress` resource

- Resource group name: `sa-rg`
- Properties: `[Nodelist=clnode1,clnode2 Mode=Failover Failback=False...]`

| Resource Name | Resource Type | Properties |
|---|---|---|
| `apache-lh` | `SUNW.SharedAddress` | `HostnameList=apache-lh Netiflist=therapy@1,therapy@2` |

ORACLE



The table displays the properties and contents of one of the two resource groups required to implement a scalable service.

# Resources and Their Properties in the Resource Groups

The scalable group: Resource group name: `apache-rg`

- Properties: `[Nodelist=clnode1,clnode2 Mode=Scalable Desired_primaries=2 Maximum_primaries=2]`

| Resource Name | Resource Type | Properties |
|---|---|---|
| web-stor | SUNW.HAStoragePlus | FilesystemMountPoints=/global/web<br>AffinityOn=False |
| apache-res | SUNW.apache | Bin_dir=/global/web/bin<br>Load_balancing_policy=LB_WEIGHTED<br>Load_balancing_weights=3@1,1@2<br>Scalable=TRUE<br>Port_list=80/tcp<br>Resource_dependencies=web-stor, apache-lh |

ORACLE



The table displays the properties and contents of the second resource group required to implement a scalable service.

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

# Properties for Scalable Groups and Services

- `Desired_primaries` and `Maximum_primaries` properties (group)
- `Load_balancing_policy` property (service)
- `Load_balancing_weights` property (service)
- `Resource_dependencies` property (service)



Certain resource group properties and certain resource properties are of particular importance for scalable services.

**`Desired_primaries` and `Maximum_primaries` Properties**

These are group properties that indicate how many nodes the service should run on. The `rgmd` tries to run the service on `Desired_primaries` nodes, but you can manually bring it online on more nodes by using the `clrg online` command, up to `Maximum_primaries`.

If these values are greater than `1`, the `Mode=Scalable` property is automatically set.

If you create a resource group using `clrg create -S`, the `Desired_primaries` and `Maximum_primaries` properties are both automatically set to the number of nodes in the `nodelist` for the group.

**`Load_balancing_policy` Property**

This is a property of the data service resource. It has one of the following values:

- **`Lb_weighted`:** Client connections are all load balanced. This is the default. Repeated connections from the same client might be serviced by different nodes.
- **`Lb_sticky`:** Connections from the same client IP to the same server port all go to the same node. Load balancing is only for different clients. This is only for the ports listed in the `Port_list` property.
- **`Lb_sticky_wild`:** Connections from the same client to any server port go to the same node. This is good when port numbers are generated dynamically and not known in advance.

**`Load_balancing_weights` Property**

This property controls the weighting of the load balancer. A value such as `5@1,3@2` indicates that node 1 will handle five requests (or clients, if using sticky policies) for every three requests that are handled by node 2. The default is that all nodes handle the same volume.

**`Resource_dependencies` Property**

You must set this property on a scalable resource. Its value must include the `SharedAddress` resource (in the other resource group). This is used by the data service agent to register the data service with the load balancer associated with the `SharedAddress` resource.

The same `Resource_dependencies` property of the scalable service also typically includes the `HAStoragePlus` resource in the same group.

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

# Adding Auxiliary Nodes for a `SharedAddress` Property

- By default, the node list for the `SharedAddress` resource group must contain at least all the nodes for the scalable service.
- If you want fewer nodes in the `SharedAddress` resource group, you must add auxiliary nodes:

```
# clrg create -n clnode1,clnode2 sa-rg
# clrssa create -g sa-rg -X apricot apache-lh
```



The `SharedAddress` logic includes a routine whereby the actual IP address associated with the resource is configured on the public network adapter (IPMP groups) on the primary node and is configured as a virtual address on the loopback network on all other nodes in its `nodelist`.

This enables scalable services that are running on nodes other than the primary `SharedAddress` node to still bind to the IP address used for the `SharedAddress`.

However, the `SharedAddress` resource must know about all possible nodes on which any scalable service associated with it might run.

If the node list for the `SharedAddress` resource group is a superset of the node list of every scalable service that might run on it, you are fine. But you might want to restrict which nodes might be the primary for the `SharedAddress`, while still allowing a larger node list for the scalable services dependent on it.

The SharedAddress resource has a special auxiliary nodes property that allows you to augment the node list of its group, just for the purposes of supporting more nodes for scalable services. This is set with the -X option to clrssa create.

In the following example, you want only nodes clnode1 and clnode2 to be the primaries for the SharedAddress (to actually host it on the public net and do the load balancing). But you might be supporting scalable services that run on clnode1, clnode2, and apricot:

```
# clrg create -n clnode1,clnode2 sa-rg
# clrssa create -g sa-rg -X apricot apache-lh
```

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

ORACLE

# Reviewing Command Examples for a Scalable Service

1. Register the resource types.
2. Create the failover resource group for `SharedAddress`.
3. Create the `SharedAddress` resource.
4. Bring the resource group online.
5. Create the scalable resource group.
6. Create the `HAStoragePlus` resource.
7. Create the Apache service.
8. Bring the resource group online.



These examples assume that the Apache web server agent was added from the Oracle Solaris Cluster Data Services CD by using the Oracle Solaris Cluster installer or the `pkgadd` command.

1. Register the resource types:

```
# clrt register SUNW.apache
# clrt register SUNW.HAStoragePlus
```

2. Create the failover resource group for the `SharedAddress`:

```
# clrg create -n clnode1,clnode2 sa-rg
```

3. Create the `SharedAddress` resource:

```
# clrssa create -g sa-rg apache-lh
```

4. Bring the resource group online:

```
# clrg online -M sa-rg
```

5. Create the scalable resource group:

```
# clrg create -S -n clnode1,clnode2 apache-rg
```

6. Create the `HAStoragePlus` resource:

```
# clrs create -t SUNW.HAStoragePlus -g apache-rg \
      -p FilesystemMountPoints=/global/web \
  -p AffinityOn=false \
   web-stor
```

7. Create the Apache service:

```
# clrs create -t SUNW.apache -g apache-rg \
            -p Resource_dependencies=web-stor,apache-lh \
            -p Bin_dir=/global/web/bin \
            -p Port_list=80/tcp \
            -p Scalable=TRUE \
   apache-res
```

8. Bring the resource group online:

```
# clrg online apache-rg
```

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

# Controlling Scalable Resources and Resource Groups

Resource group operations

- Bring an offline group online onto preferred nodes:
  - With `-M` (manages group first if it is unmanaged )

```
# clrg online [-M][-e] apache-rg
# clrg online -n node1,node2 [-M][-e] apache-rg
```

- Switch a group to specific nodes (from different nodes or from offline):

```
# clrg switch [-M][-e] -n node1,node2,.. apache-rg
```

ORACLE



The following examples demonstrate how to use the `clrg` and `clrs` commands to control the state of resource groups, resources, and data service fault monitors.

**Resource Group Operations**

Use the following commands to:

- Bring an offline resource group online onto its preferred nodes

```
# clrg online [-M] [-e] apache-rg
```

- Bring a resource group online on specific nodes (Any nodes that are not mentioned are not affected. If the group is running on other nodes, it just stays running there.)

```
# clrg online -n node,node [-M] [-e] apache-rg
```

- Switch a scalable resource group to the specified nodes (On any other nodes, it is switched off.)

```
# clrg switch [-M] [-e] -n node,node,.. apache-rg
```

# Controlling Scalable Resources and Resource Groups

Resource group operations

- Remaster a group (switch back to preferred nodes):

```
# clrg remaster apache-rg
```

- Bring a resource group offline (it goes off on all nodes):

```
# clrg offline apache-rg
```

- Bring a resource group offline on specific nodes:

```
# clrg offline -n node1,node2 apache-rg
```

- Restart a resource group (bounce):

```
# clrg restart apache-rg
```

- Restart a resource group on a particular node or nodes:

```
# clrg restart -n node1,node2,... apache-rg
```



The following are descriptions of the commands listed in the slide:

- Remaster a scalable resource group. The group switches to its most preferred nodes (the number of nodes is controlled by `Desired_primaries`). It may go offline on some nodes if it was previously running on more nodes or on nodes that were less preferred.
- Bring a resource group offline (it goes off on all nodes)
- Bring a resource group offline on specific nodes (Any nodes not mentioned are not affected. If the group is running on other nodes, it just stays running there.)
- Restart a resource group
- Restart a resource group on a particular node or nodes

# Controlling Scalable Resources and Resource Groups

Resource operations

- Disable a resource and its fault monitor:

```
# clrs disable apache-res
```

- Disable a resource and its fault monitor on specific nodes:

```
# clrs disable -n node1,node2,... apache-res
```

- Enable a resource and its fault monitor:

```
# clrs enable apache-res
```

- Enable a resource and its fault monitor (specific nodes):

```
# clrs enable -n apache-res
```

ORACLE

# Controlling Scalable Resources and Resource Groups

Fault monitor operations

- Disable a fault monitor (but leave resource enabled):

```
# clrs unmonitor apache-res
```

- Disable a fault monitor (specific nodes):

```
# clrs unmonitor -n node1,node2,.. apache-res
```

- Enable a fault monitor:

```
# clrs monitor apache-res
```

- Enable a fault monitor (specific nodes):

```
# clrs monitor -n node1,node2,... apache-res
```

ORACLE

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

# Using the `clrg status` and `clrs status` Commands for a Scalable Application

```
# clrg status
Cluster Resource Groups ===
Group Name Node Name Suspended Status
---------- --------- --------- ------
sa-rg      clnode1   No        Online
             clnode2   No        Offline
web-rg       clnode1   No        Online
             clnode2   No        Online

# clrs status -g sa-rg,web-rg
Cluster Resources ===
Resource Name   Node Name    State     Status Message
-------------   ---------    -----     --------------
orangecat-web   clnode1      Online    Online - SharedAddress online.
                clnode2      Offline   Offline
web-stor        clnode1      Online    Online
                clnode2      Online    Online
apache-res      clnode1      Online    Online - Service is online
                clnode2      Online    Online - Service is online.
```



If the `status` subcommands are used as shown in the slide, they will show the state of resource groups and resources for a scalable application.

# Agenda

- Identifying scalable services and shared addresses
- Describing the characteristics of scalable services
- Describing a `SharedAddress` resource
- Describing the properties of resource groups and scalable groups
- Describing how the `SharedAddress` resource works with scalable services
- Adding auxiliary nodes for a `SharedAddress` property
- Reviewing command examples for a scalable service
- Controlling scalable resources and resource groups
- Viewing scalable resources and group status
- Describing advanced resource group relationships

ORACLE

# Advanced Resource Group Relationships

Resource group affinities

- Preference or requirement that resource groups run on the same or different nodes
- Syntax (example shows a strong positive affinity):

```
# clrg set -p RG_affinities=++rg1 rg2
```



Oracle Solaris Cluster software offers a series of advanced resource group relationships called resource group affinities.

Resource group affinities provide a mechanism for specifying either a preference (weak affinities) or a requirement (strong affinities) that certain resource groups either run on the same node (positive affinities) or do not run on the same node (negative affinities).

In this section, the words *source* and *target* are used to refer to the resource groups with an affinities relationships. The source is the group for which the value of `RG_affinities` is set, and the target is the group referred to by the value of the property. So `rg2` is referred to as the "source" and `rg1` as the "target" in the following example:

```
# clrg set -p RG_affinities=++rg1 rg2
```

# Advanced Resource Group Relationships

Weak positive affinities and weak negative affinities

- A preference, not a requirement

```
# clrg set -p RG_affinities=-myrg1 myrg2
```

- Affects:
  - Failover of the `myrg2` group
  - Bringing online when no group is mentioned



Weak positive affinities and weak negative affinities place only a preference, not a requirement, on the location of the source group.

Setting a weak positive affinity says that the source group prefers to switch to the node already running the target group, if any. If the target group is not running at all, that is fine. You can freely switch either group online and offline, and you can freely and explicitly place either on any node that you want.

Setting a weak negative affinity means that the source group prefers any other node besides the target. Note that it is just a preference. You can freely and explicitly place either group on any node that you want.

Weak positive and weak negative group affinities are denoted by a single plus sign (+) or single minus sign (–), respectively.

In this example, group `rg2` is given a weak positive affinity for `rg1`. This does not affect the current location of either group.

```
# clrg set -p RG_affinities=+rg1 rg2
WARNING: resource group rg2 declares a weak positive affinity for
resource group rg1; but this affinity is not satisfied by the
current state of the two resource groups
```

The following will be affected by a weak positive affinity (if the target group is actually online):

- **Failover of the source group:** If the target is online, when the source group needs to fail over it will fail over to the node running the target group, even if that node is not a preferred node on the source group's node list.
- **Putting the resource group online onto a nonspecified node:**

  ```
  # clrg online source-grp
  ```

  Similarly, when a source group goes online and you do not specify a specific node, it will go onto the same node as the target, even if that node is not a preferred node on the source group's node list.

However, weak affinities are not enforced when you manually bring or switch a group onto a specific node. The following command will succeed even if the source group has a weak affinity for a target running on a different node.

```
# clrg switch -n specific-node src-grp
```

There can be as many resource groups as the value of the property. That is, a source can have more than one target. In addition, a source can have both weak positive and weak negative affinities. In these cases, the source prefers to choose a node satisfying the greatest possible number of weak affinities. For example, select a node that satisfies two weak positive affinities and two weak negative affinities rather than a node that satisfies three weak positive affinities and no weak negative affinities.

# Advanced Resource Group Relationships

## Strong positive affinities

```
# clrg set -p RG_affinities=++rg1 rg2
```

- `rg2` will run only on the node where `rg1` is already running.
- `rg1` will drag `rg2` with it (parent/child relationship).
- You cannot switch `rg1` to another node by itself.
- Why not just make them one big group?
  - You could still offline `rg1` and leave `rg2` running.
  - You might have restrictions against putting resources in the same group but still want collocation.

ORACLE



Strong positive affinities, which are indicated with a double plus sign (`++`) before the value of the target, place a requirement that the source run on the same node as the target. This example sets a strong positive affinity:

```
# clrg set -p RG_affinities=++rg1 rg2
```

Note the following:

- The only node or nodes on which the source can be online are nodes on which the target is online.
- If the source and target are currently running on one node, and if you switch the target to another node, it drags the source with it.
- If you offline the target group, it will offline the source as well.
- An attempt to switch the source to a node where the target is not running will fail.
- If a resource in the source group fails, the source group still cannot fail over to a node where the target is not running. (See the next section for the solution to this example.)

The source and target are closely tied together. If you have two failover resource groups with a strong positive affinity relationship, it might make sense to make them just one group. So why does strong positive affinity exist?

- The relationship can be between a failover group (source) and a scalable group (target). That is, the failover group must run on the same node already running the scalable group.
- You might want to be able to offline the source group but leave the target group running, which works with this relationship.
- For some reason, you might not be able to put some resources in the same group (the resources might check and reject you if you try to put it in the same group as another resource). But you still want them all to be running on the same node or nodes.

# Advanced Resource Group Relationships

Strong positive affinity with failover delegation

```
# clrg set -p RG_affinities=+++rg1 rg2
```

- Similar to strong positive affinities, but this one allows a fault monitor of a resource in `rg2` to suggest a failover



A slight variation on strong positive affinity is set with the triple plus sign (+++) syntax:

```
# clrg set -p RG_affinities=+++rg1 rg2
```

The only difference between +++ and ++ is that here (with +++), if a resource in the source group fails and its fault monitor suggests a failover, the failover can succeed. The RGM moves the target group to where the source wants to fail over, and then the source is dragged correctly.

# Advanced Resource Group Relationships

## Strong negative affinities

```
# clrg set -p RG_affinities=--rg1 rg2
```

- `rg2` categorically refuses to run on the same node as `rg1`.
- Even if only one node is left, `rg2` will go offline.



A strong negative affinity is set using the following syntax:

```
# clrg set -p RG_affinities=--rg1 rg2
```

Here, the source cannot run on the same node as the target. It absolutely refuses to switch to any node where the target is running.

If you switch the target to the node where the source is running, it moves the source out of the way and the source switches to a different node (if any). If there are no more nodes, the source switches offline.

For example, if you have a two-node cluster with both the source and target groups online (on different nodes), and if one node crashes (whichever it is), only the target group can remain running because the source group categorically refuses to run on the same node.

You need to be careful with strong negative affinities because the HA of the source group can be compromised.

# Quiz

Match the resource group affinity with its description:

Weak positive affinity

a. It has a requirement that the source run on the same node as the target.
b. The source group prefers to switch to the node already running the target group.
c. The source cannot run on the same node as the target.
d. The source group prefers any other node besides the target.



**Answer: b**

# Quiz

Match the resource group affinity with its description:

Weak negative affinity

a. It has a requirement that the source run on the same node as the target.
b. The source group prefers to switch to the node already running the target group.
c. The source cannot run on the same node as the target.
d. The source group prefers any other node besides the target.



**Answer: d**

# Quiz

Match the resource group affinity with its description:

Strong positive affinity

a. It has a requirement that the source run on the same node as the target.
b. The source group prefers to switch to the node already running the target group.
c. The source cannot run on the same node as the target.
d. The source group prefers any other node besides the target.



**Answer: a**

# Quiz

Match the resource group affinity with its description:

Strong negative affinity

a. It has a requirement that the source run on the same node as the target.
b. The source group prefers to switch to the node already running the target group.
c. The source cannot run on the same node as the target.
d. The source group prefers any other node besides the target.



**Answer: c**

# Summary

In this lesson, you should have learned how to:

- Identify scalable services and shared addresses
- Describe the characteristics of scalable services
- Describe a `SharedAddress` resource
- Describe the properties of resource groups and scalable groups
- Describe how the `SharedAddress` resource works with scalable services
- Add auxiliary nodes for a `SharedAddress` property
- Review command examples for a scalable service
- Control scalable resources and resource groups
- View scalable resources and group status
- Describe advanced resource group relationships

ORACLE

# Practice 11 Overview: Installing and Configuring Apache as a Scalable Service on Oracle Solaris Cluster

This practice covers the following topics:

- Task 1: Preparing for Apache Data Service configuration
- Task 2: Configuring the Apache environment
- Task 3: Testing the server on each node before configuring the data service resources
- Task 4: Registering and configuring the Apache Data Service
- Task 5: Verifying Apache web server access
- Task 6: Observing cluster failures
- Task 7: Configuring advanced resource group relationships

ORACLE

12

# Using Oracle Solaris Zones in Oracle Solaris Cluster

# Objectives

After completing this lesson, you should be able to:

- Describe Oracle Solaris Zones in Oracle Solaris 11
- Explain the use of HA for Zones
- Configure a failover zone
- Describe zone clusters
- Create a zone cluster
- Identify cross-cluster affinities and dependencies

# Agenda

- Oracle Solaris Zones in Oracle Solaris 11
- Identifying HA for Zones
- Configuring a failover zone
- Identifying zone clusters
- Creating a zone cluster
- Identifying cross-cluster affinities and dependencies

# Oracle Solaris Zones in Oracle Solaris 11

- The default zone brand in the Oracle Solaris 11 release is the `ipkg` brand.
- Features of global zones:
  - ID 0
  - Single instance of the Oracle Solaris kernel that is bootable and running on the system
  - Complete installation of the Oracle Solaris system software packages
  - Contains additional software packages or additional software, directories, files, and other data not installed through packages
  - Complete product database that contains information about all software components installed in the global zone
  - Holds configuration information specific to the global zone only
  - Aware of all devices and all file systems
  - Knowledge of nonglobal zone existence and configuration
  - Nonglobal zone can be configured, installed, managed, or uninstalled

ORACLE



Oracle Solaris Zone features are as follows:

- Is assigned ID 0 by the system
- Provides the single instance of the Oracle Solaris kernel that is bootable and running on the system
- Contains a complete installation of the Oracle Solaris system software packages
- Can contain additional software packages or additional software, directories, files, and other data not installed through packages
- Provides a complete and consistent product database that contains information about all software components installed in the global zone
- Holds configuration information specific to the global zone only, such as the global zone host name and file system table
- Is the only zone that is aware of all devices and all file systems
- Is the only zone with knowledge of nonglobal zone existence and configuration
- Is the only zone from which a nonglobal zone can be configured, installed, managed, or uninstalled

# Agenda

- Oracle Solaris Zones in Oracle Solaris 11
- Identifying HA for Zones
- Configuring a failover zone
- Identifying zone clusters
- Creating a zone cluster
- Identifying cross-cluster affinities and dependencies

# HA for Zones

- Cluster-unaware zone
- Whole zone fails over (zone is the resource).
- Benefits
  - Supports Oracle Solaris 10 (`solaris10`) and Oracle Solaris 11 (`solaris`) branded zones
  - You can run an application as a standard, nonclustered application in the zone. HA as whole zone fails over.
- Normal application agents are not used with this feature.
- Two ways to run applications:
  - Not controlled by cluster at all (failover with zone)
  - As a special generic resource

ORACLE



The Oracle Solaris Cluster Data Service for zones allows you to boot and fail over cluster-unaware nonglobal zones.

**Benefit of "Fail Over Whole Zone" Model**

Entire zone is treated as a resource that can fail over within the cluster. Nothing in the zone needs to be cluster-aware. Thus anything you want to run in the zone, and that you want to make fail over (because the whole zone fails over), can use this feature. This includes:

- Supports Oracle Solaris 10 (`solaris10`) and Oracle Solaris 11 (`solaris`) branded zones. These can fail over using this feature just as well as a regular native zone.
- Applications that run satisfactorily in a zone but are not "certified" as cluster applications in any other way

**Normal Cluster Data Service (Application) Agents are *Not* Used with This Feature**

The zones feature does not allow you to implement traditional Solaris Cluster agents (such as HA Oracle) in the zone. If you want to implement or migrate traditional applications supported in Oracle Solaris Cluster into the zone, you may be much more likely to use one of the other zone features.

**Running Applications Inside Highly Available Zones**

Using the zones feature, the agent itself running in the global zone will always be responsible for booting the nonglobal zones.

To run your applications inside the zone, you have a choice:

- After the zone is booted, you can treat it as a cluster-unaware entity. If you want to configure applications to run automatically when the zone boots, you do it the same way as in a nonclustered Oracle Solaris, using boot scripts or Service Management Facility (SMF) services.
- Alternatively, you could use additional optional pieces of the Oracle Solaris Cluster zone agent, which automatically use `zlogin` to access your zone, and launch and probe your application. The benefit here is that applications in these zones can still be viewed, managed, and monitored just like any other Solaris Cluster resource, and can have dependencies on other kinds of resources, including those running in traditional global zone resource groups and those running in resource groups related to other zone features.

# Failover Zones and Multiple Master Zones

- Zonepath in a failover file system
- IP addresses in zone
  - Without `LogicalHostname`
    - Configured in zone (just fail over with everything else)
    - Supports shared-IP and exclusive-IP zones
  - With `LogicalHostname`
    - Controlled by zone agent, which moves them into zone
    - Supports shared-IP zones only
    - Will make zone fail over if public network is broken





The application agent provides two different models for booting and controlling zones.

Failover zones have the following characteristics:

- The `zonepath` must be in shared storage configured as a failover file system.
- The zone is configured and installed manually only from one physical node, and then the zone's configuration is manually copied over to the other node.
- The control of the zone is managed by a failover resource group, as described in detail in the following sections.
- The zone can have a combination of IP address types as follows:
  - Some configured as `SUNW.LogicalHostname` instances and controlled by the zone agent. This works only with shared-IP zones.

Both types of IP addresses will appear to fail over along with the zone from node to node. The advantage of `SUNW.LogicalHostname` addresses is that their presence will also cause the zone to fail from node to node if all physical adapters in the IPMP group fail. However, if you have only this kind of address, your zone will not fail over, even if all network adapters in the IPMP group on that node are broken.

# Failover Zones and Multiple Master Zones

- Separate zones in local storage
- Agent controls booting of separate zones on nodes simultaneously
- No load balancing





Multiple master zones have the following characteristics:

- The zones are created manually on each node separately.
- The `zonepath` for the zones must be local to each node.
- The zone name must be the same on each node, but the `zonepath` can be the same or different.
- The zone is controlled by a scalable resource group, as described in detail in the following sections.
- The agent does not provide any internal load balancing or make use of the global interface feature in any way.
- The agent boots the same zone name on all nodes simultaneously. The idea is that they can contain instances of some application that needs to be load-balanced externally.

# Zone Boot (`sczbt`), Zone Script (`sczsh`), and Zone SMF (`sczsmf`) Resources

- "Chief resource" is the zone boot resource (controls booting and halting of zone, monitors health of zone).
- Other resources can control applications in zone:
  - Optional (can just launch application in zone as if nonclustered)
  - Benefits
    - Can have dependencies
    - Still have cluster control of application
  - Still always have to have the zone boot resource



The agent provides three new kinds of resources. Each of these is implemented using the generic data service type `SUNW.gds` as follows:

- **`sczbt` resource:** Provides booting and halting of the zone and, for failover zones, also manages placing any (optional) IP addresses managed by `SUNW.LogicalHostname` resources into the zone. An instance of the `sczbt` resource flavor is required in every resource group (both failover and multiple master) used to manage the zones.
- **`sczsh` resource:** Provides the ability to launch any software in the zone through a user-provided start script (that lives in the zone). Any instances of this resource are optional and, if present, must depend on the `sczbt` resource present in the same group.
- **`sczsmf` resource:** Provides the ability to enable an SMF service in the zone. Note that this resource does not configure the SMF service; it only enables it. Any instances of this resource are optional and, if present, must depend on the `sczbt` resource present in the same group.

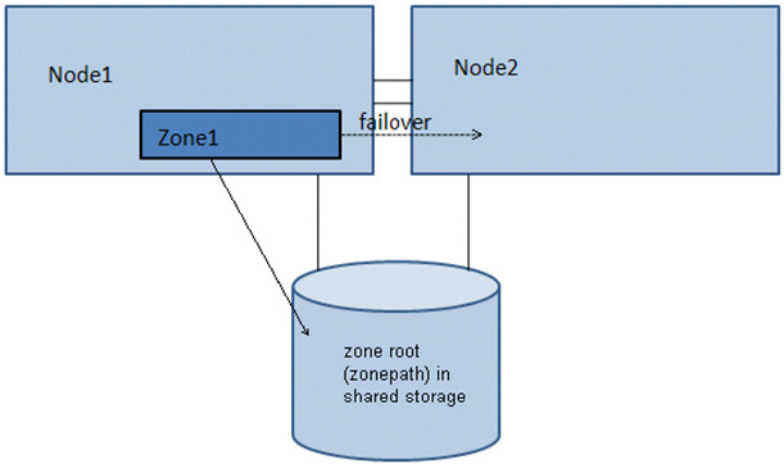
The `sczsh` and `sczsmf` resources are not required, and it is acceptable to configure your zone manually to run all the boot scripts and SMF services that you want. However, by using the zone agent resources, you gain the following benefits:

- You can have a fault-monitoring component. You can provide a custom fault probe for your resource, and its exit code can indicate a desire to have the entire resource group fail over (exit code 201) or restart (other nonzero exit code).
- You can have dependencies, even on resources that are present in other resource groups that can be online on a different node.

# Agenda

- Oracle Solaris Zones in Oracle Solaris 11
- Identifying HA for Zones
- Configuring a failover zone
- Identifying zone clusters
- Creating a zone cluster
- Identifying cross-cluster affinities and dependencies

# Failover Zones






Failover zones are one of the oldest features in Oracle Solaris Cluster. Here, a zone is treated as a black box that fails over from one node to another node. The `zonepath` for the zone is present in shared storage, and the zone is booted under control of Solaris Cluster on only one node at a time. The graphic in the slide illustrates this feature.

An advantage of this feature is that the zone can be treated as a completely cluster-unaware entity. Whichever applications happen to run when the zone boots will fail over as the zone itself fails or switches from node to node.

# Manually Configuring and Installing the Zone

```
# zonecfg -z s10zone
s10zone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:s10zone> create -t SYSsolaris10
zonecfg:s10zone> set zonepath=/s10zonepool
zonecfg:s10zone> set autoboot=true
zonecfg:s10zone> set ip-type=shared
zonecfg:s10zone> verify
zonecfg:s10zone> commit
zonecfg:s10zone> exit
```



The zone is configured and installed from one node only. The only configuration options given are the `zonepath` and an IP controlled by the zone (this is not required but merely serves as an example).

You *must not* set the `autoboot` parameter for the zone to `true`. The default is `false`.

# Manually Configuring and Installing the Zone

```
# zoneadm -z s10zone install -u -a
    /sw_install_dir/s10brandzones/s10enduser.flar
cannot create ZFS data set s10zonepool/s10zone: data set already
    exists
Log File: /var/tmp/s10zone.install.18259.log
Source: /net/server/marc/s10brandzones/s10enduser.flar
Installing: This may take several minutes...
Postprocessing: This may take several minutes...
# cd /etc/zones
# scp s10zone.xml othernode:/etc/zones
othernode:/# vi /etc/zones/index [ copy the s10zone line from
    installed node to new node]
# zoneadm -z s10zone boot
# zlogin -C s10zone
```

# Testing the Zone on Other Nodes

```
# clrg switch -n new_node s10zone-rg

# zoneadm -z s10zone boot
# zlogin s10zone
# uname -a
# exit

# zoneadm -z s10zone halt
```



Make sure you can boot the zone on each intended failover node. Leave it halted.

# Configuring the `sczbt` Resource Instance

```
# cd /opt/SUNWsczone/sczbt/util
# vi sczbt_config
...
RS=s10zone-rs
RG=s10zone-rg
PARAMETERDIR=/s10zone/params
SC_NETWORK=true
SC_LH=s10zone-lh
FAILOVER=true
HAS_RS=s10zone-stor
...
Zonename="s10zone"
Zonebrand="solaris10"
Zonebootopt=""
Milestone="multi-user-server"
LXrunlevel="3"
SLrunlevel="3"
Mounts=""
```



The `sczbt` resource instance is configured and registered using the configuration file and registration script that are typical of supported agents that are built on the `SUNW.gds` type.

Notice that the `sczbt_config` file is only used by the registration command `sczbt_register` and then is no longer needed. You will need to reuse the file if you create multiple zone boot resources (probably in different groups). Therefore, you should make a copy of it for safekeeping in case you ever need to delete the resource and add it back again.

There is a parameter file that needs to exist and is read every time the resource is stopped or started. Note that, in the `sczbt_config` file, you specify a directory name for the parameter file. This directory can be somewhere in the failover storage (as in the example), or you can manually make local copies.

# Configuring the `sczbt` Resource Instance

```
# mkdir /s10zone/params
# ./sczbt_register
...
# clrs status
# clrs enable s10zone-rs
# clrs status
# zlogin s10zone
// you should now see the s10zone-lh IP address automatically moved
   into the zone.
```



```
# mkdir /s10zone/params
# ./sczbt_register
```

Now enabling the zone boot resource instance will automatically boot the zone on the node where the failover group is primary. It will also automatically place the IP address controlled by the `SUNW.LogicalHostname` resource into the zone. This can be demonstrated by accessing the zone through this logical IP address:

```
# clrs status
# clrs enable s10zone-rs
# clrs status
# zlogin s10zone
// you should now see the s10zone-lh IP address automatically
   moved into the zone.
```

# Example: Script Resource

```
pecan:/# cd /opt/SUNWsczone/sczsh/util
pecan# cat sczsh_config
...
#
RS="s10zone-myd-rs"
RG="s10zone-rg"
SCZBT_RS="s10zone-rs"
PARAMETERDIR="/s10zone/params"
#
# The following parameters will be put in the agents parameterfile:
#
Zonename="s10zone"
ServiceStartCommand="/marc/mydaemon &"
ServiceStopCommand="/usr/bin/pkill mydaemon"
ServiceProbeCommand="/usr/bin/pgrep mydaemon >/dev/null"
pecan# ./sczsh_register
```

ORACLE





## `sczsh` and `sczsmf` Resource Instances

As mentioned previously, you can create optional resources to run some random software through a script (`sczsh`) or through an SMF service (`sczsmf`) instance.

Instances of each of these resource types *must* be present in the same resource group as a zone boot resource (`sczbt`). The configuration and registration scripts that are provided by the zone agent specifically for these resources will automatically place a restart dependency on the zone boot resource. Any other dependencies are fully customizable.

## Example: Script Resource

The following configuration file specifies a script resource that will depend on the zone boot resource defined in the previous example. The registration script, as for the zone boot resource, creates the parameter file, but if the specified directory is local on each node, the file must be copied manually:

```
pecan:/# cd /opt/SUNWsczone/sczsh/util
pecan# cat sczsh_config
…
```

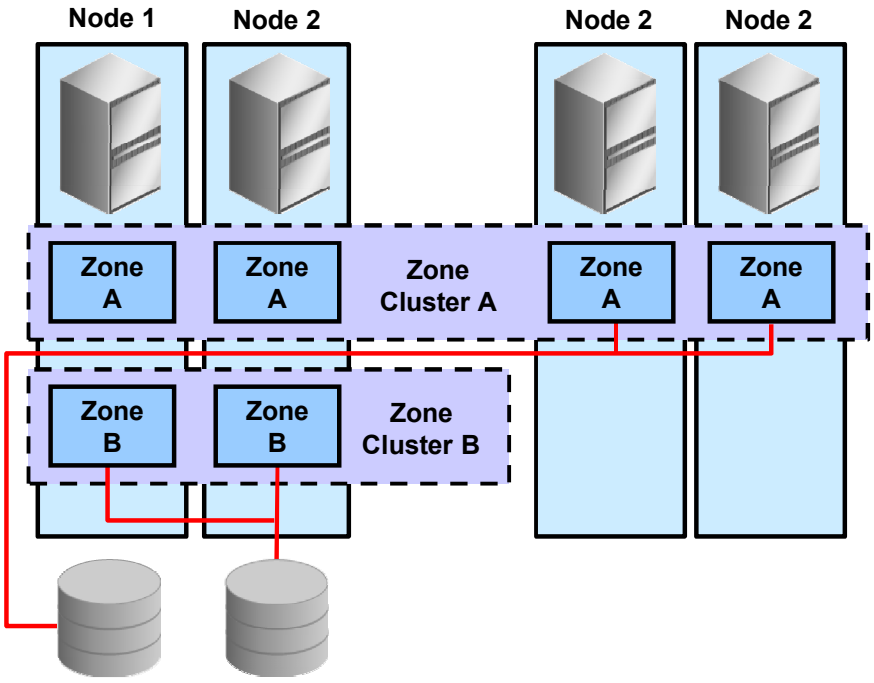
```
#
RS="s10zone-myd-rs"
RG="s10zone-rg"
SCZBT_RS="s10zone-rs"
PARAMETERDIR="/s10zone/params"
#
# The following parameters will be put in the agents
   parameterfile:
#
Zonename="s10zone"
ServiceStartCommand="/marc/mydaemon &"
ServiceStopCommand="/usr/bin/pkill mydaemon"
ServiceProbeCommand="/usr/bin/pgrep mydaemon >/dev/null"

pecan# ./sczsh_register
```

In this simple example, the new resource causes the `/marc/mydaemon` to be started in the zone after it is booted. Because the probe command used by the fault monitor is `/usr/bin/pgrep mydaemon >/dev/null`, which will never return the value `201`, failure of the probe will always cause a restart of the daemon rather than failover of the entire zone.

# Agenda

- Oracle Solaris Zones in Oracle Solaris 11
- Identifying HA for Zones
- Configuring a failover zone
- Identifying zone clusters
- Creating a zone cluster
- Identifying cross-cluster affinities and dependencies

# Zone Cluster






This is the newest model for zones. Here, you think of a set of zones (across different nodes) as forming their own cluster, a “zone cluster” within the normal “global cluster.” Several zone clusters can exist within the global cluster, which is illustrated in the graphic in the slide.

You can accomplish many of the same tasks by using either a zone cluster or a zone as a virtual node in a single “global cluster,” with a slightly different point of view. Here, the configuration of the cluster applications in the zones is isolated from the rest of the configuration, much like the processes in a zone are isolated from other zones.

# Zone Cluster

- A zone cluster is like its own cluster whose members are zones, with its own CCR.
- It has its own resource types, RGs, and resources.
- Inside a zone cluster, you see and manage only the resource types, resource groups, and resources for that zone cluster.
- From the global zone ("global cluster"), by default you can see objects for the global cluster, and you can see and manage objects from the zone cluster.
- Scoping is analogous to how processes run in zones.
- A zone cluster does *not* have its own quorum or heartbeat.
- Oracle RAC is the major validated application.

ORACLE



A good summary of a zone cluster is that it is "its own cluster whose node members are zones, with its own CCR."

A zone cluster acts like its own independent cluster, as far as nodes, resources, resource groups, and resource types are concerned.

When you are logged in to a zone that is part of a zone cluster, you manage the resources and resource groups and resource types of that zone cluster independently of those in the global cluster or in any other zone cluster.

One easy set of rules is the following:

- Inside a zone of a zone cluster, you can only manage and see RGs, resources, and resource types for that zone cluster.
- From the regular old cluster (global cluster), you can see and manage RGs, resources, and resource types for the global cluster and for all zone clusters.

**Note:** There is a conscious attempt to make scoping of resource types, resource groups, and resources inside zone cluster analogous to the scoping of processes and files inside a zone.

Zones in a zone cluster have their own virtual private network (really a subspace on the `clprivnet` network), but it is used for application traffic only (like Oracle RAC).

Currently zone clusters do not have their own quorum device or quorum configuration or their own private net heartbeat (there is no `clinterconnect` information used inside the zone).

Oracle RAC is the major application validated in a zone cluster. Some of the other general failover applications (failover MySQL, for example) are also validated.

# Zone Cluster Rules

- You must have regular old cluster first (“global cluster”).
- You can have multiple zone clusters.
- A zone can be in only one zone cluster.
- Zones of zone cluster must be on separate physical nodes.



- You must have a regular old cluster on nodes before any zones on those nodes can be involved in a zone cluster. The regular old cluster is known by a new term: the *global cluster*. There is no such thing as a freestanding zone cluster embedded in nonclustered physical nodes.
- After you have the regular old cluster (global cluster), you can have as many zone clusters as you like.
- A zone can be in only one zone cluster.
- Each zone cluster is composed of zones on different nodes (there is no such thing as a single zone cluster with multiple zones on the same node).

# Cluster Brand Zones

- New brand for zones of zone cluster
- Oracle Solaris 10 native-like zone with some special hooks for zone cluster



A new cluster brand (literally a cluster) is associated with all zones that are members of a zone cluster. An Oracle `solaris10` branded zone is a native Oracle Solaris 10 zone with a few special hooks for the special functionality of a zone cluster. There is no support for any other branded zone (no Solaris 9 or Solaris 8 or LX branded zones, for example) in the zone cluster.

# Creating and Managing Zone Clusters (`clzc` command)

- `clzonecluster` (`clzc`) command
- Is run from a single node; can configure, install, boot, halt, uninstall, or destroy entire zone cluster
- Zone naming and zone cluster
  - Actual zone name will be same across all zones in a zone cluster (will be the zone cluster name).
  - Oracle Solaris OS host name of each must be different.
  - Oracle Solaris OS host name of zone will be used as the node name in its zone cluster.

ORACLE



A new CLI command, `clzonecluster (clzc)`, can build the entire zone cluster from a single node. Underneath, the command communicates across the network with all the nodes and uses standard `zonecfg` and `zoneadm` commands to configure, install, and boot the zones of a zone cluster. You perform these functions as separate steps (`clzc configure`, `clzc install`, and so on), but all from a single node.

Here is a brief summary of how `clzc` subcommands work:

- **`clzc configure`:** Runs from a single node in the global cluster to configure a new zone cluster or modify the configuration of an existing zone cluster. The environment is very similar to the `zonecfg` command; it adds an outer scope for configurations that are common to the whole zone cluster. A full annotated example appears in the next section.

- **`clzc install`:** Runs from a single node in the global cluster to install all the zones of a zone cluster. Calls `zoneadm install` remotely on all cluster nodes to actually implement the installation. Adds automation for `sysidcfg` parts of the installation so that zone cluster nodes (zones) can configure themselves on the very first boot without manual intervention.
- **`clzc boot`** | **`halt`** | **`reboot`:** Can boot, halt, or reboot all the nodes (zones) of a zone cluster

**Zone Naming and Zone Cluster**

The name of the zones for a zone cluster (as seen by traditional global zone `zoneadm` and `zlogin` commands, for example) will be the same as the name of the zone cluster. All zones in the zone cluster will have this same name.

The Solaris OS host name of all these zones *must be different and unique within the zone cluster*. It is the Solaris OS host name that will be used as the "node name of the zone within its cluster.

The slight confusion is that all cluster utilities (regular `clrg`, `clrs` utilities, for example) will use the node name for a particular zone (the Solaris OS host name). However, you may still want to use traditional `zlogin` utilities to access the zone; here the name of the zone will be the zone cluster name and will be the same for all members of a zone cluster.

# Agenda

- Oracle Solaris Zones in Oracle Solaris 11
- Identifying HA for Zones
- Configuring a failover zone
- Identifying zone clusters
- Creating a zone cluster
- Identifying cross-cluster affinities and dependencies

# Installing and Booting a Zone Cluster

```
# clzc install marczc

# clzc boot marczc
```



The whole zone cluster is installed and booted from one physical node in the global zone. If you add the verbose (`-v`) option to the installation, you get traditional `zoneadm install` output on the console of each physical node.

# Example: Viewing Cluster Status

```
host01# clzc status
Zone Clusters ===
--- Zone Cluster Status ---
Name Node Name Zone HostName Status Zone Status
---- --------- ------------ ------ -----------
marczc host01  marc1         Online Running
       host02  marc2         Online Running
       host03  marc3         Online Running

host01# zoneadm list -cvi
ID NAME  STATUS   PATH         BRAND   IP
0 global running /             native  shared
2 marczc running /zones/marczc cluster shared

# zlogin marczc
[Connected to zone 'marczc' pts/4]
...
```



This is the first of several slides that show examples of using cluster commands in a zone cluster

The rules here are fairly simple:

- When you are logged in to a zone of a zone cluster, you get a view of resource types, resource groups, and resources that is completely private to that zone cluster.
- When you are logged in to a global zone:
  - By default, you see only configurations in the global zone. Configurations of zone clusters are hidden.
  - You can view configurations in a zone cluster by using `-Z zcname` or `-Z all`
  - You can manage configurations in a zone cluster by using `-Z` *`zcname`*

# Example: Viewing Cluster Node Status

```
marc1# clnode status
=== Cluster Nodes ===
--- Node Status ---
Node Name Status
--------- ------
marc1     Online
marc2     Online
marc3     Online
marc1# clrt register gds
marc1# clrg create -n marc1,marc2,marc3 marcrg

marc1# clrs create -g marcrg -t gds \
-p Start_command="/usr/bin/sleep 82000" \
-p Network_aware=false \
-p Probe_command=/bin/true marcres

marc1# clrg online -M marcrg
```

# Example: Viewing Cluster Resource Group Status

```
marc1# clrg status
=== Cluster Resource Groups ===
Group Name Node Name Suspended Status
---------- --------- --------- ------
marcrg     marc1     No        Online
           marc2     No        Offline
           marc3     No        Offline

marc1# clrs status
=== Cluster Resources ===
Resource Name Node Name State Status Message
------------- --------- ----- --------------
marcres       marc1 Online  Online - Service is online.
              marc2 Offline Offline
              marc3 Offline Offline
```

# Example: Viewing Cluster Resource Group Status in a Zone

```
// Note that if you were logged in to zones of any other
// zone cluster, this resource and group
// would not be visible at all.

// If you are back on the physical node, objects in the
// zone cluster
// are invisible by default, but you can still request
// to see a them using the new -Z option
host01# clrg status -Z marczc
Cluster Resource Groups ===
Group Name Node Name Suspended Status
---------- --------- --------- ------
marczc:marcrg  marc1 No Online
               marc2 No Offline
               marc3 No Offline
```

# Agenda

- Oracle Solaris Zones in Oracle Solaris 11
- Identifying HA for Zones
- Configuring a failover zone
- Identifying zone clusters
- Creating a zone cluster
- Identifying cross-cluster affinities and dependencies

# Cross-Cluster Affinities and Dependencies

- Affinities and dependencies between RG/resource in zone cluster and those of another zone cluster or global cluster
- Must be created or set from global cluster:

```
host01# clrg create -Z marczc -p \
RG_affinities=++global:somerg newrg

host01# clrs create -Z marczc -g newrg -p \
Resource_dependencies=global:someres newrs
```



You can set resource group affinities and resource dependencies between objects running in different zone clusters, or between objects running in a zone cluster and those running in the global cluster.

These configurations are somewhat unusual because they are visible inside a zone cluster (assuming the dependent or source RG of an affinity is in that zone cluster) but cannot be configured in the zone cluster. They must be configured in the global cluster:

```
host01# clrg create -Z marczc -p \
RG_affinities=++global:somerg newrg

host01# clrs create -Z marczc -g newrg -p \
Resource_dependencies=global:someres newrs
```

Note in these examples that the new resource group and its contained resource are in the zone cluster `marczc`. The resource and group can still be viewed and their state can be manipulated from within the zone cluster. The dependency will still be visible (with the value `global:someres`, for example) within the zone cluster.

# Summary

In this lesson, you should have learned how to:

- Describe Oracle Solaris Zones in Oracle Solaris 11
- Explain the use of HA for Zones
- Configure a failover zone
- Describe zone clusters
- Create a zone cluster
- Identify cross-cluster affinities and dependencies

# Practice 12-1 Overview: Running Failover Zones with HA for Zones and `solaris10` Branded Zones

This practice covers the following topics:

- Task 1: Verifying the Oracle `solaris10` branded zone support
- Task 2: Verifying the HA for Zones agent support
- Task 3: Finding a free disk and creating a failover ZFS pool and ZFS file system for the zonepath
- Task 4: Creating an Oracle `solaris10` branded zone with zonepath in the shared storage
- Task 5: Creating a logical host name entry for the zone
- Task 6: Creating a skeleton resource group for the failover zone

ORACLE

# Practice 12-1 Overview: Running Failover Zones with HA for Zones and `solaris10` Branded Zones

- Task 7: Halting the zone and replicating the zone configuration on the other cluster node
- Task 8: Switching the skeleton resource group to the other cluster node
- Task 9: Creating the resource to control the failover zone
- Task 10: Investigating node failures and zone failures with the failover zone

# Practice 12-2 Overview: Building a Zone Cluster

This practice covers the following topics:

- Task 1: Creating static addresses for zone cluster nodes
- Task 2: Configuring a zone cluster
- Task 3: Installing the zone cluster
- Task 4: Booting the zone cluster
- Task 5: Accessing the zone consoles
- Task 6: Exploring the zone cluster from the global zone point of view
- Task 7: Experimenting with zone cluster failures

ORACLE

# Practice 12-3 Overview: Configuring a Scalable Application in Zone Cluster

This practice covers the following topics:

- Task 1: Deleting the scalable Apache data service from the global cluster
- Task 2: Adding the cluster file system to a zone cluster
- Task 3: Configuring the Apache web server environment in the zone cluster
- Task 4: Creating a scalable Apache data service in the zone cluster
- Task 5: Verifying that the web server works

ORACLE