Abstract

This manual provides information about the advanced features for this version of Oracle Linux that have been engineered by Oracle.

Document generated on: 2015-11-11 (revision: 3386)
# Table of Contents

Preface ................................................................................................................................................... ix

1 The Unbreakable Enterprise Kernel ..................................................................................................... 1
   1.1 About the Unbreakable Enterprise Kernel ....................................................................................... 1
      1.1.1 About UEK Release 1 .................................................................................................................. 1
      1.1.2 About UEK Release 2 .................................................................................................................. 3
      1.1.3 About UEK Release 3 .................................................................................................................. 5
   1.2 Obtaining and Installing the UEK Packages ..................................................................................... 6
   1.3 For More Information About the UEK ............................................................................................ 7

2 Yum ....................................................................................................................................................... 9
   2.1 About Yum ....................................................................................................................................... 9
   2.2 Yum Configuration .......................................................................................................................... 9
      2.2.1 Configuring Use of a Proxy Server ............................................................................................... 10
      2.2.2 Yum Repository Configuration .................................................................................................. 11
   2.3 Downloading the Oracle Public Yum Repository Files .................................................................... 11
   2.4 Using Yum from the Command Line ............................................................................................... 12
   2.5 Yum Groups .................................................................................................................................... 13
   2.6 Installing and Using the Yum Security Plugin .................................................................................. 13
   2.7 Switching CentOS or Scientific Linux Systems to Use the Oracle Public Yum Server ................ 16
   2.8 Creating and Using a Local ULN Mirror .......................................................................................... 16
      2.8.1 Prerequisites for the Local ULN Mirror .................................................................................... 16
      2.8.2 Setting up a Local ULN Mirror .................................................................................................. 17
      2.8.3 ULN Mirror Configuration ......................................................................................................... 20
      2.8.4 Updating the Repositories on a Local ULN Mirror .................................................................. 20
      2.8.5 Configuring yum on a Local ULN Mirror ................................................................................ 21
      2.8.6 Configuring Oracle Linux Yum Clients of a Local ULN Mirror .............................................. 21
   2.9 Creating a Local Yum Repository Using an ISO Image ..................................................................... 23
   2.10 Setting up a Local Yum Server Using an ISO Image ....................................................................... 24
   2.11 For More Information About Yum ............................................................................................... 25

3 The Unbreakable Linux Network ........................................................................................................ 27
   3.1 About the Unbreakable Linux Network ............................................................................................ 27
   3.2 About ULN Channels ....................................................................................................................... 27
   3.3 About Software Errata .................................................................................................................... 29
   3.4 Registering as a ULN User .............................................................................................................. 29
   3.5 Registering an Oracle Linux 6 or Oracle Linux 7 System ................................................................. 30
   3.6 Registering an Oracle Linux 4 or Oracle Linux 5 System .................................................................. 30
   3.7 Configuring an Oracle Linux 5 System to Use yum with ULN ......................................................... 30
   3.8 Disabling Package Updates ............................................................................................................ 31
   3.9 Subscribing Your System to ULN Channels .................................................................................... 31
   3.10 Browsing and Downloading Errata Packages .............................................................................. 32
   3.11 Downloading Available Errata for a System .................................................................................. 32
   3.12 Updating System Details .............................................................................................................. 33
   3.13 Deleting a System .......................................................................................................................... 33
   3.14 About CSI Administration ............................................................................................................ 33
      3.14.1 Becoming a CSI Administrator ................................................................................................ 34
      3.14.2 Listing Active CSIs and Transferring Their Registered Servers ............................................. 35
      3.14.3 Listing Expired CSIs and Transferring Their Registered Servers ........................................... 36
      3.14.4 Removing a CSI Administrator ............................................................................................... 37
   3.15 Switching from RHN to ULN .......................................................................................................... 37
   3.16 For More Information About ULN .............................................................................................. 38

4 Ksplice Uptrack .................................................................................................................................... 39
   4.1 About Ksplice Uptrack .................................................................................................................... 39
Preface

The *Oracle Linux Administrator's Solutions Guide* provides information about the advanced features of Oracle Linux and, in particular, the Unbreakable Enterprise Kernel (UEK).

Audience

This document is intended for administrators who need to configure the advanced features of Oracle Linux and the Unbreakable Enterprise Kernel (UEK). It is assumed that readers are familiar with web and virtualization technologies and have a general understanding of the Linux operating system.

Document Organization

The document is organized as follows:

- **Chapter 1, The Unbreakable Enterprise Kernel** describes the advanced features that are available with the Unbreakable Enterprise Kernel (UEK).
- **Chapter 2, Yum** describes how to use the `yum` utility to install and upgrade software packages.
- **Chapter 3, The Unbreakable Linux Network** describes how to access and use the software channels that are available on the Unbreakable Linux Network (ULN).
- **Chapter 4, Ksplice Uptrack** describes how to configure Ksplice Uptrack to update a running system kernel.
- **Chapter 5, The Btrfs File System** describes how to deploy and use the advanced features of the btrfs file system.
- **Chapter 6, The XFS File System** describes how to deploy and use the advanced features of the XFS file system.
- **Chapter 7, Oracle Cluster File System Version 2** describes how to configure and use the Oracle Cluster File System Version 2 (OCFS2).
- **Chapter 8, Control Groups** describes how to use Control Groups (cgroups) to manage the resource utilization of sets of processes.
- **Chapter 9, Linux Containers** describes how to use Linux Containers (LXC) to isolate applications and entire operating system images from the other processes that are running on a host system.
- **Chapter 10, Docker** describes how to use the Docker Engine to create application containers.
- **Chapter 11, HugePages** describes how to set up the HugePages feature on a system that is running several Oracle Database instances.
- **Chapter 12, Using kexec for Fast Rebooting** describes how to use the `kexec` command to enable fast system rebooting.
- **Chapter 13, DTrace** introduces the dynamic tracing (DTrace) facility that you can use to examine the behavior of the operating system and the operating system kernel.
- **Chapter 14, Support Diagnostic Tools** describes the `sosreport`, Kdump, and OSWbb tools that can help diagnose problems with a system.

Documentation Accessibility

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

The documentation for this product is available at:


Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td>italic</td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Chapter 1 The Unbreakable Enterprise Kernel

Table of Contents

1.1 About the Unbreakable Enterprise Kernel ................................................................. 1
   1.1.1 About UEK Release 1 ......................................................................................... 1
   1.1.2 About UEK Release 2 ......................................................................................... 3
   1.1.3 About UEK Release 3 ......................................................................................... 5
1.2 Obtaining and Installing the UEK Packages ............................................................ 6
1.3 For More Information About the UEK ....................................................................... 7

This chapter describes the advanced features that are available with the Unbreakable Enterprise Kernel (UEK).

1.1 About the Unbreakable Enterprise Kernel

In September 2010, Oracle announced the new Unbreakable Enterprise Kernel (UEK) for Oracle Linux as a recommended kernel for deployment with Oracle Linux 5. Beginning with Oracle Linux 5.5, you could choose to use either the Red Hat Compatible Kernel or the UEK. In Oracle Linux 5.6, the UEK became the default kernel.

The prime motivation for creating the UEK was to provide a modern, high performance Linux kernel for the Exadata and Exalogic engineered systems. The kernel needed to scale as the number of CPUs, memory and InfiniBand connects was increased.

Oracle tests the UEK intensively with demanding Oracle workloads, and recommends the UEK for Oracle deployments and all other enterprise deployments. Oracle is committed to offering compatibility with Red Hat, and continues to release and support the Red Hat Compatible Kernel as part of Oracle Linux for customers that require strict RHEL compatibility. Under the Oracle Linux Support Program, customers can receive full support for Oracle Linux running with either kernel.

Oracle releases new versions of the UEK every 12-18 months. The latest version of the UEK receives quarterly patch updates including drivers for new hardware support, bug fixes, and critical security patches. Oracle also provides critical security patches for previous versions of the UEK. These patches are available as new installable kernels and, with the exception of device driver updates, as Ksplice patches.

Using the UEK instead of the Red Hat Compatible Kernel changes only the operating system kernel. There are no changes to any libraries, APIs, or any user-space applications Existing applications run unchanged regardless of which kernel you use. Using a different kernel does not change system libraries such as glibc. The version of glibc in Oracle Linux 6 remains the same, regardless of the kernel version.

1.1.1 About UEK Release 1

Release 1 of the UEK is based on a stable 2.6.32 Linux kernel and provides additional performance improvements, including:

- Improved IRQ (interrupt request) balancing.
- Reduced lock contention across the kernel.
- Improved network I/O by the use of receive packet steering and RDS improvements.
• Improved virtual memory performance.

The UEK release 1 includes optimizations developed in collaboration with Oracle’s Database, Middleware, and Hardware engineering teams to ensure stability and optimal performance for demanding enterprise workloads. In addition to performance improvements for large systems, the following UEK features are relevant to using Linux in the data center:

• The Infiniband OpenFabrics Enterprise Distribution (OFED) 1.5.1 implements Remote Direct Memory Access (RDMA) and kernel bypass mechanisms to deliver high-efficiency computing, wire-speed messaging, ultra-low microsecond latencies and fast I/O for servers, block storage and file systems. This also includes an improved RDS (reliable datagram sockets) stack for high-speed, low-latency networking. As an InfiniBand Upper Layer Protocol (ULP), RDS allows the reliable transmission of IPC datagrams up to 1 MB in size, and is currently used in Oracle Real Application Clusters (RAC), and in the Exadata and Exalogic products.

• A number of additional patches significantly improve the performance of Non-Uniform Memory Access (NUMA) systems with many CPUs, CPU cores, and memory nodes.

• Receive Packet Steering (RPS) is a software implementation of Receive Side Scaling (RSS) that improves overall networking performance, especially for high loads. RPS distributes the load of received network packet processing across multiple CPUs and ensures that the same CPU handles all packets for a specific combination of IP address and port.

To configure the list of CPUs to which RPS can forward traffic, use `/sys/class/net/interface/queues/rx-N/rps_cpus`, which implements a CPU bitmap for a specified network interface and receive queue. The default value is zero, which disables RPS and results in the CPU that is handling the network interrupt also processing the incoming packet. To enable RPS and allow a particular set of CPUs to handle interrupts for the receive queue on an interface, set the value of their positions in the bitmap to 1. For example, to enable RPS to use CPUs 0, 1, 2, and 3 for the `rx-0` queue on `eth0`, set the value of `rps_cpus` to `f` (that is, `1+2+4+8 = 15` in hexadecimal):

```
# cat f > /sys/class/net/eth0/queues/rx-0/rps_cpus
```

There is no benefit in configuring RPS on a system with a multiquene network device as RSS is usually automatically configured to map a CPU to each receive queue.

For an interface with a single transmit queue, you should typically set `rps_cpus` for CPUs in the same memory domain so that they share the same queue. On a non-NUMA system, this means that you would set all the available CPUs in `rps_cpus`.

**Tip**

To verify which CPUs are handling receive interrupts, use the command `watch -n1 cat /proc/softirqs` and monitor the value of `NET_RX` for each CPU.

• Receive Flow Steering (RFS) extends RPS to coordinate how the system processes network packets in parallel. RFS performs application matching to direct network traffic to the CPU on which the application is running.

To configure RFS, use `/proc/sys/net/core/rps_sock_flow_entries`, which sets the number of entries in the global flow table, and `/sys/class/net/interface/queues/rx-N/rps_flow_cnt`, which sets the number of entries in the per-queue flow table for a network interface. The default values are both zero, which disables RFS. To enable RFS, set the value of `rps_sock_flow_entries` to the maximum expected number of concurrently active connections, and the value of `rps_flow_cnt` to `rps_sock_flow_entries/Nq`, where `Nq` is the number of receive queues on a device. Any value that you enter is rounded up to the nearest power of 2. The suggested value of `rps_sock_flow_entries` is `32768` for a moderately loaded server.
• The kernel can detect solid state disks (SSDs), and tune itself for their use by bypassing the optimization code for spinning media and by dispatching I/O without delay to the SSD.

• The data integrity features verify data from the database all the way down to the individual storage spindle or device. The Linux data integrity framework (DIF) allows applications or kernel subsystems to attach metadata to I/O operations, allowing devices that support DIF to verify the integrity before passing them further down the stack and physically committing them to disk. The Data Integrity Extensions (DIX) feature enables the exchange of protection metadata between the operating system and the host bus adapter (HBA), and helps to prevent silent data corruption. The data-integrity enabled Automatic Storage Manager (ASM) that is available as an add-on with Oracle Database also protects against data corruption from application to disk platter.

For more information about the data integrity features, including programming with the block layer integrity API, see http://www.kernel.org/doc/Documentation/block/data-integrity.txt.

• Oracle Cluster File System 2 (OCFS2) version 1.6 includes a large number of features. For more information, see Chapter 7, Oracle Cluster File System Version 2.

1.1.2 About UEK Release 2

Note
The kernel version in UEK Release 2 (UEK R2) is stated as 2.6.39, but it is actually based on the 3.0-stable Linux kernel. This renumbering allows some low-level system utilities that expect the kernel version to start with 2.6 to run without change.

UEK R2 includes the following improvements over release 1:

• Interrupt scalability is refined, and scheduler tuning is improved, especially for Java workloads.

• Transcendent memory helps the performance of virtualization solutions for a broad range of workloads by allowing a hypervisor to cache clean memory pages and eliminating costly disk reads of file data by virtual machines, allowing you to increase their capacity and usage level. Transcendent memory also implements an LZO-compressed page cache, or zcache, which reduces disk I/O.

• Transmit packet steering (XPS) distributes outgoing network packets from a multiqueue network device across the CPUs. XPS chooses the transmit queue for outgoing packets based on the lock contention and NUMA cost on each CPU, and it selects which CPU uses that queue to send a packet.

To configure the list of CPUs to which XPS can forward traffic, use /sys/class/net/interface/queues/tx-N/xps_cpus, which implements a CPU bitmap for a specified network interface and transmit queue. The default value is zero, which disables XPS. To enable XPS and allow a particular set of CPUs to use a specified transmit queue on an interface, set the value of their positions in the bitmap to 1. For example, to enable XPS to use CPUs 4, 5, 6, and 7 for the tx-0 queue on eth0, set the value of rps_cpus to f0 (that is, 16+32+64+128 = 240 in hexadecimal):

```
cat f0 > /sys/class/net/eth0/queues/tx-0/xps_cpus
```

There is no benefit in configuring XPS for a network device with a single transmit queue.

For a system with a multiqueue network device, configure XPS so that each CPU maps onto one transmit queue. If a system has an equal number of CPUs and transit queues, you can configure exclusive pairings in XPS to eliminate queue contention. If a system has more CPUs than queues, configure CPUs that share the same cache to the same transmit queue.

• The btrfs file system for Linux is designed to meet the expanding scalability requirements of large storage subsystems. For more information, see Chapter 5, The Btrfs File System.
• Cgroups provide fine-grained control of CPU, I/O and memory resources. For more information, see Chapter 8, Control Groups.

• Linux containers provide multiple user-space versions of the operating system on the same server. Each container is an isolated environment with its own process and network space. For more information, see Chapter 9, Linux Containers.

• Transparent huge pages take advantage of the memory management capabilities of modern CPUs to allow the kernel to manage physical memory more efficiently by reducing overhead in the virtual memory subsystem, and by improving the caching of frequently accessed virtual addresses for memory-intensive workloads. For more information, see Chapter 11, HugePages.

• DTrace allows you to explore your system to understand how it works, to track down performance problems across many layers of software, or to locate the causes of aberrant behavior. DTrace is currently available only on ULN. For more information, see Chapter 13, DTrace.

• The configfs virtual file system, engineered by Oracle, allows you to configure the settings of kernel objects where a file system or device driver implements this feature. configfs provides an alternative mechanism for changing the values of settings to the ioctl() system call, and complements the intended functionality of sysfs as a means to view kernel objects.

The cluster stack for OCFS2, O2CB, uses configfs to set cluster timeouts and to examine the cluster status.

The low-level I/O (LIO) driver uses configfs as a multiprotocol SCSI target to support the configuration of FCoE, Fibre Channel, iSCSI and InfiniBand using the lio-utils tool set.

For more information about the implementation of configfs, see http://www.kernel.org/doc/Documentation/filesystems/configfs/configfs.txt.

• The dm-nfs feature creates virtual disk devices (LUNs) where the data is stored in an NFS file instead of on local storage. Managed networked storage has many benefits over keeping virtual devices on a disk that is local to the physical host.

The dm-nfs kernel module provides a device-mapper target that allows you to treat a file on an NFS file system as a block device that can be loopback-mounted locally.

The following sample code demonstrates how to use dmsetup to create a mapped device (/dev/mapper/$dm_nfsdev) for the file $filename that is accessible on a mounted NFS file system:

```
nblks=`stat -c '%s' $filename`
echo -n "0 $nblks nfs $filename 0" | dmsetup create $dm_nfsdev
```

A sample use case is the fast migration of guest VMs for load balancing or if a physical host requires maintenance. This functionality is also possible using iSCSI LUNs, but the advantage of dm-nfs is that you can manage new virtual drives on a local host system, rather than requiring a storage administrator to initialize new LUNs.

dm-nfs uses asynchronous direct I/O so that I/O is performed efficiently and coherently. A guest's disk data is not cached locally on the host. If the host crashes, there is a lower probability of data corruption. If a guest is frozen, you can take a clean backup of its virtual disk, as you can be certain that its data has been fully written out.
1.1.3 About UEK Release 3

Note

The kernel version in UEK Release 3 (UEK R3) is based on the mainline Linux kernel version 3.8.13. Low-level system utilities that expect the kernel version to start with 2.6 can run without change if they use the UNAME26 personality (for example, by using the uname26 wrapper utility).

UEK R3 includes the following major improvements over UEK R2:

• Integrated DTrace support in the UEK R3 kernel and user-space tracing of DTrace-enabled applications.

• Device mapper support for an external, read-only device as the origin for a thinly-provisioned volume.

• The loop driver provides the same I/O functionality as dm-nfs by extending the AIO interface to perform direct I/O. To create the loopback device, use the losetup command instead of dmsetup. The dm-nfs module is not provided with UEK R3.

• Btrfs send and receive subcommands allow you to record the differences between two subvolumes, which can either be snapshots of the same subvolume or parent and child subvolumes.

• Btrfs quota groups (qgroups) allow you to set different size limits for a volume and its subvolumes.

• Btrfs supports replacing devices without unmounting or otherwise disrupting access to the file system.

• Ext4 quotas are enabled as soon as the file system is mounted.

• TCP controlled delay management (CoDel) is a new active queue management algorithm that is designed to handle excessive buffering across a network connection (bufferbloat). The algorithm is based on how long packets are buffered in the queue rather than the size of the queue. If the minimum queuing time rises above a threshold value, the algorithm discards packets and reduces the transmission rate of TCP.

• TCP connection repair implements process checkpointing and restart, which allows a TCP connection to be stopped on one host and restarted on another host. Container virtualization can use this feature to move a network connection between hosts.

• TCP and STCP early retransmit allows fast retransmission (under certain conditions) to reduce the number of duplicate acknowledgements.

• TCP fast open (TFO) can speed up the opening of successive TCP connections between two endpoints by eliminating one round time trip (RTT) from some TCP transactions.

• The TCP small queue algorithm is another mechanism intended to help deal with bufferbloat. The algorithm limits the amount of data that can be queued for transmission by a socket.

• The secure computing mode feature (seccomp) is a simple sandbox mechanism that, in strict mode, allows a thread to transition to a state where it cannot make any system calls except from a very restricted set (_exit(), read(), sigreturn(), and write()) and it can only use file descriptors that were already open. In filter mode, a thread can specify an arbitrary filter of permitted systems calls that would be forbidden in strict mode. Access to this feature is by using the prctl() system call. For more information, see the prctl(2) manual page.

• The OpenFabrics Enterprise Distribution (OFED) 2.0 stack supports the following protocols:

  • SCSI RDMA Protocol (SRP) enables access to remote SCSI devices via remote direct memory access (RDMA)
• iSCSI Extensions for remote direct memory access (iSER) provide access to iSCSI storage devices
• Reliable Datagram Sockets (RDS) is a high-performance, low-latency, reliable connectionless protocol for datagram delivery
• Sockets Direct Protocol (SDP) supports stream sockets for RDMA network fabrics
• Ethernet over InfiniBand (EoIB)
• IP encapsulation over InfiniBand (iPoIB)
• Ethernet tunneling over InfiniBand (eIPoIB)
The OFED 2.0 stack also supports the following RDS features:
• Async Send (AS)
• Quality of Service (QoS)
• Automatic Path Migration (APM)
• Active Bonding (AB)
• Shared Request Queue (SRQ)
• Netfilter (NF)
• Paravirtualization support has been enabled for Oracle Linux guests on Windows Server 2008 Hyper-V or Windows Server 2008 R2 Hyper-V.
• The Virtual Extensible LAN (VXLAN) tunneling protocol overlays a virtual network on an existing Layer 3 infrastructure to allow the transfer of Layer 2 Ethernet packets over UDP. This feature is intended for use by a virtual network infrastructure in a virtualized environment. Use cases include virtual machine migration and software-defined networking (SDN).

The UEK R3 kernel packages are available on the ol6_x86_64_UEKR3_latest channel. For more information, see the Unbreakable Enterprise Kernel Release 3 Release Notes.

1.2 Obtaining and Installing the UEK Packages

You can obtain and install the UEK and associated firmware packages in the following ways:

• If you have a valid Oracle Linux Support subscription, you can obtain the latest Oracle Linux and UEK packages from the Unbreakable Linux Network (ULN) at http://linux.oracle.com. After you have logged in to ULN and registered your system, you can subscribe the system to the UEK channel for the appropriate Oracle Linux release and machine architecture. This channel will provide the latest Oracle Linux packages and updates for your system as they become available.

For more information about ULN, see Chapter 3, The Unbreakable Linux Network

• You can obtain Oracle Linux and UEK packages from the public yum package repository. To enable access, download the appropriate configuration file, such as http://public-yum.oracle.com/public-yum-ol6.repo to the /etc/yum.repos.d directory, and edit the file to enable the repositories from which you want to receive updates, such as ol6_UEK_base for the base Oracle Linux 6 Unbreakable Enterprise Kernel repository, ol6_UEK_latest for UEK bug fixes, errata and quarterly driver updates, and ol6_x86_64_UEKR3_latest for the kernel packages that are specific to UEK R3. You can use the yum command to download and install the packages.
For more information about *yum*, see *Chapter 2, Yum*

To list the installed kernel packages and also the kernel packages that are available to be installed from the repositories that you have enabled, use the following *yum* command:

```bash
# yum list kernel*
```

<table>
<thead>
<tr>
<th>Installed Packages</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel.x86_64</td>
<td>2.6.32-220.el6</td>
</tr>
<tr>
<td>kernel.x86_64</td>
<td>2.6.32-279.el6</td>
</tr>
<tr>
<td>kernel.x86_64</td>
<td>2.6.32-279.2.1.el6</td>
</tr>
<tr>
<td>kernel-devel.x86_64</td>
<td>2.6.32-220.el6</td>
</tr>
<tr>
<td>kernel-devel.x86_64</td>
<td>2.6.32-279.el6</td>
</tr>
<tr>
<td>kernel-devel.x86_64</td>
<td>2.6.32-279.2.1.el6</td>
</tr>
<tr>
<td>kernel-firmware.noarch</td>
<td>2.6.32-279.2.1.el6</td>
</tr>
<tr>
<td>kernel-uek.x86_64</td>
<td>2.6.39-200.24.1.el6uek installed</td>
</tr>
<tr>
<td>kernel-uek-devel.x86_64</td>
<td>2.6.32-300.32.1.el6uek @ol6_latest</td>
</tr>
<tr>
<td>kernel-uek-devel.x86_64</td>
<td>2.6.39-200.24.1.el6uek @ol6_UEK_latest</td>
</tr>
<tr>
<td>kernel-uek-devel.x86_64</td>
<td>2.6.39-200.24.1.el6uek @ol6_UEK_latest</td>
</tr>
<tr>
<td>kernel-uek-devel.x86_64</td>
<td>2.6.39-200.24.1.el6uek</td>
</tr>
<tr>
<td>kernel-uek-devel.x86_64</td>
<td>2.6.39-200.29.2.el6uek</td>
</tr>
<tr>
<td>kernel-uek-headers.x86_64</td>
<td>2.6.32-300.32.1.el6uek @ol6_latest</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Available Packages</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel.x86_64</td>
<td>2.6.32-279.5.2.el6</td>
</tr>
<tr>
<td>kernel-debug.x86_64</td>
<td>2.6.32-297.5.2.el6</td>
</tr>
<tr>
<td>kernel-debug-devel.x86_64</td>
<td>2.6.32-279.5.2.el6</td>
</tr>
<tr>
<td>kernel-doc.noarch</td>
<td>2.6.32-279.5.2.el6</td>
</tr>
<tr>
<td>kernel-firmware.noarch</td>
<td>2.6.32-279.5.2.el6</td>
</tr>
<tr>
<td>kernel-headers.x86_64</td>
<td>2.6.32-279.5.2.el6</td>
</tr>
<tr>
<td>kernel-uek.x86_64</td>
<td>2.6.39-200.29.3.el6uek @ol6_UEK_latest</td>
</tr>
<tr>
<td>kernel-uek-debug.x86_64</td>
<td>2.6.39-200.29.3.el6uek @ol6_UEK_latest</td>
</tr>
<tr>
<td>kernel-uek-debug-devel.x86_64</td>
<td>2.6.39-200.29.3.el6uek</td>
</tr>
<tr>
<td>kernel-uek-doc.noarch</td>
<td>2.6.39-200.29.3.el6uek</td>
</tr>
<tr>
<td>kernel-uek-firmware.noarch</td>
<td>2.6.39-200.29.3.el6uek</td>
</tr>
<tr>
<td>kernel-uek-headers.x86_64</td>
<td>2.6.32-300.32.1.el6uek</td>
</tr>
</tbody>
</table>

Alternatively, you can use the `rpm -qa` command to list the installed packages:

```bash
# rpm -qa | grep ^kernel | sort
```

Chapter 2 Yum

Table of Contents

2.1 About Yum .......................................................... 9
2.2 Yum Configuration .................................................. 9
   2.2.1 Configuring Use of a Proxy Server ....................... 10
   2.2.2 Yum Repository Configuration........................... 11
2.3 Downloading the Oracle Public Yum Repository Files ............. 11
2.4 Using Yum from the Command Line ................................ 12
2.5 Yum Groups ....................................................... 13
2.6 Installing and Using the Yum Security Plugin .................... 13
2.7 Switching CentOS or Scientific Linux Systems to Use the Oracle Public Yum Server .......................... 16
2.8 Creating and Using a Local ULN Mirror .......................... 16
   2.8.1 Prerequisites for the Local ULN Mirror .................. 16
   2.8.2 Setting up a Local ULN Mirror ......................... 17
   2.8.3 ULN Mirror Configuration .............................. 20
   2.8.4 Updating the Repositories on a Local ULN Mirror ....... 20
   2.8.5 Configuring yum on a Local ULN Mirror .............. 21
   2.8.6 Configuring Oracle Linux Yum Clients of a Local ULN Mirror .................................................. 21
2.9 Creating a Local Yum Repository Using an ISO Image ............ 23
2.10 Setting up a Local Yum Server Using an ISO Image .............. 24
2.11 For More Information About Yum ................................ 25

This chapter describes how you can use the yum utility to install and upgrade software packages.

2.1 About Yum

Oracle Linux provides the yum utility which you can use to install or upgrade RPM packages. The main benefit of using yum is that it also installs or upgrades any package dependencies. yum downloads the packages from repositories such as those that are available on the Oracle public yum server, but you can also set up your own repositories on systems that do not have Internet access.

The Oracle public yum server is a convenient way to install Oracle Linux and Oracle VM packages, including bug fixes, security fixes and enhancements, rather than installing them from installation media. You can access the server at http://public-yum.oracle.com/.

You can also subscribe to the Oracle Linux and Oracle VM errata mailing lists to be notified when new packages are released. You can access the mailing lists at https://oss.oracle.com/mailman/listinfo/el-errata and https://oss.oracle.com/mailman/listinfo/oraclevm-errata.

If you have registered your system with the Unbreakable Linux Network (ULN), you can use yum with ULN channels to maintain the software on your system, as described in Chapter 3, The Unbreakable Linux Network.

2.2 Yum Configuration

The main configuration file for yum is /etc/yum.conf. The global definitions for yum are located under the [main] section heading of the yum configuration file. The following table lists the important directives.

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cachedir</td>
<td>Directory used to store downloaded packages.</td>
</tr>
</tbody>
</table>
### Directive | Description
--- | ---
**debuglevel** | Logging level, from 0 (none) to 10 (all).  
**exactarch** | If set to 1, only update packages for the correct architecture.  
**exclude** | A space separated list of packages to exclude from installs or updates, for example: exclude=VirtualBox-4.* kernel*.  
**gpgcheck** | If set to 1, verify the authenticity of the packages by checking the GPG signatures. You might need to set gpgcheck to 0 if a package is unsigned, but you should be wary that the package could have been maliciously altered.  
**gpgkey** | Pathname of the GPG public key file.  
**installonly_limit** | Maximum number of versions that can be installed of any one package.  
**keepcache** | If set to 0, remove packages after installation.  
**logfile** | Pathname of the yum log file.  
**obsoletes** | If set to 1, replace obsolete packages during upgrades.  
**plugins** | If set to 1, enable plugins that extend the functionality of yum.  
**proxy** | URL of a proxy server including the port number. See Section 2.2.1, “Configuring Use of a Proxy Server”.  
**proxy_password** | Password for authentication with a proxy server.  
**proxy_username** | User name for authentication with a proxy server.  
**reposdir** | Directories where yum should look for repository files with a .repo extension.  

The default directory is /etc/yum.repos.d.  

See the yum.conf(5) manual page for more information.

The following listing shows an example [main] section from the yum configuration file.

```ini
[main]
cachedir=/var/cache/yum
keepcache=0
devlevel=2
logfile=/var/log/yum.log
 challengarch=1
obsoletes=1
gpgkey=file://media/RPM-GPG-KEY
gpgcheck=1
plugins=1
installonly_limit=3
```

It is possible to define repositories below the [main] section in /etc/yum.conf or in separate repository configuration files. By default, yum expects any repository configuration files to be located in the /etc/yum.repos.d directory unless you use the reposdir directive to define alternate directories.

### 2.2.1 Configuring Use of a Proxy Server

If your organization uses a proxy server as an intermediary for Internet access, specify the **proxy** setting in /etc/yum.conf as shown in the following example.

```ini
proxy=http://proxysvr.yourdom.com:3128
```

If the proxy server requires authentication, additionally specify the **proxy_username** and **proxy_password** settings.

```ini
proxy=http://proxysvr.yourdom.com:3128
proxy_username=yumacc
```
proxy_password=cliydenw

If you use the yum plugin (yum-rhn-plugin) to access the ULN, specify the enableProxy and httpProxy settings in /etc/sysconfig/rhn/up2date as shown in this example.

enableProxy=1
httpProxy=http://proxysvr.yourdom.com:3128

If the proxy server requires authentication, additionally specify the enableProxyAuth, proxyUser, and proxyPassword settings.

enableProxy=1
httpProxy=http://proxysvr.yourdom.com:3128
enableProxyAuth=1
proxyUser=yumacc
proxyPassword=cliydenw

Caution

All yum users require read access to /etc/yum.conf or /etc/sysconfig/rhn/up2date. If these files must be world-readable, do not use a proxy password that is the same as any user's login password, and especially not root's password.

2.2.2 Yum Repository Configuration

The yum configuration file or yum repository configuration files can contain one or more sections that define repositories.

The following table lists the basic directives for a repository.

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseurl</td>
<td>Location of the repository channel (expressed as a file://, ftp://, http://, or https:// address). This directive must be specified.</td>
</tr>
<tr>
<td>enabled</td>
<td>If set to 1, permit yum to use the channel.</td>
</tr>
<tr>
<td>name</td>
<td>Descriptive name for the repository channel. This directive must be specified.</td>
</tr>
</tbody>
</table>

Any other directive that appears in this section overrides the corresponding global definition in [main] section of the yum configuration file. See the yum.conf(5) manual page for more information.

The following listing shows an example repository section from a configuration file.

[ol6_u2_base]
name=Oracle Linux 6 U2 - $basearch - base
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

In this example, the values of gpgkey and gpgcheck override any global setting. yum substitutes the name of the current system's architecture for the variable $basearch.

2.3 Downloading the Oracle Public Yum Repository Files

Note

The following procedure assumes that yum on your system is configured to expect to find repository files in the default /etc/yum.repos.d directory.

To download the Oracle public yum repository configuration file:
Using Yum from the Command Line

1. As root, change directory to /etc/yum.repos.d.

   # cd /etc/yum.repos.d

2. Use the `wget` utility to download the repository configuration file that is appropriate for your system.


   For Oracle Linux 6, enter:


   The /etc/yum.repos.d directory is updated with the repository configuration file, in this example, `public-yum-ol6.repo`.

3. You can enable or disable repositories in the file by setting the value of the `enabled` directive to 1 or 0 as required.

### 2.4 Using Yum from the Command Line

The following table shows some examples of common tasks that you can perform using `yum`.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>yum repolist</code></td>
<td>Lists all enabled repositories.</td>
</tr>
<tr>
<td><code>yum list</code></td>
<td>Lists all packages that are available in all enabled repositories and all packages that are installed on your system.</td>
</tr>
<tr>
<td><code>yum list installed</code></td>
<td>Lists all packages that are installed on your system.</td>
</tr>
<tr>
<td><code>yum list available</code></td>
<td>Lists all packages that are available to be installed in all enabled repositories.</td>
</tr>
<tr>
<td><code>yum search string</code></td>
<td>Searches the package descriptions for the specified string.</td>
</tr>
<tr>
<td><code>yum provides feature</code></td>
<td>Finds the name of the package to which the specified file or feature belongs. For example:</td>
</tr>
<tr>
<td></td>
<td>yum provides /etc/sysconfig/atd</td>
</tr>
<tr>
<td><code>yum info package</code></td>
<td>Displays detailed information about a package. For example:</td>
</tr>
<tr>
<td></td>
<td>yum info bind</td>
</tr>
<tr>
<td><code>yum install package</code></td>
<td>Installs the specified package, including packages on which it depends. For example:</td>
</tr>
<tr>
<td></td>
<td>yum install ocfs2-tools</td>
</tr>
<tr>
<td><code>yum check-update</code></td>
<td>Checks whether updates exist for packages that are already installed on your system.</td>
</tr>
<tr>
<td><code>yum update package</code></td>
<td>Updates the specified package, including packages on which it depends. For example:</td>
</tr>
<tr>
<td></td>
<td>yum upgrade nfs-utils</td>
</tr>
<tr>
<td><code>yum update</code></td>
<td>Updates all packages, including packages on which they depend.</td>
</tr>
<tr>
<td><code>yum remove package</code></td>
<td>Removes the specified package. For example:</td>
</tr>
<tr>
<td></td>
<td>yum erase nfs-utils</td>
</tr>
</tbody>
</table>
### Command Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yum erase package</td>
<td>Removes the specified package. This command has the same effect as the yum remove command.</td>
</tr>
<tr>
<td>yum update</td>
<td>Updates all packages, including packages on which they depend.</td>
</tr>
<tr>
<td>yum clean all</td>
<td>Removes all cached package downloads and cached headers that contain information about remote packages. Running this command can help to clear problems that can result from unfinished transactions or out-of-date headers.</td>
</tr>
<tr>
<td>yum help</td>
<td>Displays help about yum usage.</td>
</tr>
<tr>
<td>yum help command</td>
<td>Displays help about the specified yum command. For example:</td>
</tr>
<tr>
<td>yum help upgrade</td>
<td></td>
</tr>
<tr>
<td>yum shell</td>
<td>Runs the yum interactive shell.</td>
</tr>
</tbody>
</table>

See the yum(8) manual page for more information.

To list the files in a package, use the repoquery utility, which is included in the yum-utils package. For example, the following command lists the files that the btrfs-progs package provides.

```
# repoquery -l btrfs-progs
/sbin/btrfs
/sbin/btrfs-convert
/sbin/btrfs-debug-tree
.
.
```

**Note**

yum makes no distinction between installing and upgrading a kernel package. yum always installs a new kernel regardless of whether you specify update or install.

### 2.5 Yum Groups

A set of packages can themselves be organized as a yum group. Examples include the groups for Eclipse, fonts, and system administration tools. The following table shows the yum commands that you can use to manage these groups.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yum grouplist</td>
<td>Lists installed groups and groups that are available for installation.</td>
</tr>
<tr>
<td>yum groupinfo groupname</td>
<td>Displays detailed information about a group.</td>
</tr>
<tr>
<td>yum groupinstall groupname</td>
<td>Installs all the packages in a group.</td>
</tr>
<tr>
<td>yum groupupdate groupname</td>
<td>Updates all the packages in a group.</td>
</tr>
<tr>
<td>yum groupremove groupname</td>
<td>Removes all the packages in a group.</td>
</tr>
</tbody>
</table>

### 2.6 Installing and Using the Yum Security Plugin

The yum-plugin-security package allows you to use yum to obtain a list of all of the errata that are available for your system, including security updates. You can also use Oracle Enterprise Manager 12c Cloud Control or management tools such as Katello, Pulp, Red Hat Satellite, Spacewalk, and SUSE Manager to extract and display information about errata.
To install the `yum-plugin-security` package, enter the following command:

```bash
# yum install yum-plugin-security
```

To list the errata that are available for your system, enter:

```bash
# yum updateinfo list
```

The output from the command sorts the available errata in order of their IDs, and it also specifies whether each erratum is a security patch (`severity/Sec.`), a bug fix (`bugfix`), or a feature enhancement (`enhancement`). Security patches are listed by their severity: Important, Moderate, or Low.

You can use the `--sec-severity` option to filter the security errata by severity, for example:

```bash
# yum updateinfo list --sec-severity=Moderate
```

To list the security errata by their Common Vulnerabilities and Exposures (CVE) IDs instead of their errata IDs, specify the keyword `cves` as an argument:

```bash
# yum updateinfo list cves
```

Similarly, the keywords `bugfix`, `enhancement`, and `security` filter the list for all bug fixes, enhancements, and security errata.

You can use the `--cve` option to display the errata that correspond to a specified CVE, for example:

```bash
# yum updateinfo list --cve CVE-2012-2677
```
Installing and Using the Yum Security Plugin

ELSA-2013-0668 Moderate/Sec. boost-devel-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-filesystem-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-iostreams-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-program-options-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-python-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-regexp-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-signals-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-serialization-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-thread-1.41.0-15.el6_4.x86_64
ELSA-2013-0668 Moderate/Sec. boost-wave-1.41.0-15.el6_4.x86_64
updateinfo list done

To display more information, specify info instead of list, for example:

```
# yum updateinfo info --cve CVE-2012-2677
```

To update all packages for which security-related errata are available to the latest versions of the packages, even if those packages include bug fixes or new features but not security errata, enter:

```
# yum --security update
```

To update all packages to the latest versions that contain security errata, ignoring any newer packages that do not contain security errata, enter:

```
# yum --security update-minimal
```

To update all kernel packages to the latest versions that contain security errata, enter:

```
# yum --security update-minimal kernel*
```

You can also update only those packages that correspond to a CVE or erratum, for example:

```
# yum update --cve CVE-2012-3954
# yum update --advisory ELSA-2012-1141
```
Note
Some updates might require you to reboot the system. By default, the boot manager will automatically enable the most recent kernel version.

For more information, see the `yum-security(8)` manual page.

2.7 Switching CentOS or Scientific Linux Systems to Use the Oracle Public Yum Server

You can use the `centos2ol.sh` script to convert CentOS 5 and 6 or Scientific Linux 5 and 6 systems to Oracle Linux. The script configures `yum` to use the Oracle’s public yum server and installs a few additional packages that are required. There is no need to reboot the system.

To perform the switch to Oracle Linux, run the following commands as `root`:

```
# curl -O https://linux.oracle.com/switch/centos2ol.sh
# sh centos2ol.sh
```

For more information, see https://linux.oracle.com/switch/centos/.

2.8 Creating and Using a Local ULN Mirror

The following sections describe how to create and use a yum server that acts as a local mirror of the ULN channels.

2.8.1 Prerequisites for the Local ULN Mirror

The system that you want to set up as a local ULN mirror must meet the following criteria:

- You must have registered the system with ULN. See The Unbreakable Linux Network.
- The system must be running Oracle Linux 5, Oracle Linux 6, or Oracle Linux 7.
- The system must have a least 6 GB of memory to create the yum metadata.
- The system must have enough disk space to store copies of the packages that it hosts. The following table shows the approximate amount of space that is required for Oracle Linux channels:

<table>
<thead>
<tr>
<th>Oracle Linux Channel</th>
<th>Space Required per Channel for Binaries Only</th>
<th>Space Required per Channel for Both Binaries and Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>[oe]1*_latest</td>
<td>Up to 10 GB</td>
<td>Up to 15 GB</td>
</tr>
<tr>
<td>[oe]1*_addons</td>
<td>600 MB</td>
<td>1 GB</td>
</tr>
<tr>
<td>[oe]1*_oracle</td>
<td>1 GB</td>
<td>Not applicable</td>
</tr>
<tr>
<td>[oe]1*_base</td>
<td>3 GB</td>
<td>5.5 GB</td>
</tr>
<tr>
<td>[oe]1*_patch</td>
<td>1 GB</td>
<td>2 GB</td>
</tr>
</tbody>
</table>

The next table shows the approximate amount of space that is required for Oracle VM channels:

<table>
<thead>
<tr>
<th>Oracle VM Channel</th>
<th>Space Required per Channel for Binaries Only</th>
<th>Space Required per Channel for Both Binaries and Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>ovm*_latest</td>
<td>500 MB</td>
<td>1 GB</td>
</tr>
<tr>
<td>ovm*_base</td>
<td>400 MB</td>
<td>800 MB</td>
</tr>
<tr>
<td>ovm*_patch</td>
<td>100 MB</td>
<td>200 MB</td>
</tr>
</tbody>
</table>
2.8.2 Setting up a Local ULN Mirror

To set up a local system as a local ULN mirror:

1. Using a browser, log in at http://linux.oracle.com with the ULN user name and password that you used to register the system, and configure its properties on ULN as follows:

   a. On the Systems tab, click the link named for your system in the list of registered machines.
   
   b. On the System Details page, click **Edit**.
   
   c. On the Edit System Properties page, select the **Yum Server** check box and click **Apply Changes**.
   
   d. On the System Details page, click **Manage Subscriptions**.
   
   e. On the System Summary page, select channels from the list of available or subscribed channels and click the arrows to move the channels between the lists.

   Modify the list of subscribed channels to include the channels that you want to make available to local systems.

   ![Note]

   You must subscribe the system to the **latest** and **addons** channels for the installed operating system release (Oracle Linux 5, Oracle Linux 6, or Oracle Linux 7) and the system architecture (i386 or x86-64) to be able to install the **yum-uln_mirror** package. This package contains the **uln-yum-mirror** script that enables the system to act as a local ULN mirror.

   For example, the following table shows some examples of the channels that are available for Oracle Linux 6 on the x86_64 architecture.

```
<table>
<thead>
<tr>
<th>Channel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ol6_ga_x86_64_base</td>
<td>All packages for Oracle Linux 6 as initially released. This channel does not include errata.</td>
</tr>
<tr>
<td>ol6_x86_64_addons</td>
<td>Oracle Linux 6 add ons, including the <strong>yum-uln_mirror</strong> package.</td>
</tr>
<tr>
<td>ol6_x86_64_ksplice</td>
<td>Oracle Ksplice clients, updates, and dependencies for Oracle Linux 6. Note that access to this channel requires an Oracle Linux Premier Support account.</td>
</tr>
<tr>
<td>ol6_x86_64_latest</td>
<td>All packages released for Oracle Linux 6, including the latest errata packages.</td>
</tr>
<tr>
<td>ol6_x86_64_UEK_latest</td>
<td>Latest Unbreakable Enterprise Kernel Release 2 packages for Oracle Linux 6.</td>
</tr>
<tr>
<td>ol6_x86_64_UEK3_latest</td>
<td>Latest Unbreakable Enterprise Kernel Release 3 packages for Oracle Linux 6.</td>
</tr>
</tbody>
</table>
```

If you subsequently update the list of channels to which the system is subscribed, the **uln-yum-mirror** script updates the channels that the system mirrors. If you want to be able to use **yum** to update the server from the repositories that it hosts rather than from ULN, follow the procedure in Section 2.8.5, “Configuring yum on a Local ULN Mirror”. 

---

Note

You must subscribe the system to the **latest** and **addons** channels for the installed operating system release (Oracle Linux 5, Oracle Linux 6, or Oracle Linux 7) and the system architecture (i386 or x86-64) to be able to install the **yum-uln_mirror** package. This package contains the **uln-yum-mirror** script that enables the system to act as a local ULN mirror.

For example, the following table shows some examples of the channels that are available for Oracle Linux 6 on the x86_64 architecture.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ol6_ga_x86_64_base</td>
<td>All packages for Oracle Linux 6 as initially released. This channel does not include errata.</td>
</tr>
<tr>
<td>ol6_x86_64_addons</td>
<td>Oracle Linux 6 add ons, including the <strong>yum-uln_mirror</strong> package.</td>
</tr>
<tr>
<td>ol6_x86_64_ksplice</td>
<td>Oracle Ksplice clients, updates, and dependencies for Oracle Linux 6. Note that access to this channel requires an Oracle Linux Premier Support account.</td>
</tr>
<tr>
<td>ol6_x86_64_latest</td>
<td>All packages released for Oracle Linux 6, including the latest errata packages.</td>
</tr>
<tr>
<td>ol6_x86_64_UEK_latest</td>
<td>Latest Unbreakable Enterprise Kernel Release 2 packages for Oracle Linux 6.</td>
</tr>
<tr>
<td>ol6_x86_64_UEK3_latest</td>
<td>Latest Unbreakable Enterprise Kernel Release 3 packages for Oracle Linux 6.</td>
</tr>
</tbody>
</table>

If you subsequently update the list of channels to which the system is subscribed, the **uln-yum-mirror** script updates the channels that the system mirrors. If you want to be able to use **yum** to update the server from the repositories that it hosts rather than from ULN, follow the procedure in Section 2.8.5, “Configuring yum on a Local ULN Mirror”.

---
Setting up a Local ULN Mirror

If you have an Oracle Linux Premier Support account and you want the yum server to host Ksplice packages for local Ksplice offline clients, subscribe to the Ksplice for Oracle Linux channels for the architectures and Oracle Linux releases that you want to support.

For a complete and up-to-date list of the available release channels, log on to ULN at http://linux.oracle.com.

f. When you have finished selecting channels, click Save Subscriptions and log out of ULN.

2. Install the Apache HTTP server.

```bash
# yum install httpd
```

3. Create a base directory for the yum repositories, for example /var/yum or /var/www/html/yum.

```bash
# mkdir -p /var/www/html/yum
```

Note

The yum repository owner must have read and write permissions on this directory.

4. If you created a base directory for the yum repository that is not under /var/www/html and SELinux is enabled in enforcing mode on your system:

   a. Use the semanage command to define the default file type of the repository root directory hierarchy as httpd_sys_content_t:

   ```bash
   # /usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/var/yum(/.*)?"
   ```

   b. Use the restorecon command to apply the file type to the entire repository.

   ```bash
   # /sbin/restorecon -R -v /var/yum
   ```

5. If you created a base directory for the yum repository that is not under /var/www/html, create a symbolic link in /var/www/html that points to the repository, for example:

```bash
# ln -s /var/yum /var/www/html/yum
```

6. Edit the HTTP server configuration file, /etc/httpd/conf/httpd.conf, as follows:

   a. Specify the resolvable domain name of the server in the argument to ServerName.

   ```bash
   ServerName server_addr:80
   ```

   If the server does not have a resolvable domain name, enter its IP address instead.

   b. Verify that the setting of the Options directive in the <Directory "/var/www/html"> section specifies Indexes and FollowSymLinks to allow you to browse the directory hierarchy, for example:

   ```bash
   Options Indexes FollowSymLinks
   ```

   c. Save your changes to the file.

7. Start the HTTP server, and configure it to start after a reboot.

   • On Oracle Linux 5 or Oracle Linux 6, enter the following commands:
Setting up a Local ULN Mirror

# service httpd start
# chkconfig httpd on

- On Oracle Linux 7, enter the following commands:

  # systemctl start httpd
  # systemctl enable httpd

8. If you have enabled a firewall on your system, configure it to allow incoming HTTP connection requests on TCP port 80.

- On Oracle Linux 5 or Oracle Linux 6, enter the following commands:

  # iptables -I INPUT -p tcp --state NEW -m tcp --dport 80 -j ACCEPT
  # service iptables save

- On Oracle Linux 7, enter the following commands:

  # firewall-cmd --add-service=http
  # firewall-cmd --permanent --add-service=http

9. Install the **uln-yum-mirror** package:

   # yum install uln-yum-mirror

   This package contains the **uln-yum-mirror** script that enables the system to act as a local ULN mirror.

   **Note**

   If you have not subscribed the system to the correct Oracle Linux latest and addons channels for your system, the command fails with the error **No package uln-yum-mirror available**.

10. To configure the operation of the **/usr/bin/uln-yum-mirror** script, edit the **/etc/sysconfig/uln-yum-mirror** file.

    For example, if the base directory for the yum repositories is not **/var/www/html/yum**, set the value of the **REP_BASE** parameter to the correct base directory:

    ```
    REP_BASE=/var/yum
    ```

    Installing the **uln-yum-mirror** package also configures an **anacron** job (**/etc/cron.daily/uln-yum-mirror**) that updates the local yum repositories once every day. You can disable this job by setting the value of **CRON_ENABLED** to 0:

    ```
    CRON_ENABLED=0
    ```

    For more information about the configuration options in **/etc/sysconfig/uln-yum-mirror** file, see **Section 2.8.3, “ULN Mirror Configuration”**.

    The repositories are populated when the **anacron** job runs the **/usr/bin/uln-yum-mirror** script. Alternatively, you can run the script manually at any time to update the repositories. See **Section 2.8.4, “Updating the Repositories on a Local ULN Mirror”**.
2.8.3 ULN Mirror Configuration

The `/etc/sysconfig/uln-yum-mirror` file contains the following configuration parameters that affect the behavior of the `/usr/bin/uln-yum-mirror` script:

- **ALL_PKGS**
  Specifies whether `uln-yum-mirror` mirrors all versions of every available package or downloads only the latest version of each package. The default value of 1 causes `uln-yum-mirror` to mirror all versions of every available package. A value of 0 causes `uln-yum-mirror` to download only the latest version of each package.

- **CRON_ENABLED**
  Specifies whether `uln-yum-mirror` runs automatically once per day. The default value of 1 enables `uln-yum-mirror` to be run automatically as an `anacron` job. A value of 0 disables the job. You must run `uln-yum-mirror` manually to update the packages.

- **HARDLINK_RPMS**
  Specifies whether `uln-yum-mirror` runs `hardlinkpy` to create hard links between identical RPMs after the mirror process finishes. The default value of 1 enables hard linking, which saves storage space. It is not possible to create hard links across file systems. Set the value to 0 if the repository storage spans more than one file system.

- **LOG_OUTPUT**
  Specifies whether `uln-yum-mirror` logs its output. The default value of 1 enables logging. A value of 0 disables logging.

- **REP_BASE**
  Specifies the base directory for the repositories. The default setting is `/var/www/html/yum`. Do not change this setting unless you customize the configuration of the HTTP server.

- **REP_EL, REP_ENG, REP_OLM, REP_OVM, REP_UEM**
  Specify the names of the repositories. If required, you can configure alternate names.

- **REPO_FILE_DIR**
  Not currently used.

- **SRC**
  Specifies whether `uln-yum-mirror` mirrors source RPMs in addition to binary RPMs. The default value of 0 prevents `uln-yum-mirror` from mirroring source RPMs. A value to 1 causes `uln-yum-mirror` to mirror source RPMs.

- **YUM_GLOBAL_CACHE**
  Specifies the yum global cache directory. The default setting is `/var/cache/yum`. Do not change this setting unless you customize the configuration of the HTTP server.

2.8.4 Updating the Repositories on a Local ULN Mirror

To update the repositories for the subscribed channels immediately without waiting for the `anacron` job to run or if you have disabled the job, enter the following command on the local ULN mirror server:

```
# /usr/bin/uln-yum-mirror
```

**Note**
If you have not yet set up the contents of the repositories, it can take many hours to download all the packages.
2.8.5 Configuring yum on a Local ULN Mirror

The following procedure configures the `yum` command on a server that is acting as a local ULN mirror to install package updates from itself rather than from ULN. The procedure does not affect the operation of the `uln-yum-mirror` script.

To configure a server that is acting as a local ULN Mirror to be able to install updated packages from itself:

1. Use the following command to list the channels that the server is mirroring from ULN:

```
# yum repolist
Loaded plugins: rhnplugin, security
This system is receiving updates from ULN.
0 packages excluded due to repository protections
```

<table>
<thead>
<tr>
<th>repo id</th>
<th>repo name</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ol6_addons</td>
<td>Oracle Linux 6 Server Add ons (x86_64)</td>
<td>112</td>
</tr>
<tr>
<td>ol6_x86_64_latest</td>
<td>Oracle Linux 6 Latest (x86_64)</td>
<td>17,976</td>
</tr>
<tr>
<td>ol6_x86_64_UEKR3_latest</td>
<td>Latest Unbreakable Enterprise Kernel Release 3 for Oracle Linux 6 (x86_64)</td>
<td>41</td>
</tr>
</tbody>
</table>

In this example, the server mirrors the `ol6_addons`, `ol6_x86_64_latest`, and `ol6_x86_64_UEKR3_latest` channels from ULN.

2. Edit `/etc/yum/pluginconf.d/rhnplugin.conf` and disable the mirrored channels by adding the following stanza for each channel:

```
[repo_id]
enabled=0
```

For example, to disable the `ol6_addons`, `ol6_x86_64_latest`, and `ol6_x86_64_UEKR3_latest` channels, you would add the following stanzas:

```
[ol6_addons]
enabled=0

[ol6_x86_64_latest]
enabled=0

[ol6_x86_64_UEKR3_latest]
enabled=0
```

3. Configure the server as a yum client as described in Section 2.8.6, “Configuring Oracle Linux Yum Clients of a Local ULN Mirror”.

2.8.6 Configuring Oracle Linux Yum Clients of a Local ULN Mirror

If you have set up a local ULN mirror, you can configure your local Oracle Linux systems to receive yum updates from that server.

To configure an Oracle Linux system as a yum client:

1. Import the GPG key:

```
# rpm --import /usr/share/rhn/RPM-GPG_KEY
```
2. In the `/etc/yum.repos.d` directory, edit the existing repository file, such as `public-yum-ol6.repo` or `ULN-base.repo`, and disable all entries by setting `enabled=0`.

3. In the `/etc/yum.repos.d` directory, create the file `local-yum.repo`, which contains entries such as the following for an Oracle Linux 6 yum client:

   ```
   [local_ol6_latest]
   name=Oracle Linux $releasever - $basearch - latest
   baseurl=http://local_uln_mirror/yum/OracleLinux/OL6/latest/$basearch/
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
   gpgcheck=1
   enabled=1

   [local_ol6_UEKR3_latest]
   name=Unbreakable Enterprise Kernel Release 3 for Oracle Linux $releasever - $basearch - latest
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
   gpgcheck=1
   enabled=1

   [local_ol6_addons]
   name=Oracle Linux $releasever - $basearch - addons
   baseurl=http://local_uln_mirror/yum/OracleLinux/OL6/addons/$basearch/
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
   gpgcheck=1
   enabled=1
   ```

   To distinguish the local repositories from the ULN repositories, prefix the names of their entries with a string such as `local_`. Replace `local_uln_mirror` with the IP address or resolvable host name of the local ULN mirror.

   The example configuration enables the `local_ol6_latest`, `local_ol6_UEKR3_latest`, and `local_ol6_addons` channels.

4. To test the configuration:
   a. Clear the yum metadata cache:
      ```
      # yum clean metadata
      ```
   b. Use `yum repolist` to verify the configuration, for example:
      ```
      # yum repolist
      Loaded plugins: rhnplugin, security
      This system is receiving updates from ULN.
      0 packages excluded due to repository protections
      repo id                         repo name                                    status
      local_ol6_addons                Oracle Linux 6 - x86_64 - latest             112
      local_ol6_x86_64_latest         Oracle Linux 6 - x86_64 - latest             17,976
      local_ol6_x86_64_UEKR3_latest   Unbreakable Enterprise Kernel Release 3 for Oracle Linux 6 - x86_64 - latest 41
      ```
      If `yum` cannot connect to the local ULN mirror, check that the firewall settings on the local ULN mirror server allow incoming TCP connections to the HTTP port (usually, port 80).

5. You can now run `yum update` to pick up new updates from the local ULN mirror.
2.9 Creating a Local Yum Repository Using an ISO Image

Note
The system must have sufficient storage space to host a full Oracle Linux Media Pack DVD image (approximately 3.5 GB for Oracle Linux Release 6 Update 3).

To create a local yum repository (for example, if a system does not have Internet access):

1. On a system with Internet access, download a full Oracle Linux DVD image from the Oracle Software Delivery Cloud at http://edelivery.oracle.com/linux onto removable storage (such as a USB memory stick). For example, V33411-01.iso contains the Oracle Linux Release 6 Update 3 Media Pack for x86 (64 bit).

Note
You can verify that the ISO was copied correctly by comparing its checksum with the digest value that is listed on edelivery.oracle.com, for example:

```
# sha1sum V33411-01.iso
7daae91cc0437f6a98a4359ad97066d78a9f19de V33411-01.iso
```

2. Transfer the removable storage to the system on which you want to create a local yum repository, and copy the DVD image to a directory in a local file system.

```
# cp /media/USB_stick/V33411-01.iso /ISOs
```

3. Create a suitable mount point, for example /var/OSimage/OL6.3_x86_64, and mount the DVD image on it.

```
# mkdir -p /var/OSimage/OL6.3_x86_64
# mount -o loop,ro /ISOs/V33411-01.iso /var/OSimage/OL6.3_x86_64
```

Note
Include the read-only mount option (ro) to avoid changing the contents of the ISO by mistake.

4. Create an entry in /etc/fstab so that the system always mounts the DVD image after a reboot.

```
/ISOs/V33411-01.iso /var/OSimage/OL6.3_x86_64 iso9660 loop,ro 0 0
```

5. In the /etc/yum.repos.d directory, edit the existing repository files, such as public-yum-ol6.repo or ULN-base.repo, and disable all entries by setting enabled=0.

6. Create the following entries in a new repository file (for example, /etc/yum.repos.d/OL63.repo).

```
[OL63]
name=Oracle Linux 6.3 x86_64
baseurl=file:///var/OSimage/OL6.3_x86_64
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1
```

7. Clean up the yum cache.

```
# yum clean all
```

8. Test that you can use yum to access the repository.

```
# yum repolist
```
2.10 Setting up a Local Yum Server Using an ISO Image

To set up a local yum server (for example, if you have a network of systems that do not have Internet access):

1. Choose one of the systems to be the yum server, and create a local yum repository on it as described in Section 2.9, “Creating a Local Yum Repository Using an ISO Image”.

2. Install the Apache HTTP server from the local yum repository.

   `# yum install httpd`

3. If SELinux is enabled in enforcing mode on your system:
   a. Use the `semanage` command to define the default file type of the repository root directory hierarchy as `httpd_sys_content_t`:

      ```bash
      # /usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/var/OSimage(/.*)?"
      ```

   b. Use the `restorecon` command to apply the file type to the entire repository.

      ```bash
      # /sbin/restorecon -R -v /var/OSimage
      ```

   Note
   The `semanage` and `restorecon` commands are provided by the `policycoreutils-python` and `policycoreutils` packages.

4. Create a symbolic link in `/var/www/html` that points to the repository:

   ```bash
   # ln -s /var/OSimage /var/www/html/OSimage
   ```

5. Edit the HTTP server configuration file, `/etc/httpd/conf/httpd.conf`, as follows:
   a. Specify the resolvable domain name of the server in the argument to `ServerName`.

      ```
      ServerName server_addr:80
      ```

      If the server does not have a resolvable domain name, enter its IP address instead.

   b. Verify that the setting of the `Options` directive in the `<Directory "/var/www/html">` section specifies `Indexes` and `FollowSymLinks` to allow you to browse the directory hierarchy, for example:

      ```
      Options Indexes FollowSymLinks
      ```

   c. Save your changes to the file.

6. Start the Apache HTTP server, and configure it to start after a reboot.

   ```bash
   # service httpd start
   # chkconfig httpd on
   ```

7. If you have enabled a firewall on your system, configure it to allow incoming HTTP connection requests on TCP port 80.
For example, the following command configures `iptables` to allow incoming HTTP connection requests and saves the change to the firewall configuration:

```
# iptables -I INPUT -p tcp --state NEW -m tcp --dport 80 -j ACCEPT
# service iptables save
```

8. Edit the repository file on the server (for example, `/etc/yum.repos.d/OL63.repo`):

```
[OL63]
name=Oracle Linux 6.3 x86_64
baseurl=http://server_addr/OSimage/OL6.3_x86_64
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1
```

Replace `server_addr` with the IP address or resolvable host name of the local yum server.

9. On each client, copy the repository file from the server to the `/etc/yum.repos.d` directory.

10. In the `/etc/yum.repos.d` directory, edit any other repository files, such as `public-yum-ol6.repo` or `ULN-base.repo`, and disable all entries by setting `enabled=0`.

11. On the server and each client, test that you can use `yum` to access the repository.

```
# yum repolist
```

2.11 For More Information About Yum

For more information about yum, see `http://yum.baseurl.org/`.

For more information about how to download the latest packages from the Unbreakable Linux Network and make the packages available through a local yum server, see `http://www.oracle.com/technetwork/articles/servers-storage-admin/yum-repo-setup-1659167.html`.
Chapter 3 The Unbreakable Linux Network

Table of Contents

3.1 About the Unbreakable Linux Network ................................................................. 27
3.2 About ULN Channels ......................................................................................... 27
3.3 About Software Errata .................................................................................... 29
3.4 Registering as a ULN User ................................................................................ 29
3.5 Registering an Oracle Linux 6 or Oracle Linux 7 System ................................. 30
3.6 Registering an Oracle Linux 4 or Oracle Linux 5 System ................................. 30
3.7 Configuring an Oracle Linux 5 System to Use yum with ULN ......................... 30
3.8 Disabling Package Updates .............................................................................. 31
3.9 Subscribing Your System to ULN Channels ..................................................... 31
3.10 Browsing and Downloading Errata Packages ............................................... 32
3.11 Downloading Available Errata for a System ................................................... 32
3.12 Updating System Details ............................................................................... 32
3.13 Deleting a System ......................................................................................... 33
3.14 About CSI Administration ............................................................................. 33
   3.14.1 Becoming a CSI Administrator .................................................................. 34
   3.14.2 Listing Active CSIs and Transferring Their Registered Servers ............... 35
   3.14.3 Listing Expired CSIs and Transferring Their Registered Servers .............. 36
   3.14.4 Removing a CSI Administrator ................................................................ 37
3.15 Switching from RHN to ULN .......................................................................... 37
3.16 For More Information About ULN ................................................................. 38

This chapter describes how to access and use the software channels that are available on the Unbreakable Linux Network (ULN).

3.1 About the Unbreakable Linux Network

If you have a subscription to Oracle Unbreakable Linux support, you can use the comprehensive resources of the Unbreakable Linux Network (ULN). ULN offers software patches, updates, and fixes for Oracle Linux and Oracle VM, as well as information on yum, Ksplice, and support policies. You can also download useful packages that are not included in the original distribution. The ULN Alert Notification Tool periodically checks with ULN and alerts you when updates are available. You can access ULN at https://linux.oracle.com/, where you will also find instructions for registering with ULN, for creating local yum repositories, and for switching from the Red Hat Network (RHN) to ULN.

If you want to use yum with ULN to manage your systems, you must register the systems with ULN and subscribe each system to one or more ULN channels. When you register a system with ULN, the channel that contains the latest version is chosen automatically according to the architecture and operating system revision of the system.

When you run yum, it connects to the ULN server repository and downloads the latest software packages in RPM format onto your system. yum then presents you with a list of the available packages so that you can choose which ones you want to install.

3.2 About ULN Channels

ULN provides more than 100 unique channels, which support the i386, x86_64, and ia64 architectures, for releases of Oracle Linux 4 update 6 and later.
You can choose for your system to remain at a specific OS revision, or you can allow the system to be updated with packages from later revisions.

You should subscribe to the channel that corresponds to the architecture of your system and the update level at which you want to maintain it. Patches and errata are available for specific revisions of Oracle Linux, but you do not need to upgrade from a given revision level to install these fixes. ULN channels also exist for MySQL, Oracle VM, OCFS2, RDS, and productivity applications.

The following table describes the main channels that are available.

<table>
<thead>
<tr>
<th>Channel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_latest</td>
<td>Provides all the packages in a distribution, including any errata that are also provided in the patch channel. Unless you explicitly specify the version, any package that you download on this channel will be the most recent that is available. If no vulnerabilities have been found in a package, the package version might be the same as that included in the original distribution. For other packages, the version will be the same as that provided in the patch channel for the highest update level. For example, the ol6_arch_latest channel for Oracle Linux 6 Update 3 contains a combination of the ol6_u3_arch_base and ol6_u3_arch_patch channels.</td>
</tr>
<tr>
<td>_base</td>
<td>Provides the packages for each major version and minor update of Oracle Linux and Oracle VM. This channel corresponds to the released ISO media image. For example, there is a base channel for each of the updates to Oracle Linux 6 as well as for Oracle Linux 6. Oracle does not publish security errata and bugfixes on these channels.</td>
</tr>
<tr>
<td>_patch</td>
<td>Provides only those packages that have changed since the initial release of a major or minor version of Oracle Linux or Oracle VM. The patch channel always provides the most recent version of a package, including all fixes that have been provided since the initial version was released.</td>
</tr>
<tr>
<td>_addons</td>
<td>Provides packages that are not included in the base distribution, such as the package that you can use to create a yum repository on Oracle Linux 6.</td>
</tr>
<tr>
<td>_oracle</td>
<td>Provides freely downloadable RPMs from Oracle that you can install on Oracle Linux, such as ASMLib and Oracle Instant Client.</td>
</tr>
<tr>
<td>_optional</td>
<td>Provides optional packages for Oracle Linux 7 that have been sourced from upstream. This channel includes most development packages (*-devel).</td>
</tr>
</tbody>
</table>

Other channels may also be available, such as _beta channels for the beta versions of packages.

As each new major version or minor update of Oracle Linux becomes available, Oracle creates new base and patch channels for each supported architecture to distribute the new packages. The existing base and patch channels for the previous versions or updates remain available and do not include the new packages. The _latest channel distributes the highest possible version of any package, and tracks the top of the development tree independently of the update level.

Caution

You can choose to maintain your system at a specific update level of Oracle Linux and selectively apply errata to that level by subscribing the system to the _base and _patch channels and unsubscribing it from the _latest channel. However, for Oracle Linux 7, patches are not added to the _patch channel for previous updates after a new update has been released. For example, after the release of Oracle Linux 7 Update 1, no further errata will be released on the ol7_x86_64_u0_patch channel.
Oracle recommends that you keep your system subscribed to the _latest channel. If you unsubscribe from the _latest channel, your system will become vulnerable to security-related issues when a new update is released.

3.3 About Software Errata

Oracle releases important changes to Oracle Linux and Oracle VM software as individual package updates known as errata, which are made available for download on ULN before they are gathered into a release or are distributed via the _patch channel.

Errata packages can contain:

- Security advisories, which have names prefixed by ELSA-* (for Oracle Linux) and OVMSA-* (for Oracle VM).
- Bug fix advisories, which have names prefixed by ELBA-* and OVMBA-*.
- Feature enhancement advisories, which have names prefixed by ELEA-* and OVMEA-*.

To be notified when new errata packages are released, you can subscribe to the Oracle Linux and Oracle VM errata mailing lists at https://oss.oracle.com/mailman/listinfo/el-errata and https://oss.oracle.com/mailman/listinfo/oraclevm-errata.

If you are logged into ULN, you can also subscribe to these mailing lists by following the Subscribe to Enterprise Linux Errata mailing list and Subscribe to Oracle VM Errata mailing list links that are provided on the Errata tab.

3.4 Registering as a ULN User

When you register a system with ULN, your Oracle Single Signon (SSO) user name is also registered as your ULN user name. If you want to use ULN without first registering a system, you can register as a ULN user provided that you have a valid customer support identifier (CSI) for Oracle Linux support or Oracle VM support. To purchase Oracle Linux or Oracle VM support, go to the online Oracle Linux Store or contact your sales representative.

To register as a ULN user:

1. In a browser, go to https://linux.oracle.com/register.
2. If you do not have an SSO account, click Create New Single Signon Account and follow the onscreen instructions to create one.
   
   If you already have an SSO account, click Sign On.
3. Log in using your SSO user name and password.
4. On the Create New ULN User page, enter your CSI and click Create New User.

Note

If no administrator is currently assigned to manage the CSI, you are prompted to click Confirm to become the CSI administrator. If you click Cancel, you cannot access the CSI administration feature. See Section 3.14, “About CSI Administration”.

If your user name already exists on the system, you are prompted to proceed to ULN by clicking the link Unbreakable Linux Network. If you enter a different
3.5 Registering an Oracle Linux 6 or Oracle Linux 7 System

To register an Oracle Linux 6 or Oracle Linux 7 system with ULN.

1. Run the `uln_register` command.

```
# uln_register
```

Alternatively, if you use the GNOME graphical user desktop, select System > Administration > ULN Registration on Oracle Linux 6 or Applications > System Tools > ULN Registration on Oracle Linux 7. You can also register your system with ULN if you configure networking when installing Oracle Linux 6 or Oracle Linux 7.

2. When prompted, enter your ULN user name, password, and customer support identifier (CSI).

3. Enter a name for the system that will allow you to identify it on ULN, and choose whether to upload hardware and software profile data that allows ULN to select the appropriate packages for the system.

4. If you have an Oracle Linux Premier Support account, you can choose to configure an Oracle Linux 6 or Oracle Linux 7 system that is running a supported kernel to receive kernel updates from Oracle Ksplice. See Section 4.2, “Registering to Use Ksplice Uptrack”.

The `yum-rhn-plugin` is enabled and your system is subscribed to the appropriate software channels.

If you use a proxy server for Internet access, see Section 2.2.1, “Configuring Use of a Proxy Server”.

3.6 Registering an Oracle Linux 4 or Oracle Linux 5 System

To register an Oracle Linux 4 or Oracle Linux 5 system with ULN.

1. Import the RPM GPG key.

```
# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY
```

2. Run the text-mode version of the `up2date` command.

```
# up2date-nox --register
```

3. When prompted, enter your ULN user name, password, and CSI.

4. Enter the name of the system that will be displayed on ULN, and choose whether to upload hardware and software profile data that will allow ULN to select the appropriate packages for your system.

3.7 Configuring an Oracle Linux 5 System to Use yum with ULN

If your Oracle Linux 5 system is registered with ULN, you can use `yum` instead of `up2date` to download and install packages. If you have installed a full update since Oracle Linux 5 Update 6 was released on January 20, 2010, your system should already be able to use `yum` with ULN.

To enable `yum` support:

1. Install `yum-rhn-plugin`.

```
# up2date --install yum-rhn-plugin
```
Disabling Package Updates

2. If your organization uses a proxy server as an intermediary for Internet access, specify the **enableProxy** and **httpProxy** settings in `/etc/sysconfig/rhn/up2date` as shown in this example.

   ```
   enableProxy=1
   httpProxy=http://proxysvr.yourdom.com:3128
   ```

   If the proxy server requires authentication, additionally specify the **enableProxyAuth**, **proxyUser**, and **proxyPassword** settings:

   ```
   enableProxy=1
   enableProxyAuth=1
   httpProxy=http://proxysvr.yourdom.com:3128
   proxyUser=yumacc
   proxyPassword=clydenw
   ```

   **Caution**

   All **yum** users require read access to `/etc/sysconfig/rhn/up2date`. If this file must be world-readable, do not use a password that is the same as any user's login password, and especially not **root**'s password.

With the plugin installed, you can immediately start to use **yum** instead of **up2date**.

### 3.8 Disabling Package Updates

To disable package updates by ULN (for example, if you have deleted your system from ULN), edit the `/etc/yum/pluginconf.d/rhnplugin.conf` file, and change the value of `enabled` flag from 1 to 0 in the `[main]` section, for example:

```
[main]
enabled = 0
```

To disable updates for particular packages, add an `exclude` statement to the `[main]` section of the `/etc/yum.conf` file. For example, to exclude updates for **VirtualBox** and **kernel**:

```
exclude=VirtualBox* kernel*
```

**Note**

Excluding certain packages from being updated can cause dependency errors for other packages. Your machine might also become vulnerable to security-related issues if you do not install the latest updates.

### 3.9 Subscribing Your System to ULN Channels

If you have registered your system with ULN, you can subscribe the system to the channels that are available for the level of support associated with the CSI.

To subscribe your system to ULN channels:

1. Log in to [http://linux.oracle.com](http://linux.oracle.com) with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select channels from the list of available or subscribed channels and click the arrows to move the channels between the lists.
5. When you have finished selecting channels, click **Save Subscriptions**.

### 3.10 Browsing and Downloading Errata Packages

You can browse the advisories that are available on ULN, and download the errata RPMs for the supported combinations of the software release and the system architecture.

To browse the advisories and download errata RPMs:

1. Log in to [http://linux.oracle.com](http://linux.oracle.com) with your ULN user name and password.
2. Select the Errata tab.

   The Errata page displays a table of the available errata for all releases that are available on ULN.

3. On the Errata page, you can perform the following actions on the displayed errata:

   - To sort the table of available errata, click the title of the **Type**, **Severity**, **Advisory**, **Systems Affected**, or **Release Date** column. Click the title again to reverse the order of sorting.

     **Note**

     The **Systems Affected** column shows how many of your systems are potentially affected by an advisory.

   - To display or hide advisories of different types, select or deselect the **Bug**, **Enhancement**, and **Security** check boxes and click **Go**.

   - To display only advisories for a certain release of Oracle Linux or Oracle VM, select that release from the **Release** drop-down list and click **Go**.

   - To search within the table, enter a string in the **Search** field and click **Go**.

4. To see more detail about an advisory and to download the RPMs:

   a. Click the link for the advisory.

   b. On the Errata Detail page for an advisory, you can download the RPMs for the supported releases and system architectures. The **Superseded By Advisory** column displays a link to the most recent advisory (if any) that replaces the advisory you are browsing.

### 3.11 Downloading Available Errata for a System

You can download a comma-separated values (CSV) report file of the errata that are available for your system and you can download errata RPMs.

To download a CSV report or the errata RPMs:

1. Log in to [http://linux.oracle.com](http://linux.oracle.com) with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.

   The System Details page lists the available errata for the system in the Available Errata table, which might be split over several pages.

3. To download the CSV report file, click the link **Download All Available Errata for this System**.
4. To see more detail about an advisory and download the RPMs:
a. Click the link for the advisory.

b. On the System Errata Detail page for an advisory, you can download the RPMs for the affected releases and system architectures.

### 3.12 Updating System Details

If you have registered your system with ULN, you can update the details that ULN records for the system.

To update the details for your system:

1. Log in to [http://linux.oracle.com](http://linux.oracle.com) with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Edit**.
4. On the Edit System Properties page, you can change the name associated with your system, register it as a local yum server for your site, or change the CSI with which it is registered.

   **Note**
   
   You cannot change the CSI of a system unless it is registered to your user name.

5. When you have finished making changes, click **Apply Changes**.

### 3.13 Deleting a System

To delete a system that is registered on ULN:

1. Log in to [http://linux.oracle.com](http://linux.oracle.com) with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Delete**.

   **Note**
   
   You cannot delete a system unless it is registered to your user name.

4. When prompted to confirm the deletion, click **OK**.

### 3.14 About CSI Administration

The CSI administration feature of ULN provides a unified view of all of your organization's CSIs and the systems that are registered with those CSIs. To be able to manage the registered systems, you must become an administrator for one or more of your organization's CSIs. To be able to view and change the details of any system that is not registered to your ULN user name, you must become an administrator for the CSI under which that system is registered.

If you are registered as a CSI administrator, you can access the CSI Administration tab while logged in to ULN and perform the following tasks:

- Assign yourself as administrator of a CSI, or assign someone else as administrator of a CSI. See Section 3.14.1, "Becoming a CSI Administrator".
• List active CSIs, list the servers that are currently registered with an active CSI, and transfer those servers to another user or to another CSI. See Section 3.14.2, “Listing Active CSIs and Transferring Their Registered Servers”.

• List expired CSIs, list the servers that are currently registered with an expired CSI, and transfer those servers to another user or to another CSI. See Section 3.14.3, “Listing Expired CSIs and Transferring Their Registered Servers”.

• Remove yourself or someone else as administrator of a CSI. See Section 3.14.4, “Removing a CSI Administrator”.

3.14.1 Becoming a CSI Administrator

You can become an administrator of a CSI in one of the following ways:

• When you register with ULN, if no administrator is currently assigned to manage the CSI, you are prompted to click Confirm to become the CSI administrator. If you click Cancel, you cannot access the CSI administration feature.

• When logged into ULN, if you access the System tab and no administrator is currently assigned to manage one of the CSIs for which you are registered, you are prompted to choose whether to become the CSI administrator.

To become a CSI administrator:

1. Click the red link labeled enter the CSI you would like to be the administrator for in this page.
2. On the Add CSI page, verify the CSI and click Confirm.

Note
On the Systems page, the CSIs of all systems that have no assigned administrator are also shown in red.

• If you are already an administrator of a CSI, you can add yourself as administrator of another CSI provided that you have registered either a server or your ULN user name with the other CSI.

To assign yourself as administrator of an additional CSI:

1. Log in to ULN and select the CSI Administration tab.
2. On the Managed CSIs page, click Add CSI.
3. On the Assign Administrator page, enter the CSI, and click Add.
4. If there are existing administrators, the page lists these administrators and prompts you to click Confirm to confirm your request. Each administrator is sent an email to inform them that you have added yourself as an administrator of the CSI.

• An administrator for a CSI can add you as an administrator for the same CSI.

To assign another administrator to a CSI:

1. Log in to ULN as administrator of the CSI, and select the CSI Administration tab.
2. On the Managed CSIs page, click List Administrators.
3. On the CSI Administrators page, click Assign Administrator.
4. On the Assign Administrator page in the Select New Administrator list, click the + icon that is next to the user name of the user that you want to add as an administrator. Their user name is added to the Administrator box.

5. If you administer more than one CSI, select the CSI that the user will administer from the CSI drop down list.

6. Click Assign Administrator.

Note
If you want to become the administrator of a CSI but the person to whom it is registered is no longer with your organization, contact an Oracle support representative to request that you be made the administrator for the CSI.

3.14.2 Listing Active CSIs and Transferring Their Registered Servers

To list details of the active CSIs for which you are the administrator:

1. Log in to ULN as administrator of the CSI, and select the CSI Administration tab.

2. On the Managed CSIs page in the Select Managed CSI Services pane, select the Active link. The Managed Active CSI Services pane displays the service details for each active CSI that you administer.

3. Click the View # Server(s) link to display the details of the servers that are registered to an active CSI.

4. On the Registered Servers page, you can transfer one or more systems to another user or to another CSI that you administer.

Note
If you transfer a system to another user, at least one of the following conditions must be true:

- His or her user name must be registered to this CSI.
- One or more of the servers, for which they are the owner, must be registered to this CSI.
- He or she must be an administrator of at least one CSI for which you are also an administrator.

To transfer systems to another user:

a. Select the Transfer System check boxes for the systems that you want to transfer.

b. Click Transfer Selected Systems to Another Owner.

c. On the Transfer Registered System(s) - Owner page in the Transfer To column, click the red arrow icon that is next to the user name of the user to whom you want to transfer ownership.

d. On the Confirm Transfer Profile - Owner page, click Apply Changes to confirm the transfer to the new owner.

To transfer systems to another CSI:

a. Select the Transfer System check boxes for the systems that you want to transfer.
b. Click **Transfer Selected Systems to Another CSI**.

c. On the Transfer Registered System(s) - CSI page in the Transfer To column, click the red arrow icon that is next to the CSI to which you want to transfer the systems.

d. On the Confirm Transfer Profile - CSI page, click **Apply Changes** to confirm the transfer to the new CSI.

3.14.3 **Listing Expired CSIs and Transferring Their Registered Servers**

To list details of the expired CSIs for which you are the administrator:

1. Log in to ULN as administrator of the CSI, and select the CSI Administration tab.

2. On the Managed CSIs page in the Select Managed CSI Services pane, select the **Expired** link. The Managed Expired CSI Services pane displays the service details for each expired CSI that you administer.

3. Click the **View #  Server(s)** link to display the details of the servers that are registered to an expired CSI.

4. On the Registered Servers page, you can transfer one or more systems to another user or to another CSI that you administer.

   **Note**

   If you transfer a system to another user, at least one of the following conditions must be true:
   - His or her user name must be registered to this CSI.
   - One or more of the servers, for which they are the owner, must be registered to this CSI.
   - He or she must be an administrator of at least one CSI for which you are also an administrator.

   To transfer systems to another user:

   a. Select the **Transfer System** check boxes for the systems that you want to transfer.

   b. Click **Transfer Selected Systems to Another Owner**.

   c. On the Transfer Registered System(s) - Owner page in the Transfer To column, click the red arrow icon that is next to the user name of the user to whom you want to transfer ownership.

   d. On the Confirm Transfer Profile - Owner page, click **Apply Changes** to confirm the transfer to the new owner.

   To transfer systems to another CSI:

   a. Select the **Transfer System** check boxes for the systems that you want to transfer.

   b. Click **Transfer Selected Systems to Another CSI**.

   c. On the Transfer Registered System(s) - CSI page in the Transfer To column, click the red arrow icon that is next to the CSI to which you want to transfer the systems.
Removing a CSI Administrator

d. On the Confirm Transfer Profile - CSI page, click **Apply Changes** to confirm the transfer to the new CSI.

### 3.14.4 Removing a CSI Administrator

To remove an administrator who is registered for a CSI:

1. Log in to ULN and select the CSI Administration tab.
2. On the Managed CSIs page, click **List Administrators**.
3. On the CSI Administrators page in the Delete? column, click the trash can icon that is next to the user name of the user that you want to remove as administrator for the CSI specified in the same row.
4. When prompted to confirm that you want to revoke administration privileges for the CSI from that user, click **OK**.

### 3.15 Switching from RHN to ULN

**Note**

This procedure is for a Red Hat Enterprise Linux 6 system. For details of equivalent procedures for Red Hat Enterprise Linux 3, 4, and 5, see [http://linux.oracle.com/switch.html](http://linux.oracle.com/switch.html).

If you have an Oracle Linux 6 system that is registered with the Red Hat Network (RHN), you can use the `uln_register` utility to register it as described in Section 3.5, “Registering an Oracle Linux 6 or Oracle Linux 7 System”.

You must have a ULN account before you can register a system with ULN. You can create a ULN account at [http://linux.oracle.com/register](http://linux.oracle.com/register).

To register your system with ULN instead of RHN:


   If the `rhn-setup-gnome` package is already installed on your system, also download the `uln_register-gnome.tgz` from the same URL.

2. Extract the packages using the following command.

   ```
   # tar -xzf uln_register.tgz
   ``

   If the `rhn-setup-gnome` package is installed on your system, extract the packages from `uln_register-gnome.tgz`.

   ```
   # tar -xzf uln_register-gnome.tgz
   ```

3. Change to the `uln_migrate` directory and install the registration packages.

   ```
   # cd ./uln_migrate
   # rpm -Uvh *.rpm
   ```

4. Run the `uln_register` command.

   ```
   # uln_register
   ```
5. Follow the instructions on the screen to complete the registration. The `uln_register` utility collects information about your system and uploads it to Oracle.

3.16 For More Information About ULN

You can find out more information about ULN at [https://linux.oracle.com/](https://linux.oracle.com/).
Chapter 4 Ksplice Uptrack

4.1 About Ksplice Uptrack

Ksplice Uptrack can update a running Linux kernel without requiring an immediate reboot of the system. You can apply Ksplice updates to both the Unbreakable Enterprise Kernel and the Red Hat Compatible Kernel. Oracle creates each Ksplice patch from a kernel update that originates from either Oracle or the Linux kernel community. Ksplice Uptrack allows you to apply the latest kernel security errata for Common Vulnerabilities and Exposures (CVEs) without halting the system or restarting applications. Ksplice Uptrack applies the update patches in the background with a negligible impact, usually consisting of a pause of at most a few milliseconds. Ksplice Uptrack allows you to keep your systems secure and highly available. You can use Ksplice Uptrack and still upgrade your kernel using your usual mechanism, such as by using `yum`.

4.1.1 Supported Kernels

You can use Ksplice Uptrack to bring the following Oracle Linux kernels up to date with the latest important security and bug fix patches:

- All Oracle Unbreakable Enterprise Kernel versions for Oracle Linux 5 and Oracle Linux 6 starting with 2.6.32-100.28.9 (released March 16, 2011).
- All Oracle Linux 6 kernels starting with the official release.
- All Oracle Linux 5 Red Hat Compatible Kernels starting with Oracle Linux 5.4 (2.6.18-164.el5, released September 9, 2009).
- All Oracle Linux 5 Red Hat Compatible Kernels with bug fixes added by Oracle starting with Oracle Linux 5.6 (2.6.18-238.0.0.0.1.el5, released January 22, 2011).

To confirm whether a particular kernel is supported, install the Uptrack client on a system that is running the kernel.

If you have a question about supported kernels, send e-mail to ksplice-support_ww@oracle.com.
4.2 Registering to Use Ksplice Uptrack

When you register your systems with ULN, you can opt to use Oracle Ksplice if you have an Oracle Linux Premier Support account. If you choose to use Ksplice, you can subscribe your systems to the Ksplice for Oracle Linux channel and install the Ksplice Uptrack software on them. To install the `uptrack` package after registration is complete, you can use `yum` on an Oracle Linux 6 system or `up2date` on an Oracle Linux 5 system. The Uptrack client downloads the access key from ULN and automatically configures itself so that you can immediately begin to use Ksplice Uptrack.

If you already have an account on ULN, you can register your system to use Ksplice Uptrack at http://linux.oracle.com.

1. From your browser, log in to ULN with your existing user name and password. If your subscription grants you access to Ksplice, the ULN home page displays the Ksplice Uptrack Registration button.
2. Click Ksplice Uptrack Registration. The screen displays all valid Customer Support Identifiers (CSIs) for your account.
3. Select the CSI that you want to use and click Register. The screen displays an acknowledgment that a Ksplice account has been created and that an e-mail containing the Ksplice access key, a temporary password for Ksplice, and a URL for confirming your registration has been sent to your e-mail account.
4. When you receive the e-mail, open the URL that it contains.
5. Complete the form to confirm your registration, and click Continue.

After registering to use Ksplice Uptrack, you can log in at https://uptrack.ksplice.com using your e-mail address as your user name, and the temporary password. You must change your password when you first log in. You can view the status of your registered systems, the patches that have been applied, and the patches that are available. You can also create access control groups for your registered systems.

4.3 Installing Ksplice Uptrack

If you have an Oracle Linux Premier Support account and you have registered to use Oracle Ksplice, you can configure your registered systems to use Ksplice Uptrack through the Ksplice for Oracle Linux channel on ULN by using `yum`.

The system on which you want to install Ksplice Uptrack must meet the following criteria:

- The system must be registered with ULN.
- The operating system must be Oracle Linux 5 or Oracle Linux 6 with a supported version of either the Unbreakable Enterprise Kernel or the Red Hat Compatible Kernel installed. You can verify the kernel version by using the `uname -a` command. See Section 4.1.1, “Supported Kernels”.
- The kernel that is running currently is assumed to be the one that you want to update. Ksplice Uptrack applies updates only to the running kernel.
- The system must have access to the Internet.

To install Ksplice Uptrack from ULN:

1. Log in as `root` on the system.
2. If you use an Internet proxy, configure the HTTP and HTTPS settings for the proxy in the shell.
   - For the `sh`, `ksh`, or `bash` shells, use commands such as the following:

   ```
   # http_proxy=http://proxy_URL:http_port
   ```
Configuring Ksplice Uptrack

3. Using a browser, log in at http://linux.oracle.com with the ULN user name and password that you used to register the system, and perform the following steps:
   a. On the Systems tab, click the link named for your system in the list of registered machines.
   b. On the System Details page, click Manage Subscriptions.
   c. On the System Summary page, select the Ksplice for Oracle Linux channel for the correct release and your system's architecture (i386 or x86_64) from the list of available channels and click the right arrow (>) to move it to the list of subscribed channels.
   d. Click Save Subscriptions and log out of the ULN.

4. On your system, use yum to install the uptrack package.

   # yum install -y uptrack

   The access key for Ksplice Uptrack is retrieved from ULN and added to /etc/uptrack/uptrack.conf, for example:

   [Auth]
   accesskey = 0e1859ad8aea14b0b4306349142ce9160353297daee30240dab4d61f4ea4e59b

5. To enable the automatic installation of updates, change the following entry in /etc/uptrack/uptrack.conf:

   autoinstall = no

   so that it reads:

   autoinstall = yes

   For information about configuring Ksplice Uptrack, see Section 4.4, “Configuring Ksplice Uptrack”.

   For information about managing Ksplice updates, see Section 4.5, “Managing Ksplice Updates”.

4.4 Configuring Ksplice Uptrack

The configuration file for Ksplice Uptrack is /etc/uptrack/uptrack.conf. You can modify this file to configure a proxy server, to install updates automatically at boot time, or to check for and apply new updates automatically.

Ksplice Uptrack communicates with the Uptrack server by connecting to https://updates.ksplice.com:443. You can either configure your firewall to allow connection via port 443, or you can configure Ksplice Uptrack to use a proxy server. To configure Ksplice Uptrack to use a proxy server, set the following entry in /etc/uptrack/uptrack.conf:

https_proxy = https://proxy_URL:https_port

You receive e-mail notification when Ksplice updates are available for your system.

To make Ksplice Uptrack install all updates automatically as they become available, set the following entry:
Managing Ksplice Updates

autoinstall = yes

Note

Enabling automatic installation of updates does not automatically update Ksplice Uptrack itself. Oracle notifies you by e-mail when you can upgrade the Ksplice Uptrack software using `yum`.

To install updates automatically at boot time, the following entry must appear in `/etc/uptrack/uptrack.conf`:

`install_on_reboot = yes`

When you boot the system into the same kernel, the `/etc/init.d/uptrack` script reapplies the installed Ksplice updates to the kernel.

To prevent Ksplice Uptrack from automatically reapplying updates to the kernel when you reboot the system, set the entry to:

`install_on_reboot = no`

To install all available updates at boot time, even if you boot the system into a different kernel, uncomment the following entry in `/etc/uptrack/uptrack.conf`:

`#upgrade_on_reboot = yes`

so that it reads:

`upgrade_on_reboot = yes`

4.5 Managing Ksplice Updates

Ksplice patches are stored in `/var/cache/uptrack`. Following a reboot, Ksplice Uptrack automatically re-applies these patches very early in the boot process before the network is configured, so that the system is hardened before any remote connections can be established.

To list the available Ksplice updates, use the `uptrack-upgrade` command:

`# uptrack-upgrade -n`

To install all available Ksplice updates, enter:

`# uptrack-upgrade -y`

To install an individual Ksplice update, specify the update’s ID as the argument (in this example, the ID is `dfvn0zq8`):

`# uptrack-upgrade dfvn0zq8`

After Ksplice has applied updates to a running kernel, the kernel has an effective version that is different from the original boot version displayed by the `uname -a` command. Use the `uptrack-uname` command to display the effective version of the kernel:

`# uptrack-uname -a`

`uptrack-uname` supports the commonly used `uname` flags, including `-a` and `-r`, and provides a way for applications to detect that the kernel has been patched. The effective version is based on the version number of the latest patch that Ksplice Uptrack has applied to the kernel.

To view the updates that Ksplice has made to the running kernel:

`# uptrack-show`
To view the updates that are available to be installed:

```bash
# uptrack-show --available
```

To remove all updates from the kernel:

```bash
# uptrack-remove --all
```

To prevent Ksplice Uptrack from reapplying the updates at the next system reboot, create the empty file `/etc/uptrack/disable`:

```bash
# touch /etc/uptrack/disable
```

Alternatively, specify `nouptrack` as a parameter on the boot command line when you next restart the system.

### 4.6 Patching and Updating Your System

Ksplice patches allow you to keep a system up to date while it is running. You should also use `yum` or `rpm` to install the regular kernel RPM packages for released errata that are available from the Unbreakable Linux Network (ULN) or the Oracle Public Yum server. Your system will then be ready for the next maintenance window or reboot. When you do restart the system, you can boot it from a newer kernel version. Ksplice Uptrack uses the new kernel as a baseline for applying patches as they become available.

### 4.7 Removing the Ksplice Uptrack software

To remove the Ksplice Uptrack software from a system, enter:

```bash
# yum -y remove uptrack
```

### 4.8 About Ksplice Offline Client

Ksplice Offline Client removes the requirement for a server on your intranet to have a direct connection to the Oracle Uptrack server. All available Ksplice updates for each supported kernel version are bundled into an RPM that is specific to that version, and this package is updated every time that a new Ksplice patch becomes available for the kernel.

A Ksplice offline client does not require a network connection to be able to apply the update package to the kernel. For example, you could use `rpm` to install the update package from a memory stick. However, a more usual arrangement would be to create a local yum server that acts as a mirror of the Ksplice for Oracle Linux channels on ULN. At regular intervals, you download the latest Ksplice update packages to this server. After installing Ksplice Offline Client on your local systems, they can connect to the local ULN mirror to receive updates. They do not require access the Oracle Uptrack server.

**Note**

You cannot use the web interface or the Ksplice Uptrack API to monitor systems that are running Ksplice Offline Client as such systems are not registered with https://uptrack.ksplice.com.

### 4.8.1 Modifying a Local Yum Server to Act as a Ksplice Mirror

The system that you want to set up as a Ksplice mirror must meet the following criteria:

- You must have registered the system with ULN.
- You must have configured the system as a local yum server. See Section 2.8, “Creating and Using a Local ULN Mirror”.

---

43
• The system should also have enough disk space to store copies of the packages that it hosts. As a general rule, you require between 6 and 10 GB of space for the packages of each major release.

To set up a local yum server as a Ksplice mirror:

1. Using a browser, log in at http://linux.oracle.com with the ULN user name and password that you used to register the system.

2. On the Systems tab, click the link named for your system in the list of registered machines.

3. On the System Details page, click Edit.

4. On the Edit System Properties page, verify that the Yum Server check box is selected and click Apply Changes.

5. On the System Details page, click Manage Subscriptions.

6. On the System Summary page, select channels from the list of available or subscribed channels and click the arrows to move the channels between the lists.

Modify the subscribed channels to include Ksplice for Oracle Linux for the system architectures that you want to support as well as any other channels that you want to make available to local systems.

For example, the following table shows the channels that are available for Ksplice on Oracle Linux.

<table>
<thead>
<tr>
<th>Channel Name</th>
<th>Channel Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ksplice for Oracle Linux 5 (i386)</td>
<td>ol5_i386_ksplice</td>
<td>Oracle Ksplice clients, updates, and dependencies for Oracle Linux 5 on i386 systems.</td>
</tr>
<tr>
<td>Ksplice for Oracle Linux 5 (x86_64)</td>
<td>ol5_x86_64_ksplice</td>
<td>Oracle Ksplice clients, updates, and dependencies for Oracle Linux 5 on x86_64 systems.</td>
</tr>
<tr>
<td>Ksplice for Oracle Linux 6 (i386)</td>
<td>ol6_i386_ksplice</td>
<td>Oracle Ksplice clients, updates, and dependencies for Oracle Linux 6 on i386 systems.</td>
</tr>
<tr>
<td>Ksplice for Oracle Linux 6 (x86_64)</td>
<td>ol6_x86_64_ksplice</td>
<td>Oracle Ksplice clients, updates, and dependencies for Oracle Linux 6 on x86_64 systems.</td>
</tr>
<tr>
<td>Ksplice for Oracle Linux 7 (x86_64)</td>
<td>ol7_x86_64_ksplice</td>
<td>Oracle Ksplice clients, updates, and dependencies for Oracle Linux 7 on x86_64 systems.</td>
</tr>
</tbody>
</table>

For more information about the release channels that are available, see http://www.oracle.com/technetwork/articles/servers-storage-admin/yum-repo-setup-1659167.html.

7. When you have finished selecting channels, click Save Subscriptions and log out of ULN.

### 4.8.2 Configuring Ksplice Offline Clients

Once you have set up a local yum server that can act as a Ksplice mirror, you can configure your other systems to receive yum and Ksplice updates.

To configure a system as a Ksplice offline client:

1. In the /etc/yum.repos.d directory, edit the existing repository file, such as public-yum-ol6.repo or ULN-base.repo, and disable all entries by setting enabled=0.

2. In the /etc/yum.repos.d directory, create the file local-yum.repo, which contains entries such as the following for an Oracle Linux 6 client:

```bash
[o16_x86_64_ksplice]
name=Ksplice for $releasever - $basearch
```
Configuring Ksplice Offline Clients

```bash
Baseurl=http://local_yum_server/yum/OracleLinux/OL6/ksplice/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[ol6_latest]
name=Oracle Linux $releasever - $basearch - latest
baseurl=http://local_yum_server/yum/OracleLinux/OL6/latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=1

[ol6_addons]
name=Oracle Linux $releasever - $basearch - addons
baseurl=http://local_yum_server/yum/OracleLinux/OL6/addons/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[ol6_oracle]
name=Oracle Linux $releasever - $basearch - oracle
baseurl=http://local_yum_server/yum/OracleLinux/OL6/oracle/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[ol6_ga_base]
name=Oracle Linux $releasever GA - $basearch - base
baseurl=http://local_yum_server/yum/OracleLinux/OL6/0/base/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[ol6_u1_base]
name=Oracle Linux $releasever U1 - $basearch - base
baseurl=http://local_yum_server/yum/OracleLinux/OL6/1/base/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[ol6_u2_base]
name=Oracle Linux $releasever U2 - $basearch - base
baseurl=http://local_yum_server/yum/OracleLinux/OL6/2/base/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[ol6_u3_base]
name=Oracle Linux $releasever U3 - $basearch - base
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[ol6_ga_patch]
name=Oracle Linux $releasever GA - $basearch - patch
baseurl=http://local_yum_server/yum/OracleLinux/OL6/0/patch/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0

[ol6_u1_patch]
name=Oracle Linux $releasever U1 - $basearch - patch
baseurl=http://local_yum_server/yum/OracleLinux/OL6/1/patch/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY
gpgcheck=1
enabled=0
```
Replace `local_yum_server` with the IP address or resolvable host name of the local yum server.

In the sample configuration, only the `ol6_latest` and `ol6_x86_64_ksplice` channels are enabled.

### Note

As an alternative to specifying a `gpgkey` entry for each repository definition, you can use the following command to import the GPG key:

```bash
# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY
```

3. Install the Ksplice offline client.

```bash
# yum install uptrack-offline
```

If `yum` cannot connect to the local yum server, check that the firewall settings on that server allow incoming TCP connections to port 80.

4. Install the Ksplice updates that are available for the kernel.

```bash
# yum install uptrack-updates-`uname -r`
```

For an Oracle Linux 5 client, use this form of the command instead:

```bash
# yum install uptrack-updates-`uname -r`.`uname -m`
```

As new Ksplice updates are made available, you can use this command to pick up these updates and apply them. It is recommended that you set up a `cron` job to perform this task. For example, the following `crontab` entry for `root` runs the command once per day at 7am:

```
0 7 * * * yum install uptrack-updates-`uname -r`
```

To display information about Ksplice updates, use the `rpm -qa | grep uptrack-updates` and `uptrack-show` commands.

### 4.9 For More Information About Ksplice Uptrack

You can find out more information about Ksplice Uptrack at [http://www.ksplice.com/](http://www.ksplice.com/).
Chapter 5 The Btrfs File System

Table of Contents

5.1 About the Btrfs File System ................................................................. 47
5.2 Creating a Btrfs File System ............................................................... 48
5.3 Modifying a Btrfs File System ............................................................ 49
5.4 Compressing and Defragmenting a Btrfs File System .......................... 50
5.5 Resizing a Btrfs File System ............................................................... 51
5.6 Creating Subvolumes and Snapshots .................................................. 51
  5.6.1 Cloning Virtual Machine Images and Linux Containers .................. 53
5.7 Using the Send/Receive Feature ......................................................... 53
  5.7.1 Using Send/Receive to Implement Incremental Backups ................. 53
5.8 Using Quota Groups ................................................................. 54
5.9 Replacing Devices on a Live File System ........................................... 54
5.10 Creating Snapshots of Files .............................................................. 55
5.11 Installing a Btrfs root File System .................................................... 55
  5.11.1 Converting a Non-root File System ............................................. 55
  5.11.2 Converting the root File System .................................................. 56
  5.11.3 Mounting the Image of the Original File System ......................... 57
  5.11.4 Deleting the Snapshot of the Original File System ....................... 58
  5.11.5 Recovering an Original Non-root File System ............................. 58
5.12 Installing a Btrfs File System .......................................................... 58
  5.12.1 Setting up a New NFS Server .................................................... 59
  5.12.2 Configuring an Existing NFS Server ........................................... 60
  5.12.3 Setting up a New HTTP Server ................................................... 60
  5.12.4 Configuring an Existing HTTP Server ........................................ 61
  5.12.5 Setting up a Network Installation Server .................................... 62
  5.12.6 Installing from a Network Installation Server ............................. 63
  5.12.7 About the Installation root File System ..................................... 64
  5.12.8 Creating Snapshots of the root File System ............................... 65
  5.12.9 Mounting Alternate Snapshots as the root File System ............... 65
  5.12.10 Deleting Snapshots of the root File System .................................. 65
5.13 For More Information About Btrfs ..................................................... 66

This chapter describes how to deploy and use the advanced features of the btrfs file system.

5.1 About the Btrfs File System

The btrfs file system is designed to meet the expanding scalability requirements of large storage subsystems. As the btrfs file system uses B-trees in its implementation, its name derives from the name of those data structures, although it is not a true acronym. A B-tree is a tree-like data structure that enables file systems and databases to efficiently access and update large blocks of data no matter how large the tree grows.

The btrfs file system provides the following important features:

- Copy-on-write functionality allows you to create both readable and writable snapshots, and to roll back a file system to a previous state, even after you have converted it from an ext3 or ext4 file system.
- Checksum functionality ensures data integrity.
• Transparent compression saves disk space.

• Transparent defragmentation improves performance.

• Integrated logical volume management allows you to implement RAID 0, RAID 1, or RAID 10 configurations, and to dynamically add and remove storage capacity.

Starting with Oracle Linux 6 Update 3, the UEK Boot ISO (which boots the Unbreakable Enterprise Kernel as the installation kernel) allows you to configure a btrfs root file system. Prior to Oracle Linux 6 Update 3, you could not create a btrfs root file system during installation. For more information, see Section 5.12, “Installing a Btrfs root File System”.

With UEK R3, btrfs supports the following additional features:

• The send/receive feature allows you to record the differences between two subvolumes, which can either be snapshots of the same subvolume or parent and child subvolumes.

• Quota groups (qgroups) allow you to set different size limits for a volume and its subvolumes.

• You can replace devices without unmounting or otherwise disrupting access to the file system.

### 5.2 Creating a Btrfs File System

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mkfs.btrfs block_device</td>
<td>Create a btrfs file system on a single device. For example: mkfs.btrfs /dev/sdb1</td>
</tr>
<tr>
<td>mkfs.btrfs -L label block_device</td>
<td>Create a btrfs file system with a label that you can use when mounting the file system. For example: mkfs.btrfs -L myvolume /dev/sdb2</td>
</tr>
<tr>
<td>mkfs.btrfs -m single block_device</td>
<td>Create a btrfs file system on a single device, but do not duplicate the metadata on that device. For example: mkfs.btrfs -m single /dev/sdc</td>
</tr>
</tbody>
</table>

Note

If the btrfs-progs package is not already installed on your system, use yum to install it.

You can use the mkfs.btrfs command to create a btrfs file system that is laid out across one or more block devices. The default configuration is to stripe the file system data and to mirror the file system metadata across the devices. If you specify a single device, the metadata is duplicated on that device unless you specify that only one copy of the metadata is to be used. The devices can be simple disk partitions, loopback devices (that is, disk images in memory), multipath devices, or LUNs that implement RAID in hardware.

The following table illustrates how to use the mkfs.btrfs command to create various btrfs configurations.
Modifying a Btrfs File System

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mkfs.btrfs block_device1 block_device2 ...</code></td>
<td>Stripe the file system data and mirror the file system metadata across several devices. For example: <code>mkfs.btrfs /dev/sdd /dev/sde</code></td>
</tr>
<tr>
<td><code>mkfs.btrfs -m raid0 block_device1 block_device2 ...</code></td>
<td>Stripe both the file system data and metadata across several devices. For example: <code>mkfs.btrfs -m raid0 /dev/sdd /dev/sde</code></td>
</tr>
<tr>
<td><code>mkfs.btrfs -d raid1 block_device1 block_device2 ...</code></td>
<td>Mirror both the file system data and metadata across several devices. For example: <code>mkfs.btrfs -d raid1 /dev/sdd /dev/sde</code></td>
</tr>
<tr>
<td><code>mkfs.btrfs -d raid10 -m raid10 block_device1 block_device2 block_device3 block_device4</code></td>
<td>Stripe the file system data and metadata across several mirrored devices. You must specify an even number of devices, of which there must be at least four. For example: <code>mkfs.btrfs -d raid10 -m raid10 /dev/sdf /dev/sdg /dev/sdh /dev/sdi /dev/sdj /dev/sdk</code></td>
</tr>
</tbody>
</table>

When you want to mount the file system, you can specify it by any of its component devices, for example:

```
# mkfs.btrfs -d raid10 -m raid10 /dev/sd[fghijk]
# mount /dev/sdf /raid10_mountpoint
```

To find out the RAID configuration of a mounted btrfs file system, use this command:

```
# btrfs filesystem df mountpoint
```

**Note**

The `btrfs filesystem df` command displays more accurate information about the space used by a btrfs file system than the `df` command does.

Use the following form of the `btrfs` command to display information about all the btrfs file systems on a system:

```
# btrfs filesystem show
```

5.3 Modifying a Btrfs File System

The following table shows how you can use the `btrfs` command to add or remove devices, and to rebalance the layout of the file system data and metadata across the devices.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>btrfs device add device mountpoint</code></td>
<td>Add a device to the file system that is mounted on the specified mount point. For example: <code>btrfs device add /dev/sdd /myfs</code></td>
</tr>
<tr>
<td><code>btrfs device delete device mountpoint</code></td>
<td>Remove a device from a mounted file system. For example:</td>
</tr>
</tbody>
</table>
5.4 Compressing and Defragmenting a Btrfs File System

You can compress a btrfs file system to increase its effective capacity, and you can defragment it to increase I/O performance.

To enable compression of a btrfs file system, specify one of the following mount options:

<table>
<thead>
<tr>
<th>Mount Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compress=lzo</td>
<td>Use LZO compression.</td>
</tr>
<tr>
<td>compress=zlib</td>
<td>Use zlib compression.</td>
</tr>
</tbody>
</table>

LZO offers a better compression ratio, while zlib offers faster compression.

You can also compress a btrfs file system at the same time that you defragment it.

To defragment a btrfs file system, use the following command:

```
# btrfs filesystem defragment filesystem_name
```

To defragment a btrfs file system and compress it at the same time:

```
# btrfs filesystem defragment -c filesystem_name
```

You can also defragment, and optionally compress, individual file system objects, such as directories and files, within a btrfs file system.

```
# btrfs filesystem defragment [-c] file_name ...
```

Note

You can set up automatic defragmentation by specifying the `autodefrag` option when you mount the file system. However, automatic defragmentation is not recommended for large databases or for images of virtual machines.

Defragmenting a file or a subvolume that has a copy-on-write copy results breaks the link between the file and its copy. For example, if you defragment a subvolume...
that has a snapshot, the disk usage by the subvolume and its snapshot will increase because the snapshot is no longer a copy-on-write image of the subvolume.

5.5 Resizing a Btrfs File System

You can use the `btrfs` command to increase the size of a mounted btrfs file system if there is space on the underlying devices to accommodate the change, or to decrease its size if the file system has sufficient available free space. The command does not have any effect on the layout or size of the underlying devices.

For example, to increase the size of `/mybtrfs1` by 2 GB:

```
# btrfs filesystem resize +2g /mybtrfs1
```

Decrease the size of `/mybtrfs2` by 4 GB:

```
# btrfs filesystem resize -4g /mybtrfs2
```

Set the size of `/mybtrfs3` to 20 GB:

```
# btrfs filesystem resize 20g /mybtrfs3
```

5.6 Creating Subvolumes and Snapshots

The top level of a btrfs file system is a subvolume consisting of a named b-tree structure that contains directories, files, and possibly further btrfs subvolumes that are themselves named b-trees that contain directories and files, and so on. To create a subvolume, change directory to the position in the btrfs file system where you want to create the subvolume and enter the following command:

```
# btrfs subvolume create subvolume_name
```

Snapshots are a type of subvolume that records the contents of their parent subvolumes at the time that you took the snapshot. If you take a snapshot of a btrfs file system and do not write to it, the snapshot records the state of the original file system and forms a stable image from which you can make a backup. If you make a snapshot writable, you can treat it as an alternate version of the original file system. The copy-on-write functionality of btrfs file system means that snapshots are quick to create, and consume very little disk space initially.

**Note**

Taking snapshots of a subvolume is not a recursive process. If you create a snapshot of a subvolume, every subvolume or snapshot that the subvolume contains is mapped to an empty directory of the same name inside the snapshot.

The following table shows how to perform some common snapshot operations:
Creating Subvolumes and Snapshots

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>btrfs subvolume snapshot pathname</code></td>
<td>Create a snapshot <code>snapshot_path</code> of a parent subvolume or snapshot specified by <code>pathname</code>. For example: <code>btrfs subvolume snapshot /mybtrfs /mybtrfs/snapshot1</code></td>
</tr>
<tr>
<td><code>btrfs subvolume list pathname</code></td>
<td>List the subvolumes or snapshots of a subvolume or snapshot specified by <code>pathname</code>. For example: <code>btrfs subvolume list /mybtrfs</code></td>
</tr>
</tbody>
</table>

**Note**
You can use this command to determine the ID of a subvolume or snapshot.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>btrfs subvolume set-default ID pathname</code></td>
<td>By default, mount the snapshot or subvolume specified by its ID instead of the parent subvolume. For example: <code>btrfs subvolume set-default 4 /mybtrfs</code></td>
</tr>
<tr>
<td><code>btrfs subvolume get-default pathname</code></td>
<td>Displays the ID of the default subvolume that is mounted for the specified subvolume. For example: <code>btrfs subvolume get-default /mybtrfs</code></td>
</tr>
</tbody>
</table>

You can mount a btrfs subvolume as though it were a disk device. If you mount a snapshot instead of its parent subvolume, you effectively roll back the state of the file system to the time that the snapshot was taken. By default, the operating system mounts the parent btrfs volume, which has an ID of 0, unless you use `set-default` to change the default subvolume. If you set a new default subvolume, the system will mount that subvolume instead in future. You can override the default setting by specifying either of the following `mount` options:

<table>
<thead>
<tr>
<th>Mount Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>subvolid= snapshot_ID</code></td>
<td>Mount the subvolume or snapshot specified by its subvolume ID instead of the default subvolume.</td>
</tr>
<tr>
<td><code>subvol=pathname/snapshot_path</code></td>
<td>Mount the subvolume or snapshot specified by its pathname instead of the default subvolume.</td>
</tr>
</tbody>
</table>

**Note**
The subvolume or snapshot must be located in the root of the btrfs file system.

When you have rolled back a file system by mounting a snapshot, you can take snapshots of the snapshot itself to record its state.

When you no longer require a subvolume or snapshot, use the following command to delete it:

```bash
# btrfs subvolume delete subvolume_path
```
5.6.1 Cloning Virtual Machine Images and Linux Containers

You can use a btrfs file system to provide storage space for virtual machine images and Linux Containers. The ability to quickly clone files and create snapshots of directory structures makes btrfs an ideal candidate for this purpose. For an example of using the snapshot feature of btrfs to implement Linux Containers, see Section 9.2, “Configuring Operating System Containers”.

5.7 Using the Send/Receive Feature

The send/receive feature requires that you boot the system using UEK R3.

The send operation compares two subvolumes and writes a description of how to convert one subvolume (the parent subvolume) into the other (the sent subvolume). You would usually direct the output to a file for later use or pipe it to a receive operation for immediate use.

The simplest form of the send operation writes a complete description of a subvolume:

```bash
# btrfs send [-v] [-f sent_file] ... subvol
```

You can specify multiple instances of the `-v` option to display increasing amounts of debugging output. The `-f` option allows you to save the output to a file. Both of these options are implicit in the following usage examples.

The following form of the send operation writes a complete description of how to convert one subvolume into another:

```bash
# btrfs send -p parent_subvol sent_subvol
```

If a subvolume such as a snapshot of the parent volume, known as a clone source, will be available during the receive operation from which some of the data can be recovered, you can specify the clone source to reduce the size of the output file:

```bash
# btrfs send [-p parent_subvol] -c clone_src [-c clone_src] ... subvol
```

You can specify the `-c` option multiple times if there is more than one clone source. If you do not specify the parent subvolume, btrfs chooses a suitable parent from the clone sources.

You use the receive operation to regenerate the sent subvolume at a specified path:

```bash
# btrfs receive [-f sent_file] mountpoint
```

5.7.1 Using Send/Receive to Implement Incremental Backups

The following procedure is a suggestion for setting up an incremental backup and restore process for a subvolume.

1. Create a read-only snapshot of the subvolume to serve as an initial reference point for the backup:

```bash
# btrfs subvolume snapshot -r /vol /vol/backup_0
```
2. Run `sync` to ensure that the snapshot has been written to disk:
   
   ```
   # sync
   ```

3. Create a subvolume or directory on a btrfs file system as a backup area to receive the snapshot, for example, `/backupvol`.

4. Send the snapshot to `/backupvol`:
   
   ```
   # btrfs send /vol/backup_0 | btrfs receive /backupvol
   ```

   This command creates the subvolume `/backupvol/backup_0`.

   Having created the reference backup, you can then create incremental backups as required.

5. To create an incremental backup:
   a. Create a new snapshot of the subvolume:
      
      ```
      # btrfs subvolume snapshot -r /vol /vol/backup_1
      ```
   
   b. Run `sync` to ensure that the snapshot has been written to disk:
      
      ```
      # sync
      ```
   
   c. Send only the differences between the reference backup and the new backup to the backup area:
      
      ```
      # btrfs send -p /vol/backup_0 /vol/backup_1 | btrfs receive /backupvol
      ```

      This command creates the subvolume `/backupvol/backup_1`.

5.8 Using Quota Groups

---

**Note**

The quota groups feature requires that you boot the system using UEK R3.

To enable quotas, use the following command on a newly created btrfs file system before any creating any subvolumes:

```
# btrfs quota enable volume
```

To assign a quota-group limit to a subvolume, use the following command:

```
# btrfs qgroup limit size /volume/subvolume
``` 

For example:

```
# btrfs qgroup limit 1g /myvol/subvol1
# btrfs qgroup limit 512m /myvol/subvol2
``` 

To find out the quota usage for a subvolume, use the `btrfs qgroup show path` command:

5.9 Replacing Devices on a Live File System

---

**Note**

The device replacement feature requires that you boot the system using UEK R3.
Creating Snapshots of Files

You can replace devices on a live file system. You do not need to unmount the file system or stop any tasks that are using it. If the system crashes or loses power while the replacement is taking place, the operation resumes when the system next mounts the file system.

Use the following command to replace a device on a mounted btrfs file system:

```
# btrfs replace start source_dev target_dev [-r] mountpoint
```

*source_dev* and *target_dev* specify the device to be replaced (source device) and the replacement device (target device). *mountpoint* specifies the file system that is using the source device. The target device must be the same size as or larger than the source device. If the source device is no longer available or you specify the `-r` option, the data is reconstructed by using redundant data obtained from other devices (such as another available mirror). The source device is removed from the file system when the operation is complete.

You can use the `btrfs replace status mountpoint` and `btrfs replace cancel mountpoint` commands to check the progress of the replacement operation or to cancel the operation.

### 5.10 Creating Snapshots of Files

You can use the `--relink` option to the `cp` command to create lightweight copies of a file within the same subvolume of a btrfs file system. The copy-on-write mechanism saves disk space and allows copy operations to be almost instantaneous. The btrfs file system creates a new inode that shares the same disk blocks as the existing file, rather than creating a complete copy of the file's data or creating a link that points to the file's inode. The resulting file appears to be a copy of the original file, but the original data blocks are not duplicated. If you subsequently write to one of the files, the btrfs file system makes copies of the blocks before they are written to, preserving the other file's content.

For example, the following command creates the snapshot `bar` of the file `foo`:

```
# cp -relink foo bar
```

### 5.11 Converting an Ext2, Ext3, or Ext4 File System to a Btrfs File System

You can use the `btrfs-convert` utility to convert an `ext2`, `ext3`, or `ext4` file system to `btrfs`. The utility preserves an image of the original file system in a snapshot named `ext2_saved`. This snapshot allows you to roll back the conversion, even if you have made changes to the btrfs file system.

If you convert the root file system to btrfs, you can use snapshots to roll back changes such as upgrades that you have made to the file system.

**Note**

You cannot convert a bootable partition, such as `/boot`, to a btrfs file system.

### 5.11.1 Converting a Non-root File System

**Caution**

Before performing a file system conversion, make a backup of the file system from which you can restore its state.

To convert an `ext2`, `ext3`, or `ext4` file system other than the root file system to `btrfs`:

1. Unmount the file system.
Converting the root File System

2. Run the correct version of `fsck` (for example, `fsck.ext4`) on the underlying device to check and correct the integrity of file system.

```bash
# fsck.ext4 -i device
```

3. Convert the file system to a btrfs file system.

```bash
# btrfs-convert device
```

4. Edit the file `/etc/fstab`, and change the file system type of the file system to btrfs, for example:

```bash
/dev/sdb               /myfs          btrfs    defaults  0 0
```

5. Mount the converted file system on the old mount point.

```bash
# mount device mountpoint
```

5.11.2 Converting the root File System

⚠️ **Caution**

Before performing a root file system conversion, make a full system backup from which you can restore its state.

To convert an ext2, ext3, or ext4 root file system to btrfs:

1. Run the `mount` command to determine the device that is currently mounted as the root file system, and the type of the file system.

   In the following example, the root file system is configured as an LVM logical volume `lv_root` in the volume group `vg_hostol6`, and the file system type is `ext4`. Using the `ls -l` command confirms that the mapped device corresponds to `/dev/vg_hostol6/lv_root`.

   ```bash
   # mount
   /dev/mapper/vg_hostol6-lv_root on / type ext4 (rw)
   ...
   # ls -l /dev/mapper/vg_hostol6-lv_root
   lrwxrwxrwx. 1 root root 7 Sep 14 14:00 /dev/mapper/vg_hostol6-lv_root -> ../dm-0
   # ls -l /dev/vg_hostol6/lv_root
   lrwxrwxrwx. 1 root root 7 Sep 14 14:00 /dev/vg_hostol6/lv_root -> ../dm-0
   ```

   In the next example, the root file system corresponds to the disk partition `/dev/sda2`:

   ```bash
   # mount
   /dev/sda2 on / type ext4 (rw)
   ...
   ```

2. Shut down the system.

3. Boot the system from an Oracle Linux 6 Update 3 or later UEK Boot ISO (which you can burn to CD or DVD if necessary). You can download the UEK Boot ISO from [https://edelivery.oracle.com/linux](https://edelivery.oracle.com/linux).

⚠️ **Note**

You must use the UEK Boot ISO. You cannot use the RHCK Boot ISO to perform the conversion.
4. From the installation menu, select **Rescue Installed System**. When prompted, choose a language and keyboard, select **Local CD/DVD** as the installation media, select **No** to bypass starting the network interface, and select **Skip** to bypass selecting a rescue environment.

5. Select **Start shell** to obtain a `bash` shell prompt (`bash-4.1#`) at the bottom of the screen.

6. If the existing root file system is configured as an LVM volume, use the following command to start the volume group (for example, `vg_hostol6`):

```
bash-4.1# lvchange -ay vg_hostol6
```

7. Run the correct version of **fsck** (for example, `fsck.ext3` or `fsck.ext4`) to check and correct the integrity of the file system.

```
bash-4.1# fsck.extN -f device
```

   where `device` is the root file system device (for example, `/dev/vg_hostol6/lv_root` or `/dev/sda2`).

8. Convert the file system to a btrfs file system.

```
bash-4.1# btrfs-convert device
```

9. Create a mount point (`/mnt1`) and mount the converted root file system on it.

```
bash-4.1# mkdir /mnt1
bash-4.1# mount -t btrfs device /mnt1
```

10. Use the **vi** command to edit the file `/mnt1/etc/fstab`, and change the file system type of the root file system to **btrfs**, for example:

```
/dev/mapper/vg_hostol6-lv_root / btrfs defaults 1 1
```

11. Create the file `.autorelabel` in the root of the mounted file system.

```
bash-4.1# touch /mnt1/.autorelabel
```

   The presence of the `.autorelabel` file in `/` instructs SELinux to recreate the security attributes of all files on the file system.

   **Note**
   If you do not create the `.autorelabel` file, you might not be able to boot the system successfully. If you forget to create the file and the reboot fails, either disable SELinux temporarily by specifying `selinux=0` to the kernel boot parameters, or run SELinux in permissive mode by specifying `enforcing=0`.

12. Unmount the converted root file system.

```
bash-4.1# umount /mnt1
```

13. Remove the boot CD, DVD, or ISO, and reboot the system.

### 5.11.3 Mounting the Image of the Original File System

To mount the image of the original file system read-only:

1. Mount the snapshot of the original file system on a temporary mount point.

```
# mount -t btrfs -o subvol=ext2_saved device temp_mountpoint1
```
2. Mount the image of the original file system read-only on another temporary mount point, specifying the correct file system type (ext2, ext3, or ext4) to the -t option.

```bash
# mount -t extN -o loop,ro temp_mountpoint1/image temp_mountpoint2
```

### 5.11.4 Deleting the Snapshot of the Original File System

**Caution**

If you delete the snapshot of the original file system to save storage space, you will no longer be able to recover the original file system.

To delete the snapshot of the original file system and recover the space that it uses:

1. Delete the ext2_saved subvolume.

```bash
# btrfs subvolume delete mountpoint/ext2_saved
```

For example, if you converted the root file system (/) file system, you would enter:

```bash
# btrfs subvolume delete //ext2_saved
```

For another file system, such as /usr, you would enter:

```bash
# btrfs subvolume delete /usr/ext2_saved
```

2. Rebalance the btrfs file system.

```bash
# btrfs filesystem balance device
```

### 5.11.5 Recovering an Original Non-root File System

**Caution**

If you roll back a conversion, you will lose any changes that you have made to the btrfs file system. Make a back up of the changes that you want to reapply to the restored file system.

To roll back the conversion of the file system and recover the original file system:

1. Unmount the btrfs file system and all of its snapshots and images in the reverse order from which you originally mounted them.

```bash
# umount temp_mountpoint2
# umount temp_mountpoint1/image
# umount mountpoint
```

2. Roll back the conversion.

```bash
# btrfs-convert -r device
```

3. Mount the original file system.

```bash
# mount -t extN device mountpoint
```

### 5.12 Installing a Btrfs root File System

For compatibility reasons, the default installation image of Oracle Linux boots the Red Hat compatible kernel to perform the installation. Oracle provides an alternative installation image (UEK Boot ISO) that
supports the installation of Oracle Linux 6 Update 3 or later using the Unbreakable Enterprise Kernel (UEK) as the installation kernel. This installation method allows you to create a btrfs root file system.

As the UEK Boot ISO contains only the bootable installation image, you must set up a network installation server for the RPM packages. This server must have sufficient storage space to host the full Oracle Linux Release 6 Update 3 or later Media Pack DVD image (approximately 3.5 GB), and you must configure it to serve the image files using either NFS or HTTP to the target system on which you want to install Oracle Linux 6 Update 3 or later.

• Section 5.12.1, “Setting up a New NFS Server”
• Section 5.12.2, “Configuring an Existing NFS Server”
• Section 5.12.3, “Setting up a New HTTP Server”
• Section 5.12.4, “Configuring an Existing HTTP Server”
• Section 5.12.5, “Setting up a Network Installation Server”
• Section 5.12.6, “Installing from a Network Installation Server”

5.12.1 Setting up a New NFS Server

Note
This procedure assumes that you are setting up an Oracle Linux 6 system as an NFSv4 server. Using NFSv4 greatly simplifies firewall configuration as you need only configure a single rule for TCP port 2049.

To set up an NFS server:
1. Install the nfs-utils package.

   # yum install nfs-utils

2. Create the directory where you will copy the full Oracle Linux Release 6 Media Pack DVD image, for example /var/OSimage/OL6.3:

   # mkdir -p /var/OSimage/OL6.3

3. Edit the configuration file, /etc/exports, as follows.
   a. Add an entry for the directory where you will copy the DVD image.

      The following example allows read-only access to the directory /var/OSimage/OL6.3 for any NFS client on the 192.168.1 subnet:

      /var/OSimage/OL6.3 192.168.1.0/24(ro)

   b. Save your changes to the file.

4. Start the NFS server, and configure it to start after a reboot.

   # service rpcbind start
   # service nfs start
   # service nfslock start
   # chkconfig rpcbind on
   # chkconfig nfs on
   # chkconfig nfslock on
5. If you have configured a firewall on your system, configure it to allow incoming NFSv4 requests from NFS clients.

For example, use the following commands to configure `iptables` to allow NFSv4 connections and save the change to the firewall configuration:

```
# iptables -I INPUT -p tcp --state NEW -m tcp --dport 2049 -j ACCEPT
# service iptables save
```

### 5.12.2 Configuring an Existing NFS Server

To configure an existing NFS server:

1. Create the directory where you will copy the full Oracle Linux Release 6 Media Pack DVD image, for example `/var/OSimage/OL6.3`:

   ```
   # mkdir -p /var/OSimage/OL6.3
   ```

2. Use the `exportfs` command to export the directory.

   ```
   # exportfs -i -o options client:export_dir
   ```

   For example, to allow read-only access to the directory `/var/OSimage/OL6.3` for any NFS client on the 192.168.1 subnet:

   ```
   # exportfs -i -o ro 192.168.1.0/24:/var/OSimage/OL6.3
   ```

### 5.12.3 Setting up a New HTTP Server

**Note**

These instructions assume that you are setting up an Oracle Linux 6 system as an Apache HTTP server.

To set up an HTTP server:

1. Install the Apache HTTP server package.

   ```
   # yum install httpd
   ```

2. Create the directory where you will copy the full Oracle Linux Release 6 Media Pack DVD image, for example `/var/www/html/OSimage/OL6.3`:

   ```
   # mkdir -p /var/www/html/OSimage/OL6.3
   ```

   **Note**

   If SELinux is enabled in enforcing mode on your system, create the directory under the `/var/www/html` directory hierarchy so that the `httpd_sys_content_t` file type is set automatically on all the files in the repository.

3. Edit the HTTP server configuration file, `/etc/httpd/conf/httpd.conf`, as follows:

   a. Specify the resolvable domain name of the server in the argument to `ServerName`.

   ```
   ServerName server_addr:80
   ```

   If the server does not have a resolvable domain name, enter its IP address instead. For example, the following entry would be appropriate for an HTTP server with the IP address 192.168.1.100.
Configuring an Existing HTTP Server

ServerName 192.168.1.100:80

b. If the directory to which you will copy the DVD image is not under /var/www/html, change the default setting of DocumentRoot.

In this example, the DVD image will be copied to /var/www/html/OSimage/OL6.3 so the setting of DocumentRoot can remain unchanged.

DocumentRoot "/var/www/html"

c. Verify that the <Directory> setting points to the same setting as DocumentRoot.

# This should be changed to whatever you set DocumentRoot to.
# <Directory "/var/www/html">

Note

The Indexes option is not required for installation.

d. If you want to be able to browse the directory hierarchy, verify that the Options directive specifies the Indexes option, for example:

Options Indexes FollowSymLinks

e. Save your changes to the file.

4. Start the Apache HTTP server, and configure it to start after a reboot.

# service httpd start
# chkconfig httpd on

5. If you have enabled a firewall on your system, configure it to allow incoming HTTP connection requests on TCP port 80.

For example, the following command configures iptables to allow incoming HTTP connection requests and saves the change to the firewall configuration:

# iptables -I INPUT -p tcp --state NEW -m tcp --dport 80 -j ACCEPT
# service iptables save

5.12.4 Configuring an Existing HTTP Server

To configure an existing Apache HTTP server:

1. Under the DocumentRoot hierarchy that is defined in the HTTP server configuration file (/etc/httpd/conf/httpd.conf), create the directory where you will copy the full Oracle Linux Release 6 Media Pack DVD image, for example /var/www/html/OSimage/OL6.3:

# mkdir -p /var/www/html/OSimage/OL6.3

2. Edit the HTTP server configuration file, /etc/httpd/conf/httpd.conf, and add a <Directory> section, for example:

<Directory "/var/www/html/OSimage/OL6.3">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
</Directory>
5.12.5 Setting up a Network Installation Server

Note
This procedure assumes that you have set up the system as an NFS or HTTP server.

To set up a network installation server:

1. Download the full Oracle Linux Media Pack DVD image (for example, V41362-01.iso for x86_64 (64 bit) Oracle Linux Release 6 Update 5) from the Oracle Software Delivery Cloud at http://edelivery.oracle.com/linux.

2. Mount the DVD image on a suitable mount point (for example, /mnt):

   ```bash
   # mount -t iso9660 -o loop V41362-01.iso mount_dir
   ```

3. Use the following command to extract the contents of the DVD image into a directory (output_dir) whose contents are shareable using NFS or HTTP:

   ```bash
   # cp -a -T mount_dir output_dir
   ```

   For example, to copy the DVD image mounted on /mnt to /var/OSimage/OL6.5:

   ```bash
   # cp -a -T /mnt /var/OSimage/OL6.5
   ```

   or to /var/www/html/OSimage/OL6.5:

   ```bash
   # cp -a -T /mnt /var/www/html/OSimage/OL6.5
   ```

4. Unmount the DVD image:

   ```bash
   # umount mount_dir
   ```

5. Download the UEK Boot ISO image for the desired architecture (for example, V41364-01.iso for x86_64 (64 bit)).

6. Mount the UEK Boot ISO image:

   ```bash
   # mount -t iso9660 -o loop V41364-01.iso
   ```

7. Replace the contents of the images directory that you copied from the DVD image with the contents of the images directory from the UEK Boot ISO image:

   ```bash
   # rm -rf output_dir/images
   # cp -r mount_dir/images output_dir
   ```

   For example, to replace /var/OSimage/OL6.5/images:

   ```bash
   # rm -rf /var/OSimage/OL6.5/images
   ```
Installing from a Network Installation Server

To install a target system from a network installation server:

1. Boot the target system using the UEK Boot ISO.

2. Select Install or upgrade an existing system, press Tab, and enter askmethod as an additional parameter on the boot command line:

```
> vmlinuz initrd=initrd.img askmethod
```

3. On the Installation Method screen, select either NFS directory or URL depending on whether you configured your installation server to use NFS or HTTP respectively.

4. After configuring the network settings, enter the settings for the NFS or HTTP installation server.

   For installation using NFS, enter the path of the full DVD image, for example /var/OSimage/OL6.5.

   For installation using HTTP, enter the URL of the full DVD image, for example http://192.168.1.100/OSimage/OL6.5.

5. The default disk layout creates a btrfs root file system.

   **Note**
   You cannot configure a bootable partition, such as /boot, as a btrfs file system.
5.12.7 About the Installation root File System

The mounted root file system is a snapshot (named \texttt{install}) of the root file system taken at the end of installation. To find out the ID of the parent of the root file system subvolume, use the following command:

```
# btrfs subvolume list /
ID 258 top level 5 path install
```

In this example, the installation root file system subvolume has an ID of 5. The subvolume with ID 258 (\texttt{install}) is currently mounted as \texttt{/}. Figure 5.1, "Layout of the root File System Following Installation" illustrates the layout of the file system:

![Layout of the root File System Following Installation](image)

The top-level subvolume with ID 5 records the contents of the root file system file system at the end of installation. The default subvolume (\texttt{install}) with ID 258 is currently mounted as the active root file system.

The \texttt{mount} command shows the device that is currently mounted as the root file system:

```
# mount
/dev/mapper/vg_btrfs-lv_root on / type btrfs (rw)
```

To mount the installation root file system volume, you can use the following commands:

```
# mkdir /instroot
# mount -o subvolid=5 /dev/mapper/vg_btrfs-lv_root /instroot
```

If you list the contents of \texttt{/instroot}, you can see both the contents of the installation root file system volume and the \texttt{install} snapshot, for example:

```
# ls /instroot
bin  cgroup  etc  install  lib64  misc  net  proc  sbin  srv  tmp  var
boot  dev  home  lib  media  mnt  opt  root  selinux  sys  usr
```

The contents of \texttt{/} and \texttt{/instroot/install} are identical as demonstrated in the following example where a file (\texttt{foo}) created in \texttt{/instroot/install} is also visible in \texttt{/}:

```
# touch /instroot/install/foo
# ls /
```

---

64
Creating Snapshots of the root File System

5.12.8 Creating Snapshots of the root File System

To take a snapshot of the current root file system:

1. Mount the top level of the root file system on a suitable mount point.

   ```
   # mount -o subvolid=5 /dev/mapper/vg_btrfs-lv_root /mnt
   ```

2. Change directory to the mount point and take the snapshot. In this example, the `install` subvolume is currently mounted as the root file system system.

   ```
   # cd /mnt
   # btrfs subvolume snapshot install root_snapshot_1
   Create a snapshot of 'install' in './root_snapshot_1'
   ```

3. Change directory to `/` and unmount the top level of the file system.

   ```
   # cd /
   # umount /mnt
   ```

   The list of subvolumes now includes the newly created snapshot.

   ```
   # btrfs subvolume list /
   ID 258 top level 5 path install
   ID 260 top level 5 path root_snapshot_1
   ```

5.12.9 Mounting Alternate Snapshots as the root File System

If you want to roll back changes to your system, you can mount a snapshot as the root file system by specifying its ID as the default subvolume, for example:

```
# btrfs subvolume set-default 260 /
```

Reboot the system for the change to take effect.

5.12.10 Deleting Snapshots of the root File System

To delete a snapshot:

1. Mount the top level of the file system, for example:

   ```
   # mount -o subvolid=5 /dev/mapper/vg_btrfs-lv_root /mnt
   ```

2. Change directory to the mount point and delete the snapshot.

   ```
   # cd /mnt
   # btrfs subvolume delete install
   Delete subvolume '/mnt/install'
   ```
3. Change directory to `/` and unmount the top level of the file system.

```bash
# cd /
# umount /mnt
```

The list of subvolumes now does not include `install`.

```bash
# btrfs subvolume list /
ID 260 top level 5 path root_snapshot_1
```

## 5.13 For More Information About Btrfs

You can find more information about the btrfs file system at [https://btrfs.wiki.kernel.org/index.php/Main_Page](https://btrfs.wiki.kernel.org/index.php/Main_Page).
Chapter 6 The XFS File System

Table of Contents

6.1 About the XFS File System ............................................................................................... 67
   6.1.1 About External XFS Journals .................................................................................... 68
   6.1.2 About XFS Write Barriers ......................................................................................... 69
   6.1.3 About Lazy Counters ............................................................................................... 69
6.2 Installing the XFS Packages ............................................................................................ 69
6.3 Creating an XFS File System ........................................................................................... 69
6.4 Modifying an XFS File System ......................................................................................... 70
6.5 Growing an XFS File System .......................................................................................... 71
6.6 Freezing and Unfreezing an XFS File System ................................................................. 71
6.7 Setting Quotas on an XFS File System .......................................................................... 71
   6.7.1 Setting Project Quotas ............................................................................................ 72
6.8 Backing up and Restoring XFS File Systems ................................................................. 73
6.9 Defragmenting an XFS File System ................................................................................ 75
6.10 Checking and Repairing an XFS File System ................................................................. 75
6.11 For More Information About XFS ................................................................................. 76

This chapter describes how to configure and use the XFS file system.

6.1 About the XFS File System

Note
You must have an Oracle Linux Premier Support account to obtain technical support for XFS with Oracle Linux.

The XFS file system is supported for the Unbreakable Enterprise Kernel Release 2 (2.6.39) and the Unbreakable Enterprise Kernel Release 3 (3.8.13) on the x86_64 architecture only.

XFS is a high-performance journaling file system that was initially created by Silicon Graphics, Inc. for the IRIX operating system and later ported to Linux. The parallel I/O performance of XFS provides high scalability for I/O threads, file system bandwidth, file and file system size, even when the file system spans many storage devices.

A typical use case for XFS is to implement a several-hundred terabyte file system across multiple storage servers, each server consisting of multiple FC-connected disk arrays.

XFS is not supported for use with the root (/) or boot file systems on Oracle Linux.

XFS has a large number of features that make it suitable for deployment in an enterprise-level computing environment that requires the implementation of very large file systems:

• On x86_64 systems, XFS supports a maximum file system size and maximum file size of nearly 8 EB. The maximum supported limit for XFS on Oracle Linux is 100 TB.

• XFS implements journaling for metadata operations, which guarantees the consistency of the file system following loss of power or a system crash. XFS records file system updates asynchronously to a circular buffer (the journal) before it can commit the actual data updates to disk. The journal can be located either internally in the data section of the file system, or externally on a separate device to
reduce contention for disk access. If the system crashes or loses power, it reads the journal when the file system is remounted, and replays any pending metadata operations to ensure the consistency of the file system. The speed of this recovery does not depend on the size of the file system.

- XFS is internally partitioned into allocation groups, which are virtual storage regions of fixed size. Any files and directories that you create can span multiple allocation groups. Each allocation group manages its own set of inodes and free space independently of other allocation groups to provide both scalability and parallelism of I/O operations. If the file system spans many physical devices, allocation groups can optimize throughput by taking advantage of the underlying separation of channels to the storage components.

- XFS is an extent-based file system. To reduce file fragmentation and file scattering, each file's blocks can have variable length extents, where each extent consists of one or more contiguous blocks. XFS's space allocation scheme is designed to efficiently locate free extents that it can use for file system operations. XFS does not allocate storage to the holes in sparse files. If possible, the extent allocation map for a file is stored in its inode. Large allocation maps are stored in a data structure maintained by the allocation group.

- To maximize throughput for XFS file systems that you create on an underlying striped, software or hardware-based array, you can use the `su` and `sw` arguments to the `-d` option of the `mkfs.xfs` command to specify the size of each stripe unit and the number of units per stripe. XFS uses the information to align data, inodes, and journal appropriately for the storage. On `lvm` and `md` volumes and some hardware RAID configurations, XFS can automatically select the optimal stripe parameters for you.

- To reduce fragmentation and increase performance, XFS implements *delayed allocation*, reserving file system blocks for data in the buffer cache, and allocating the block when the operating system flushes that data to disk.

- XFS supports extended attributes for files, where the size of each attribute's value can be up to 64 KB, and each attribute can be allocated to either a *root* or a *user* name space.

- Direct I/O in XFS implements high throughput, non-cashed I/O by performing DMA directly between an application and a storage device, utilising the full I/O bandwidth of the device.

- To support the snapshot facilities that volume managers, hardware subsystems, and databases provide, you can use the `xfs_freeze` command to suspend and resume I/O for an XFS file system. See Section 6.6, “Freezing and Unfreezing an XFS File System”.

- To defragment individual files in an active XFS file system, you can use the `xfs_fsr` command. See Section 6.9, “Defragmenting an XFS File System”.

- To grow an XFS file system, you can use the `xfs_growfs` command. See Section 6.5, “Growing an XFS File System”.

- To back up and restore a live XFS file system, you can use the `xfsdump` and `xfsrestore` commands. See Section 6.8, “Backing up and Restoring XFS File Systems”.

- XFS supports user, group, and project disk quotas on block and inode usage that are initialized when the file system is mounted. Project disk quotas allow you to set limits for individual directory hierarchies within an XFS file system without regard to which user or group has write access to that directory hierarchy.

### 6.1.1 About External XFS Journals

The default location for an XFS journal is on the same block device as the data. As synchronous metadata writes to the journal must complete successfully before any associated data writes can start, such a
layout can lead to disk contention for the typical workload pattern on a database server. To overcome this problem, you can place the journal on a separate physical device with a low-latency I/O path. As the journal typically requires very little storage space, such an arrangement can significantly improve the file system's I/O throughput. A suitable host device for the journal is a solid-state drive (SSD) device or a RAID device with a battery-backed write-back cache.

To reserve an external journal with a specified size when you create an XFS file system, specify the `-l logdev=device,size=size` option to the `mkfs.xfs` command. If you omit the `size` parameter, `mkfs.xfs` selects a journal size based on the size of the file system. To mount the XFS file system so that it uses the external journal, specify the `-o logdev=device` option to the `mount` command.

6.1.2 About XFS Write Barriers

A write barrier assures file system consistency on storage hardware that supports flushing of in-memory data to the underlying device. This ability is particularly important for write operations to an XFS journal that is held on a device with a volatile write-back cache.

By default, an XFS file system is mounted with a write barrier. If you create an XFS file system on a LUN that has a battery-backed, non-volatile cache, using a write barrier degrades I/O performance by requiring data to be flushed more often than necessary. In such cases, you can remove the write barrier by mounting the file system with the `-o nobarrier` option to the `mount` command.

6.1.3 About Lazy Counters

With lazy-counters enabled on an XFS file system, the free-space and inode counters are maintained in parts of the file system other than the superblock. This arrangement can significantly improve I/O performance for application workloads that are metadata intensive.

Lazy counters are enabled by default, but if required, you can disable them by specifying the `-l lazy-count=0` option to the `mkfs.xfs` command.

6.2 Installing the XFS Packages

**Note**

You can also obtain the XFS packages from Public Yum.

To install the XFS packages on a system:

1. Log in to ULN, and subscribe your system to the `ol6_x86_64_latest` channel.
2. On your system, use `yum` to install the `xfsprogs` and `xfsdump` packages:
   
   ```bash
   # yum install xfsprogs xfsdump
   ```

3. If required, use `yum` to install the XFS development and QA packages:

   ```bash
   # yum install xfsprogs-devel xfsprogs-qa-devel
   ```

6.3 Creating an XFS File System

You can use the `mkfs.xfs` command to create an XFS file system, for example.

```bash
# mkfs.xfs /dev/vg0/lv0
meta-data=/dev/vg0/lv0 isize=256 agcount=32, agsize=8473312 blks
```
Modifying an XFS File System

To create an XFS file system with a stripe-unit size of 32 KB and 6 units per stripe, you would specify the `su` and `sw` arguments to the `-d` option, for example:

```
# mkfs.xfs -d su=32k,sw=6 /dev/vg0/lv1
```

For more information, see the `mkfs.xfs(8)` manual page.

### 6.4 Modifying an XFS File System

**Note**
You cannot modify a mounted XFS file system.

You can use the `xfs_admin` command to modify an unmounted XFS file system. For example, you can enable or disable lazy counters, change the file system UUID, or change the file system label.

To display the existing label for an unmounted XFS file system and then apply a new label:

```
# xfs_admin -l /dev/sdb
label = ""
# xfs_admin -L "VideoRecords" /dev/sdb
writing all SBs
new label = "VideoRecords"
```

**Note**
The label can be a maximum of 12 characters in length.

To display the existing UUID and then generate a new UUID:

```
# xfs_admin -u /dev/sdb
UUID = cd4f1cc4-15d8-45f7-afa4-2ae87d1db2ed
# xfs_admin -U generate /dev/sdb
writing all SBs
new UUID = c1b9d5a2-f162-11cf-9ece-0020afc76f16
```

To clear the UUID altogether:

```
# xfs_admin -U nil /dev/sdb
Clearing log and setting UUID
writing all SBs
new UUID = 00000000-0000-0000-0000-000000000000
```

To disable and then re-enable lazy counters:

```
# xfs_admin -c 0 /dev/sdb
Disabling lazy-counters
# xfs_admin -c 1 /dev/sdb
Enabling lazy-counters
```

For more information, see the `mkfs_admin(8)` manual page.
6.5 Growing an XFS File System

Note
You cannot grow an XFS file system that is currently unmounted.
There is currently no command to shrink an XFS file system.

You can use the `xfs_growfs` command to increase the size of a mounted XFS file system if there is space on the underlying devices to accommodate the change. The command does not have any effect on the layout or size of the underlying devices. If necessary, use the underlying volume manager to increase the physical storage that is available. For example, you can use the `vgextend` command to increase the storage that is available to an LVM volume group and `lvextend` to increase the size of the logical volume that contains the file system.

You cannot use the `parted` command to resize a partition that contains an XFS file system. You must instead recreate the partition with a larger size and restore its contents from a backup if you deleted the original partition or from the contents of the original partition if you did not delete it to free up disk space.

For example, to increase the size of `/myxfs1` to 4 TB, assuming a block size of 4 KB:

```
# xfs_growfs -D 1073741824 /myxfs1
```

To increase the size of the file system to the maximum size that the underlying device supports, specify the `-d` option:

```
# xfs_growfs -d /myxfs1
```

For more information, see the `xfs_growfs(8)` manual page.

6.6 Freezing and Unfreezing an XFS File System

If you need to take a hardware-based snapshot of an XFS file system, you can temporarily stop write operations to it.

Note
You do not need to explicitly suspend write operations if you use the `lvcreate` command to take an LVM snapshot.

To freeze and unfreeze an XFS file system, use the `-f` and `-u` options with the `xfs_freeze` command, for example:

```
# xfs_freeze -f /myxfs
# ... Take snapshot of file system ...
# xfs_freeze -u /myxfs
```

Note
You can also use the `xfs_freeze` command with `btrfs`, `ext3`, and `ext4` file systems.

For more information, see the `xfs_freeze(8)` manual page.

6.7 Setting Quotas on an XFS File System

The following table shows the `mount` options that you can specify to enable quotas on an XFS file system:
### Setting Project Quotas

<table>
<thead>
<tr>
<th>Mount Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gqnoenforce</td>
<td>Enable group quotas. Report usage, but do not enforce usage limits.</td>
</tr>
<tr>
<td>gquota</td>
<td>Enable group quotas and enforce usage limits.</td>
</tr>
<tr>
<td>pqnoenforce</td>
<td>Enable project quotas. Report usage, but do not enforce usage limits.</td>
</tr>
<tr>
<td>pquota</td>
<td>Enable project quotas and enforce usage limits.</td>
</tr>
<tr>
<td>uqnoenforce</td>
<td>Enable user quotas. Report usage, but do not enforce usage limits.</td>
</tr>
<tr>
<td>uquota</td>
<td>Enable user quotas and enforce usage limits.</td>
</tr>
</tbody>
</table>

To show the block usage limits and the current usage in the `myxfs` file system for all users, use the `xfs_quota` command:

```
# xfs_quota -x -c 'report -h' /myxfs
```

<table>
<thead>
<tr>
<th>User ID</th>
<th>Used</th>
<th>Soft</th>
<th>Hard</th>
<th>Warn/Grace</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00 [------]</td>
</tr>
<tr>
<td>guest</td>
<td>0</td>
<td>200M</td>
<td>250M</td>
<td>00 [------]</td>
</tr>
</tbody>
</table>

The following forms of the command display the free and used counts for blocks and inodes respectively in the manner of the `df -h` command:

```
# xfs_quota -c 'df -h' /myxfs
```

```
Filesystem       Size   Used  Avail Use% Pathname
/dev/vg0/lv0     200.0G  32.2M  20.0G   1% /myxfs
```

```
# xfs_quota -c 'df -ih' /myxfs
```

```
Filesystem   Inodes   Used   Free Use% Pathname
/dev/vg0/lv0  21.0m      4  21.0m   1% /myxfs
```

If you specify the `-x` option to enter expert mode, you can use subcommands such as `limit` to set soft and hard limits for block and inode usage by an individual user, for example:

```
# xfs_quota -x -c 'limit bsoft=200m bhard=250m isoft=200 ihard=250 guest' /myxfs
```

Of course, this command requires that you mounted the file system with user quotas enabled.

To set limits for a group on an XFS file system that you have mounted with group quotas enabled, specify the `-g` option to `limit`, for example:

```
# xfs_quota -x -c 'limit -g bsoft=5g bhard=6g devgrp' /myxfs
```

For more information, see the `xfs_quota(8)` manual page.

### 6.7.1 Setting Project Quotas

User and group quotas are supported by other file systems, such as `ext4`. The XFS file system additionally allows you to set quotas on individual directory hierarchies in the file system that are known as `managed trees`. Each managed tree is uniquely identified by a `project ID` and an optional `project name`. Being able to control the disk usage of a directory hierarchy is useful if you do not otherwise want to set quota limits for a privileged user (for example, `/var/log`) or if many users or groups have write access to a directory (for example, `/var/tmp`).

To define a project and set quota limits on it:

1. Mount the XFS file system with project quotas enabled:
Backing up and Restoring XFS File Systems

2. Define a unique project ID for the directory hierarchy in the /etc/projects file:

```
# echo project_ID:mountpoint/directory >> /etc/projects
```

For example, to set a project ID of 51 for the directory hierarchy /myxfs/testdir:

```
# echo 51:/myxfs/testdir >> /etc/projects
```

3. Create an entry in the /etc/projid file that maps a project name to the project ID:

```
# echo project_name:project_ID >> /etc/projid
```

For example, to map the project name testproj to the project with ID 51:

```
# echo testproj:51 >> /etc/projid
```

4. Use the project subcommand of xfs_quota to define a managed tree in the XFS file system for the project:

```
# xfs_quota -x -c 'project -s project_name' mountpoint
```

For example, to define a managed tree in the /myxfs file system for the project testproj, which corresponds to the directory hierarchy /myxfs/testdir:

```
# xfs_quota -x -c 'project -s testproj' /myxfs
```

5. Use the limit subcommand to set limits on the disk usage of the project:

```
# xfs_quota -x -c 'limit -p arguments project_name' mountpoint
```

For example, to set a hard limit of 10 GB of disk space for the project testproj:

```
# xfs_quota -x -c 'limit -p bhard=10g testproj' /myxfs
```

For more information, see the `projects(5)`, `projid(5)`, and `xfs_quota(8)` manual pages.

### 6.8 Backing up and Restoring XFS File Systems

The `xfsdump` package contains the `xfsdump` and `xfsrestore` utilities. `xfsdump` examines the files in an XFS file system, determines which files need to be backed up, and copies them to the storage medium. Any backups that you create using `xfsdump` are portable between systems with different endian architectures. `xfsrestore` restores a full or incremental backup of an XFS file system. You can also restore individual files and directory hierarchies from backups.

**Note**

Unlike an LVM snapshot, which immediately creates a sparse clone of a volume, `xfsdump` takes time to make a copy of the file system data.

You can use the `xfsdump` command to create a backup of an XFS file system on a device such as a tape drive, or in a backup file on a different file system. A backup can span multiple physical media that are written on the same device, and you can write multiple backups to the same medium. You can write only a single backup to a file. The command does not overwrite existing XFS backups that it finds on physical
Backing up and Restoring XFS File Systems

media. You must use the appropriate command to erase a physical medium if you need to overwrite any existing backups.

For example, the following command writes a level 0 (base) backup of the XFS file system, /myxfs to the device /dev/st0 and assigns a session label to the backup:

```bash
# xfsdump -l 0 -L "Backup level 0 of /myxfs `date`" -f /dev/st0 /myxfs
```

You can make incremental dumps relative to an existing backup by using the command:

```bash
# xfsdump -l level -L "Backup level level of /myxfs `date`" -f /dev/st0 /myxfs
```

A level 1 backup records only file system changes since the level 0 backup, a level 2 backup records only the changes since the latest level 1 backup, and so on up to level 9.

If you interrupt a backup by typing Ctrl-C and you did not specify the -J option (suppress the dump inventory) to xfsdump, you can resume the dump at a later date by specifying the -R option:

```bash
# xfsdump -R -l 1 -L "Backup level 1 of /myxfs `date`" -f /dev/st0 /myxfs
```

In this example, the backup session label from the earlier, interrupted session is overridden.

You use the xfsrestore command to find out information about the backups you have made of an XFS file system or to restore data from a backup.

The xfsrestore -I command displays information about the available backups, including the session ID and session label. If you want to restore a specific backup session from a backup medium, you can specify either the session ID or the session label.

For example, to restore an XFS file system from a level 0 backup by specifying the session ID:

```bash
# xfsrestore -f /dev/st0 -S c76b3156-c37c-5b6e-7564-a0963ff8ca8f /myxfs
```

If you specify the -r option, you can cumulatively recover all data from a level 0 backup and the higher-level backups that are based on that backup:

```bash
# xfsrestore -r -f /dev/st0 -v silent /myxfs
```

The command searches the archive looking for backups based on the level 0 backup, and prompts you to choose whether you want to restore each backup in turn. After restoring the backup that you select, the command exits. You must run this command multiple times, first selecting to restore the level 0 backup, and then subsequent higher-level backups up to and including the most recent one that you require to restore the file system data.

**Note**

After completing a cumulative restoration of an XFS file system, you should delete the housekeeping directory that xfsrestore creates in the destination directory.

You can recover a selected file or subdirectory contents from the backup medium, as shown in the following example, which recovers the contents of /myxfs/profile/examples to /tmp/profile/examples from the backup with a specified session label:

```bash
# xfsrestore -f /dev/sr0 -L "Backup level 0 of /myxfs Sat Mar 2 14:47:59 GMT 2013" \
-s profile/examples /usr/tmp
```

Alternatively, you can interactively browse a backup by specifying the -i option:

```bash
# xfsrestore -f /dev/sr0 -i
```
Defragmenting an XFS File System

This form of the command allows you browse a backup as though it were a file system. You can change directories, list files, add files, delete files, or extract files from a backup.

To copy the entire contents of one XFS file system to another, you can combine `xfsdump` and `xfsrestore`, using the `-J` option to suppress the usual dump inventory housekeeping that the commands perform:

```
xfsdump -J /myxfs | xfsrestore -J /myxfsclone
```

For more information, see the `xfsdump(8)` and `xfsrestore(8)` manual pages.

### 6.9 Defragmenting an XFS File System

You can use the `xfs_fsr` command to defragment whole XFS file systems or individual files within an XFS file system. As XFS is an extent-based file system, it is usually unnecessary to defragment a whole file system, and doing so is not recommended.

To defragment an individual file, specify the name of the file as the argument to `xfs_fsr`.

```
xfs_fsr pathname
```

If you run the `xfs_fsr` command without any options, the command defragments all currently mounted, writeable XFS file systems that are listed in `/etc/mtab`. For a period of two hours, the command passes over each file system in turn, attempting to defragment the top ten percent of files that have the greatest number of extents. After two hours, the command records its progress in the file `/var/tmp/.fsrlast_xfs`, and it resumes from that point if you run the command again.

For more information, see the `xfs_fsr(8)` manual page.

### 6.10 Checking and Repairing an XFS File System

**Note**

If you have an Oracle Linux Premier Support account and encounter a problem mounting an XFS file system, send a copy of the `/var/log/messages` file to Oracle Support and wait for advice.

If you cannot mount an XFS file system, you can use the `xfs_check` command to check its consistency. Usually, you would only run this command on the device file of an unmounted file system that you believe has a problem. If `xfs_check` displays any output when you do not run it in verbose mode, the file system has an inconsistency.

```
xfscheck device
```

If you can mount the file system and you do not have a suitable backup, you can use `xfsdump` to attempt to back up the existing file system data. However, the command might fail if the file system's metadata has become too corrupted.

You can use the `xfs_repair` command to attempt to repair an XFS file system specified by its device file. The command replays the journal log to fix any inconsistencies that might have resulted from the file system not being cleanly unmounted. Unless the file system has an inconsistency, it is usually not necessary to use the command, as the journal is replayed every time that you mount an XFS file system.

```
xfs_repair device
```

If the journal log has become corrupted, you can reset the log by specifying the `-L` option to `xfs_repair`.

---

75
Warning

Resetting the log can leave the file system in an inconsistent state, resulting in data loss and data corruption. Unless you are experienced in debugging and repairing XFS file systems using `xfs_db`, it is recommended that you instead recreate the file system and restore its contents from a backup.

If you cannot mount the file system or you do not have a suitable backup, running `xfs_repair` is the only viable option unless you are experienced in using `xfs_db`.

`xfs_db` provides an internal command set that allows you to debug and repair an XFS file system manually. The commands allow you to perform scans on the file system, and to navigate and display its data structures. If you specify the `-x` option to enable expert mode, you can modify the data structures.

```
xfs_db [-x] device
```

For more information, see the `xfs_check(8)`, `xfs_db(8)` and `xfs_repair(8)` manual pages, and the help command within `xfs_db`.

**6.11 For More Information About XFS**

Chapter 7 Oracle Cluster File System Version 2

Table of Contents

7.1 About OCFS2 ................................................................. 77
7.2 Installing and Configuring OCFS2 ......................................... 78
  7.2.1 Preparing a Cluster for OCFS2 ........................................ 79
  7.2.2 Configuring the Firewall ............................................. 80
  7.2.3 Configuring the Cluster Software ...................................... 80
  7.2.4 Creating the Configuration File for the Cluster Stack ............... 80
  7.2.5 Configuring the Cluster Stack ........................................ 83
  7.2.6 Configuring the Kernel for Cluster Operation ......................... 84
  7.2.7 Starting and Stopping the Cluster Stack .............................. 85
  7.2.8 Creating OCFS2 volumes ............................................. 85
  7.2.9 Mounting OCFS2 Volumes ............................................ 87
  7.2.10 Querying and Changing Volume Parameters .......................... 87
7.3 Troubleshooting OCFS2 .................................................. 87
  7.3.1 Recommended Tools for Debugging .................................... 88
  7.3.2 Mounting the debugfs File System .................................... 88
  7.3.3 Configuring OCFS2 Tracing ........................................... 88
  7.3.4 Debugging File System Locks .......................................... 89
  7.3.5 Configuring the Behavior of Fenced Nodes ............................ 91
7.4 Use Cases for OCFS2 .................................................... 91
  7.4.1 Load Balancing ..................................................... 91
  7.4.2 Oracle Real Application Cluster (RAC) .............................. 91
  7.4.3 Oracle Databases .................................................. 92
7.5 For More Information About OCFS2 ..................................... 92

This chapter describes how to configure and use the Oracle Cluster File System Version 2 (OCFS2) file system.

7.1 About OCFS2

Oracle Cluster File System version 2 (OCFS2) is a general-purpose, high-performance, high-availability, shared-disk file system intended for use in clusters. It is also possible to mount an OCFS2 volume on a standalone, non-clustered system.

Although it might seem that there is no benefit in mounting ocfs2 locally as compared to alternative file systems such as ext4 or btrfs, you can use the reflink command with OCFS2 to create copy-on-write clones of individual files in a similar way to using the cp --reflink command with the btrfs file system. Typically, such clones allow you to save disk space when storing multiple copies of very similar files, such as VM images or Linux Containers. In addition, mounting a local OCFS2 file system allows you to subsequently migrate it to a cluster file system without requiring any conversion.

Almost all applications can use OCFS2 as it provides local file-system semantics. Applications that are cluster-aware can use cache-coherent parallel I/O from multiple cluster nodes to balance activity across the cluster, or they can use of the available file-system functionality to fail over and run on another node in the event that a node fails. The following examples typify some use cases for OCFS2:

- Oracle VM to host shared access to virtual machine images.
- Oracle VM and VirtualBox to allow Linux guest machines to share a file system.
Installing and Configuring OCFS2

• Oracle Real Application Cluster (RAC) in database clusters.
• Oracle E-Business Suite in middleware clusters.

OCFS2 has a large number of features that make it suitable for deployment in an enterprise-level computing environment:

• Support for ordered and write-back data journaling that provides file system consistency in the event of power failure or system crash.

• Block sizes ranging from 512 bytes to 4 KB, and file-system cluster sizes ranging from 4 KB to 1 MB (both in increments of powers of 2). The maximum supported volume size is 16 TB, which corresponds to a cluster size of 4 KB. A volume size as large as 4 PB is theoretically possible for a cluster size of 1 MB, although this limit has not been tested.

• Extent-based allocations for efficient storage of very large files.

• Optimized allocation support for sparse files, inline-data, unwritten extents, hole punching, reflinks, and allocation reservation for high performance and efficient storage.

• Indexing of directories to allow efficient access to a directory even if it contains millions of objects.

• Metadata checksums for the detection of corrupted inodes and directories.

• Extended attributes to allow an unlimited number of name:value pairs to be attached to file system objects such as regular files, directories, and symbolic links.

• Advanced security support for POSIX ACLs and SELinux in addition to the traditional file-access permission model.

• Support for user and group quotas.

• Support for heterogeneous clusters of nodes with a mixture of 32-bit and 64-bit, little-endian (x86, x86_64, ia64) and big-endian (ppc64) architectures.

• An easy-to-configure, in-kernel cluster-stack (O2CB) with a distributed lock manager (DLM), which manages concurrent access from the cluster nodes.

• Support for buffered, direct, asynchronous, splice and memory-mapped I/O.

• A tool set that uses similar parameters to the ext3 file system.

### 7.2 Installing and Configuring OCFS2

The procedures in the following sections describe how to set up a cluster to use OCFS2.

• Section 7.2.1, “Preparing a Cluster for OCFS2”
• Section 7.2.2, “Configuring the Firewall”
• Section 7.2.3, “Configuring the Cluster Software”
• Section 7.2.4, “Creating the Configuration File for the Cluster Stack”
• Section 7.2.5, “Configuring the Cluster Stack”
• Section 7.2.6, “Configuring the Kernel for Cluster Operation”
• Section 7.2.7, “Starting and Stopping the Cluster Stack”
• Section 7.2.9, “Mounting OCFS2 Volumes”

7.2.1 Preparing a Cluster for OCFS2

For best performance, each node in the cluster should have at least two network interfaces. One interface is connected to a public network to allow general access to the systems. The other interface is used for private communication between the nodes; the *cluster heartbeat* that determines how the cluster nodes coordinate their access to shared resources and how they monitor each other’s state. These interface must be connected via a network switch. Ensure that all network interfaces are configured and working before continuing to configure the cluster.

You have a choice of two cluster heartbeat configurations:

• Local heartbeat thread for each shared device. In this mode, a node starts a heartbeat thread when it mounts an OCFS2 volume and stops the thread when it unmounts the volume. This is the default heartbeat mode. There is a large CPU overhead on nodes that mount a large number of OCFS2 volumes as each mount requires a separate heartbeat thread. A large number of mounts also increases the risk of a node fencing itself out of the cluster due to a heartbeat I/O timeout on a single mount.

• Global heartbeat on specific shared devices. You can configure any OCFS2 volume as a global heartbeat device provided that it occupies a whole disk device and not a partition. In this mode, the heartbeat to the device starts when the cluster comes online and stops when the cluster goes offline. This mode is recommended for clusters that mount a large number of OCFS2 volumes. A node fences itself out of the cluster if a heartbeat I/O timeout occurs on more than half of the global heartbeat devices. To provide redundancy against failure of one of the devices, you should therefore configure at least three global heartbeat devices.

Figure 7.1 shows a cluster of four nodes connected via a network switch to a LAN and a network storage server. The nodes and the storage server are also connected via a switch to a private network that they use for the local cluster heartbeat.

**Figure 7.1 Cluster Configuration Using a Private Network**

It is possible to configure and use OCFS2 without using a private network but such a configuration increases the probability of a node fencing itself out of the cluster due to an I/O heartbeat timeout.
7.2.2 Configuring the Firewall

Configure or disable the firewall on each node to allow access on the interface that the cluster will use for private cluster communication. By default, the cluster uses both TCP and UDP over port 7777.

To allow incoming TCP connections and UDP datagrams on port 7777 from the private network, use the following commands:

```
# iptables -I INPUT -s subnet_addr/prefix_length -p tcp \
       --state NEW -m tcp --dport 7777 -j ACCEPT
# iptables -I INPUT -s subnet_addr/prefix_length -p udp \
       --dport 7777 -j ACCEPT
# service iptables save
```

where `subnet_addr/prefix_length` specifies the network address of the private network, for example `10.0.1.0/24`.

7.2.3 Configuring the Cluster Software

Ideally, each node should be running the same version of the OCFS2 software and a compatible version of the Oracle Linux Unbreakable Enterprise Kernel (UEK). It is possible for a cluster to run with mixed versions of the OCFS2 and UEK software, for example, while you are performing a rolling update of a cluster. The cluster node that is running the lowest version of the software determines the set of usable features.

Use `yum` to install or upgrade the following packages to the same version on each node:

- `kernel-uek`
- `ocfs2-tools`

**Note**

If you want to use the global heartbeat feature, you must install `ocfs2-tools-1.8.0-11` or later.

7.2.4 Creating the Configuration File for the Cluster Stack

You can create the configuration file by using the `o2cb` command or a text editor.

To configure the cluster stack by using the `o2cb` command:

1. Use the following command to create a cluster definition.

```
# o2cb add-cluster cluster_name
```

For example, to define a cluster named `mycluster` with four nodes:

```
# o2cb add-cluster mycluster
```

The command creates the configuration file `/etc/ocfs2/cluster.conf` if it does not already exist.

2. For each node, use the following command to define the node.

```
# o2cb add-node cluster_name node_name --ip ip_address
```

The name of the node must be same as the value of system's `HOSTNAME` that is configured in `/etc/sysconfig/network`. The IP address is the one that the node will use for private communication in the cluster.
Creating the Configuration File for the Cluster Stack

For example, to define a node named `node0` with the IP address 10.1.0.100 in the cluster `mycluster`:

```
# o2cb add-node mycluster node0 --ip 10.1.0.100
```

3. If you want the cluster to use global heartbeat devices, use the following commands.

```
# o2cb add-heartbeat cluster_name device1
# o2cb heartbeat-mode cluster_name global
```

**Note**
You must configure global heartbeat to use whole disk devices. You cannot configure a global heartbeat device on a disk partition.

For example, to use `/dev/sdd`, `/dev/sdg`, and `/dev/sdj` as global heartbeat devices:

```
# o2cb add-heartbeat mycluster /dev/sdd
# o2cb add-heartbeat mycluster /dev/sdg
# o2cb add-heartbeat mycluster /dev/sdj
# o2cb heartbeat-mode mycluster global
```

4. Copy the cluster configuration file `/etc/ocfs2/cluster.conf` to each node in the cluster.

**Note**
Any changes that you make to the cluster configuration file do not take effect until you restart the cluster stack.

The following sample configuration file `/etc/ocfs2/cluster.conf` defines a 4-node cluster named `mycluster` with a local heartbeat.

```
node:
  name = node0
  cluster = mycluster
  number = 0
  ip_address = 10.1.0.100
  ip_port = 7777

node:
  name = node1
  cluster = mycluster
  number = 1
  ip_address = 10.1.0.101
  ip_port = 7777

node:
  name = node2
  cluster = mycluster
  number = 2
  ip_address = 10.1.0.102
  ip_port = 7777

node:
  name = node3
  cluster = mycluster
  number = 3
  ip_address = 10.1.0.103
  ip_port = 7777

cluster:
```
Creating the Configuration File for the Cluster Stack

name = mycluster
heartbeat_mode = local
node_count = 4

If you configure your cluster to use a global heartbeat, the file also include entries for the global heartbeat devices.

node:
  name = node0
  cluster = mycluster
  number = 0
  ip_address = 10.1.0.100
  ip_port = 7777

node:
  name = node1
  cluster = mycluster
  number = 1
  ip_address = 10.1.0.101
  ip_port = 7777

node:
  name = node2
  cluster = mycluster
  number = 2
  ip_address = 10.1.0.102
  ip_port = 7777

node:
  name = node3
  cluster = mycluster
  number = 3
  ip_address = 10.1.0.103
  ip_port = 7777

cluster:
  name = mycluster
  heartbeat_mode = global
  node_count = 4

heartbeat:
  cluster = mycluster
  region = 7DA5015346C245E6A41AA85E2E7EA3CF

heartbeat:
  cluster = mycluster
  region = 4F9FBB0D9B6341729F21A8891B9A05BD

heartbeat:
  cluster = mycluster
  region = B423C7EEE9FC426790FC411972C91CC3

The cluster heartbeat mode is now shown as global, and the heartbeat regions are represented by the UUIDs of their block devices.

If you edit the configuration file manually, ensure that you use the following layout:

• The cluster:, heartbeat:, and node: headings must start in the first column.
• Each parameter entry must be indented by one tab space.
• A blank line must separate each section that defines the cluster, a heartbeat device, or a node.
7.2.5 Configuring the Cluster Stack

To configure the cluster stack:

1. Run the following command on each node of the cluster:

   ```
   # service o2cb configure
   ```

   The following table describes the values for which you are prompted.

<table>
<thead>
<tr>
<th>Prompt</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load O2CB driver on boot (y/n)</td>
<td>Whether the cluster stack driver should be loaded at boot time. The default response is n.</td>
</tr>
<tr>
<td>Cluster stack backing O2CB</td>
<td>The name of the cluster stack service. The default and usual response is o2cb.</td>
</tr>
<tr>
<td>Cluster to start at boot (Enter &quot;none&quot; to clear)</td>
<td>Enter the name of your cluster that you defined in the cluster configuration file, /etc/ocfs2/cluster.conf.</td>
</tr>
<tr>
<td>Specify heartbeat dead threshold (&gt;=7)</td>
<td>The number of 2-second heartbeats that must elapse without response before a node is considered dead. To calculate the value to enter, divide the required threshold time period by 2 and add 1. For example, to set the threshold time period to 120 seconds, enter a value of 61. The default value is 31, which corresponds to a threshold time period of 60 seconds.</td>
</tr>
<tr>
<td>Specify network idle timeout in ms (&gt;=5000)</td>
<td>The time in milliseconds that must elapse before a network connection is considered dead. The default value is 30,000 milliseconds.</td>
</tr>
<tr>
<td>Specify network keepalive delay in ms (&gt;=1000)</td>
<td>The maximum delay in milliseconds between sending keepalive packets to another node. The default and recommended value is 2,000 milliseconds.</td>
</tr>
<tr>
<td>Specify network reconnect delay in ms (&gt;=2000)</td>
<td>The minimum delay in milliseconds between reconnection attempts if a network connection goes down. The default and recommended value is 2,000 milliseconds.</td>
</tr>
</tbody>
</table>

To verify the settings for the cluster stack, enter the `service o2cb status` command:

```
# service o2cb status
Driver for "configfs": Loaded
Filesystem "configfs": Mounted
Stack glue driver: Loaded
Stack plugin "o2cb": Loaded
Driver for "ocfs2_dlmfs": Loaded
```
Configuring the Kernel for Cluster Operation

In this example, the cluster is online and is using local heartbeat mode. If no volumes have been configured, the O2CB heartbeat is shown as `Not active` rather than `Active`.

The next example shows the command output for an online cluster that is using three global heartbeat devices:

```
# service o2cb status
Driver for "configfs": Loaded
Filesystem "configfs": Mounted
Stack glue driver: Loaded
Stack plugin "o2cb": Loaded
Driver for "ocfs2_dlmfs": Loaded
Filesystem "ocfs2_dlmfs": Mounted
Checking O2CB cluster "mycluster": Online
Heartbeat dead threshold: 61
Network idle timeout: 30000
Network keepalive delay: 2000
Network reconnect delay: 2000
Heartbeat mode: Global
Checking O2CB heartbeat: Active
```

2. Configure the `o2cb` and `ocfs2` services so that they start at boot time after networking is enabled:

```
# chkconfig o2cb on
# chkconfig ocfs2 on
```

These settings allow the node to mount OCFS2 volumes automatically when the system starts.

7.2.6 Configuring the Kernel for Cluster Operation

For the correct operation of the cluster, you must configure the kernel settings shown in the following table:

<table>
<thead>
<tr>
<th>Kernel Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>panic</code></td>
<td>Specifies the number of seconds after a panic before a system will automatically reset itself.</td>
</tr>
<tr>
<td></td>
<td>If the value is 0, the system hangs, which allows you to collect detailed information about the panic for troubleshooting. This is the default value.</td>
</tr>
<tr>
<td></td>
<td>To enable automatic reset, set a non-zero value. If you require a memory image (vmcore), allow enough time for Kdump to create this image. The suggested value is 30 seconds, although large systems will require a longer time.</td>
</tr>
<tr>
<td><code>panic_on_oops</code></td>
<td>Specifies that a system must panic if a kernel oops occurs. If a kernel thread required for cluster operation crashes, the system must reset itself. Otherwise, another node might not be able to tell whether a node is slow to respond or unable to respond, causing cluster operations to hang.</td>
</tr>
</tbody>
</table>

On each node, enter the following commands to set the recommended values for `panic` and `panic_on_oops`:
Starting and Stopping the Cluster Stack

To make the change persist across reboots, add the following entries to the `/etc/sysctl.conf` file:

```bash
# Define panic and panic_on_oops for cluster operation
kernel.panic = 30
kernel.panic_on_oops = 1
```

7.2.7 Starting and Stopping the Cluster Stack

The following table shows the commands that you can use to perform various operations on the cluster stack.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service o2cb status</td>
<td>Check the status of the cluster stack.</td>
</tr>
<tr>
<td>service o2cb online</td>
<td>Start the cluster stack.</td>
</tr>
<tr>
<td>service o2cb offline</td>
<td>Stop the cluster stack.</td>
</tr>
<tr>
<td>service o2cb unload</td>
<td>Unload the cluster stack.</td>
</tr>
</tbody>
</table>

7.2.8 Creating OCFS2 volumes

You can use the `mkfs.ocfs2` command to create an OCFS2 volume on a device. If you want to label the volume and mount it by specifying the label, the device must correspond to a partition. You cannot mount an unpartitioned disk device by specifying a label. The following table shows the most useful options that you can use when creating an OCFS2 volume.

<table>
<thead>
<tr>
<th>Command Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-b block-size</code></td>
<td>Specifies the unit size for I/O transactions to and from the file system, and the size of inode and extent blocks. The supported block sizes are 512 (512 bytes), 1K, 2K, and 4K. The default and recommended block size is 4K (4 kilobytes).</td>
</tr>
<tr>
<td><code>--block-size</code> block-size</td>
<td>Specifies the unit size for space used to allocate file data. The supported cluster sizes are 4K, 8K, 16K, 32K, 64K, 128K, 256K, 512K, and 1M (1 megabyte). The default cluster size is 4K (4 kilobytes).</td>
</tr>
<tr>
<td><code>-C cluster-size</code></td>
<td>Specifies the unit size for space used to allocate file data. The supported cluster sizes are 4K, 8K, 16K, 32K, 64K, 128K, 256K, 512K, and 1M (1 megabyte). The default cluster size is 4K (4 kilobytes).</td>
</tr>
<tr>
<td><code>--cluster-size</code> cluster-size</td>
<td>Allows you select a set of file-system features:</td>
</tr>
<tr>
<td><code>--fs-feature-level=feature-level</code></td>
<td>Enables support for the sparse files, unwritten extents, and inline data features.</td>
</tr>
<tr>
<td><code>default</code></td>
<td>Enables only those features that are understood by older versions of OCFS2.</td>
</tr>
<tr>
<td><code>max-features</code></td>
<td>Enables all features that OCFS2 currently supports.</td>
</tr>
<tr>
<td><code>--fs_features=feature</code></td>
<td>Allows you to enable or disable individual features such as support for sparse files, unwritten extents, and backup superblocks. For more information, see the <code>mkfs.ocfs2(8)</code> manual page.</td>
</tr>
</tbody>
</table>
| `-J size=journal-size`    | Specifies the size of the write-ahead journal. If not specified, the size is determined from the file system usage type that you specify to the -
Creating OCFS2 volumes

<table>
<thead>
<tr>
<th>Command Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--journal-options size=journal-size</td>
<td>T option, and, otherwise, from the volume size. The default size of the journal is 64M (64 MB) for datafiles, 256M (256 MB) for mail, and 128M (128 MB) for vmstore.</td>
</tr>
<tr>
<td>-L volume-label</td>
<td>Specifies a descriptive name for the volume that allows you to identify it easily on different cluster nodes.</td>
</tr>
<tr>
<td>--label volume-label</td>
<td></td>
</tr>
<tr>
<td>-N number</td>
<td>Determines the maximum number of nodes that can concurrently access a volume, which is limited by the number of node slots for system files such as the file-system journal. For best performance, set the number of node slots to at least twice the number of nodes. If you subsequently increase the number of node slots, performance can suffer because the journal will no longer be contiguously laid out on the outer edge of the disk platter.</td>
</tr>
<tr>
<td>--node-slots number</td>
<td></td>
</tr>
<tr>
<td>-T file-system-usage-type</td>
<td>Specifies the type of usage for the file system:</td>
</tr>
<tr>
<td>datafiles</td>
<td>Database files are typically few in number, fully allocated, and relatively large. Such files require few metadata changes, and do not benefit from having a large journal.</td>
</tr>
<tr>
<td>mail</td>
<td>Mail server files are typically many in number, and relatively small. Such files require many metadata changes, and benefit from having a large journal.</td>
</tr>
<tr>
<td>vmstore</td>
<td>Virtual machine image files are typically few in number, sparsely allocated, and relatively large. Such files require a moderate number of metadata changes and a medium sized journal.</td>
</tr>
</tbody>
</table>

For example, create an OCFS2 volume on /dev/sdc1 labeled as myvol using all the default settings for generic usage (4 KB block and cluster size, eight node slots, a 256 MB journal, and support for default file-system features).

```sh
# mkfs.ocfs2 -L "myvol" /dev/sdc1
```

Create an OCFS2 volume on /dev/sdd2 labeled as dbvol for use with database files. In this case, the cluster size is set to 128 KB and the journal size to 32 MB.

```sh
# mkfs.ocfs2 -L "dbvol" -T datafiles /dev/sdd2
```

Create an OCFS2 volume on /dev/sde1 with a 16 KB cluster size, a 128 MB journal, 16 node slots, and support enabled for all features except reffcount trees.

```sh
# mkfs.ocfs2 -C 16K -J size=128M -N 16 --fs-feature-level=max-features --fs-features=noreffcount /dev/sde1
```

Note

Do not create an OCFS2 volume on an LVM logical volume. LVM is not cluster-aware.
Mounting OCFS2 Volumes

You cannot change the block and cluster size of an OCFS2 volume after it has been created. You can use the `tunefs.ocfs2` command to modify other settings for the file system with certain restrictions. For more information, see the `tunefs.ocfs2(8)` manual page.

If you intend the volume to store database files, do not specify a cluster size that is smaller than the block size of the database.

The default cluster size of 4 KB is not suitable if the file system is larger than a few gigabytes. The following table suggests minimum cluster size settings for different file system size ranges:

<table>
<thead>
<tr>
<th>File System Size</th>
<th>Suggested Minimum Cluster Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GB - 10 GB</td>
<td>8K</td>
</tr>
<tr>
<td>10 GB - 100 GB</td>
<td>16K</td>
</tr>
<tr>
<td>100 GB - 1 TB</td>
<td>32K</td>
</tr>
<tr>
<td>1 TB - 10 TB</td>
<td>64K</td>
</tr>
<tr>
<td>10 TB - 16 TB</td>
<td>128K</td>
</tr>
</tbody>
</table>

7.2.9 Mounting OCFS2 Volumes

As shown in the following example, specify the `_netdev` option in `/etc/fstab` if you want the system to mount an OCFS2 volume at boot time after networking is started, and to unmount the file system before networking is stopped.

```
myocfs2vol  /dbvol1  ocfs2     _netdev,defaults  0 0
```

**Note**

The file system will not mount unless you have enabled the `o2cb` and `ocfs2` services to start after networking is started. See Section 7.2.5, "Configuring the Cluster Stack".

7.2.10 Querying and Changing Volume Parameters

You can use the `tunefs.ocfs2` command to query or change volume parameters. For example, to find out the label, UUID and the number of node slots for a volume:

```
# tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots =%N" /dev/sdb
Label = myvol
UUID = CBB8D5E0C169497C8B52A0FD555C7A3E
NumSlots = 4
```

Generate a new UUID for a volume:

```
# tunefs.ocfs2 -U /dev/sda
# tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots =%N" /dev/sdb
Label = myvol
UUID = 48E56A2BBAB34A9EB1BE832B3C36AB5C
NumSlots = 4
```

7.3 Troubleshooting OCFS2

The following sections describes some techniques that you can use for investigating any problems that you encounter with OCFS2.
7.3.1 Recommended Tools for Debugging

To you want to capture an oops trace, it is recommended that you set up netconsole on the nodes.

If you want to capture the DLM's network traffic between the nodes, you can use tcpdump. For example, to capture TCP traffic on port 7777 for the private network interface eth1, you could use a command such as the following:

```
# tcpdump -i eth1 -C 10 -W 15 -s 10000 -Sw /tmp/`hostname -s`_tcpdump.log \\
-ttt 'port 7777' &
```

You can use the `debugfs.ocfs2` command, which is similar in behavior to the `debugfs` command for the ext3 file system, and allows you to trace events in the OCFS2 driver, determine lock statuses, walk directory structures, examine inodes, and so on.

For more information, see the `debugfs.ocfs2(8)` manual page.

The `o2image` command saves an OCFS2 file system's metadata (including information about inodes, file names, and directory names) to an image file on another file system. As the image file contains only metadata, it is much smaller than the original file system. You can use `debugfs.ocfs2` to open the image file, and analyze the file system layout to determine the cause of a file system corruption or performance problem.

For example, the following command creates the image `/tmp/sda2.img` from the OCFS2 file system on the device `/dev/sda2`:

```
# o2image /dev/sda2 /tmp/sda2.img
```

For more information, see the `o2image(8)` manual page.

7.3.2 Mounting the debugfs File System

OCFS2 uses the `debugfs` file system to allow access from user space to information about its in-kernel state. You must mount the `debugfs` file system to be able to use the `debugfs.ocfs2` command.

To mount the `debugfs` file system, add the following line to `/etc/fstab`:

```
defaults
```

and run the `mount -a` command.

7.3.3 Configuring OCFS2 Tracing

The following table shows some of the commands that are useful for tracing problems in OCFS2.
### Command Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debugfs.ocfs2 -l</code></td>
<td>List all trace bits and their statuses.</td>
</tr>
<tr>
<td><code>debugfs.ocfs2 -l SUPER allow</code></td>
<td>Enable tracing for the superblock.</td>
</tr>
<tr>
<td><code>debugfs.ocfs2 -l SUPER off</code></td>
<td>Disable tracing for the superblock.</td>
</tr>
<tr>
<td><code>debugfs.ocfs2 -l SUPER deny</code></td>
<td>Disallow tracing for the superblock, even if implicitly enabled by another tracing mode setting.</td>
</tr>
<tr>
<td><code>debugfs.ocfs2 -l HEARTBEAT ENTRY EXIT allow</code></td>
<td>Enable heartbeat tracing.</td>
</tr>
<tr>
<td><code>debugfs.ocfs2 -l HEARTBEAT off ENTRY EXIT deny</code></td>
<td>Disable heartbeat tracing. ENTRY and EXIT are set to deny as they exist in all trace paths.</td>
</tr>
<tr>
<td><code>debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow</code></td>
<td>Enable tracing for the file system.</td>
</tr>
<tr>
<td><code>debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE allow</code></td>
<td>Disable tracing for the file system.</td>
</tr>
<tr>
<td><code>debugfs.ocfs2 -l ENTRY EXIT DLM DLM_THREAD allow</code></td>
<td>Enable tracing for the DLM.</td>
</tr>
<tr>
<td><code>debugfs.ocfs2 -l ENTRY EXIT deny DLM DLM_THREAD allow</code></td>
<td>Disable tracing for the DLM.</td>
</tr>
</tbody>
</table>

One method for obtaining a trace is to enable the trace, sleep for a short while, and then disable the trace. As shown in the following example, to avoid seeing unnecessary output, you should reset the trace bits to their default settings after you have finished.

```bash
# debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow && sleep 10 &&
# debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

To limit the amount of information displayed, enable only the trace bits that you believe are relevant to understanding the problem.

If you believe a specific file system command, such as `mv`, is causing an error, the following example shows the commands that you can use to help you trace the error.

```bash
# debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow
# mv source destination & CMD_PID=$(jobs -p %)
# echo CMD_PID=$CMD_PID &
# debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

As the trace is enabled for all mounted OCFS2 volumes, knowing the correct process ID can help you to interpret the trace.

For more information, see the `debugfs.ocfs2(8)` manual page.

### 7.3.4 Debugging File System Locks

If an OCFS2 volume hangs, you can use the following steps to help you determine which locks are busy and the processes that are likely to be holding the locks.

1. Mount the debug file system.
Debugging File System Locks

1. Mount the debugfs file system:

```bash
# mount -t debugfs debugfs /sys/kernel/debug
```

2. Dump the lock statuses for the file system device (e.g., `/dev/sdx1` in this example):

```bash
# echo "fs_locks" | debugfs.ocfs2 /dev/sdx1 > /tmp/fslocks
```

```
Lockres: M00000000000006672078b84822 Mode: Protected Read
Flags: Initialized Attached
RO Holders: 0 EX Holders: 0
Pending Action: None Pending Unlock Action: None
Requested Mode: Protected Read Blocking Mode: Invalid
```

The **Lockres** field is the lock name used by the DLM. The lock name is a combination of a lock-type identifier, an inode number, and a generation number. The following table shows the possible lock types:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Lock Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>File data.</td>
</tr>
<tr>
<td>M</td>
<td>Metadata.</td>
</tr>
<tr>
<td>R</td>
<td>Rename.</td>
</tr>
<tr>
<td>S</td>
<td>Superblock.</td>
</tr>
<tr>
<td>W</td>
<td>Read-write.</td>
</tr>
</tbody>
</table>

3. Use the **Lockres** value to obtain the inode number and generation number for the lock:

```bash
# echo "stat <M00000000000006672078b84822>" | debugfs.ocfs2 -n /dev/sdx1
```

```
Inode: 419616   Mode: 0666   Generation: 2025343010 (0x78b84822)
```

4. Determine the file system object to which the inode number relates by using the following command:

```bash
# echo "locate <419616>" | debugfs.ocfs2 -n /dev/sdx1
```

```
419616 /linux-2.6.15/arch/i386/kernel/semaphore.c
```

5. Obtain the lock names that are associated with the file system object:

```bash
# echo "encode /linux-2.6.15/arch/i386/kernel/semaphore.c" |
debugfs.ocfs2 -n /dev/sdx1
```

```
M00000000000006672078b84822 D00000000000006672078b84822 W00000000000006672078b84822
```

In this example, a metadata lock, a file data lock, and a read-write lock are associated with the file system object.

6. Determine the DLM domain of the file system:

```bash
# echo "stats" | debugfs.ocfs2 -n /dev/sdx1 | grep UUID: | while read a b ; do echo $b ; done
```

```
82DA8137A49A47E4B187F74E09FBBB4B
```

7. Use the values of the DLM domain and the lock name with the following command, which enables debugging for the DLM:

```bash
# echo R 82DA8137A49A47E4B187F74E09FBBB4B M00000000000006672078b84822 > /proc/fs/ocfs2_dlm/debug
```

8. Examine the debug messages:

```bash
# dmesg | tail
```

```
struct dlm_ctxt: 82DA8137A49A47E4B187F74E09FBBB4B, node=3, key=965960985
lockres: M00000000000006672078b84822, owner=1, state=0 last used: 0,
on purge list: no granted queue:
```
The DLM supports 3 lock modes: no lock \((\text{type}=0)\), protected read \((\text{type}=3)\), and exclusive \((\text{type}=5)\). In this example, the lock is mastered by node 1 \((\text{owner}=1)\) and node 3 has been granted a protected-read lock on the file-system resource.

9. Run the following command, and look for processes that are in an uninterruptable sleep state as shown by the \(D\) flag in the \textsc{Stat} column.

```bash
# ps -e -o pid,stat,comm,wchan=WIDE-WCHAN-COLUMN
```

At least one of the processes that are in the uninterruptable sleep state will be responsible for the hang on the other node.

If a process is waiting for I/O to complete, the problem could be anywhere in the I/O subsystem from the block device layer through the drivers to the disk array. If the hang concerns a user lock \((\text{flock}())\), the problem could lie in the application. If possible, kill the holder of the lock. If the hang is due to lack of memory or fragmented memory, you can free up memory by killing non-essential processes. The most immediate solution is to reset the node that is holding the lock. The DLM recovery process can then clear all the locks that the dead node owned, so letting the cluster continue to operate.

### 7.3.5 Configuring the Behavior of Fenced Nodes

If a node with a mounted OCFS2 volume believes that it is no longer in contact with the other cluster nodes, it removes itself from the cluster in a process termed fencing. Fencing prevents other nodes from hanging when they try to access resources held by the fenced node. By default, a fenced node restarts instead of panicking so that it can quickly rejoin the cluster. Under some circumstances, you might want a fenced node to panic instead of restarting. For example, you might want to use \texttt{netconsole} to view the oops stack trace or to diagnose the cause of frequent reboots. To configure a node to panic when it next fences, run the following command on the node after the cluster starts:

```bash
# echo panic > /sys/kernel/config/cluster/cluster_name/fence_method
```

where \texttt{cluster_name} is the name of the cluster. To set the value after each reboot of the system, add this line to \texttt{/etc/rc.local}. To restore the default behavior, use the value \texttt{reset} instead of \texttt{panic}.

### 7.4 Use Cases for OCFS2

The following sections describe some typical use cases for OCFS2.

#### 7.4.1 Load Balancing

You can use OCFS2 nodes to share resources between client systems. For example, the nodes could export a shared file system by using Samba or NFS. To distribute service requests between the nodes, you can use round-robin DNS, a network load balancer, or specify which node should be used on each client.

#### 7.4.2 Oracle Real Application Cluster (RAC)

Oracle RAC uses its own cluster stack, Cluster Synchronization Services (CSS). You can use O2CB in conjunction with CSS, but you should note that each stack is configured independently for timeouts, nodes, and other cluster settings. You can use OCFS2 to host the voting disk files and the Oracle cluster registry (OCR), but not the grid infrastructure user’s home, which must exist on a local file system on each node.
As both CSS and O2CB use the lowest node number as a tie breaker in quorum calculations, you should ensure that the node numbers are the same in both clusters. If necessary, edit the O2CB configuration file `/etc/ocfs2/cluster.conf` to make the node numbering consistent, and update this file on all nodes. The change takes effect when the cluster is restarted.

### 7.4.3 Oracle Databases

Specify the `noatime` option when mounting volumes that host Oracle datafiles, control files, redo logs, voting disk, and OCR. The `noatime` option disables unnecessary updates to the access time on the inodes.

Specify the `nointr` mount option to prevent signals interrupting I/O transactions that are in progress.

By default, the `init.ora` parameter `filesystemio_options` directs the database to perform direct I/O to the Oracle datafiles, control files, and redo logs. You should also specify the `datavolume` mount option for the volumes that contain the voting disk and OCR. Do not specify this option for volumes that host the Oracle user's home directory or Oracle E-Business Suite.

To avoid database blocks becoming fragmented across a disk, ensure that the file system cluster size is at least as big as the database block size, which is typically 8KB. If you specify the file system usage type as `datafiles` to the `mkfs.ocfs2` command, the file system cluster size is set to 128KB.

To allow multiple nodes to maximize throughput by concurrently streaming data to an Oracle datafile, OCFS2 deviates from the POSIX standard by not updating the modification time (`mtime`) on the disk when performing non-extending direct I/O writes. The value of `mtime` is updated in memory, but OCFS2 does not write the value to disk unless an application extends or truncates the file, or performs a operation to change the file metadata, such as using the `touch` command. This behavior leads to results in different nodes reporting different time stamps for the same file. You can use the following command to view the on-disk timestamp of a file:

```
# debugfs.ocfs2 -R "stat /file_path" device | grep "mtime:"
```

### 7.5 For More Information About OCFS2

You can find more information about OCFS2 at [https://oss.oracle.com/projects/ocfs2/documentation/](https://oss.oracle.com/projects/ocfs2/documentation/).
8.1 About cgroups

A cgroup is a collection of processes (tasks) that you bind together by applying a set of criteria that control the cgroup's access to system resources. You can create a hierarchy of cgroups, in which child cgroups inherit its characteristics from the parent cgroup. You can use cgroups to manage processes in the following ways:

- Limit the CPU, I/O, and memory resources that are available to a group.
- Change the priority of a group relative to other groups.
- Measure a group's resource usage for accounting and billing purposes.
- Isolate a group's files, processes, and network connections from other groups.
- Freeze a group to allow you to create a checkpoint.

You can create and manage cgroups in the following ways:

- By editing the cgroup configuration file `/etc/cgconfig.conf`.
- By using cgroups commands such as `cgcreate`, `cgclassify`, and `cgexec`.

This chapter describes how to use Control Groups (cgroups) to manage the resource utilization of sets of processes.

## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 About cgroups</td>
<td>93</td>
</tr>
<tr>
<td>8.2 Subsystems</td>
<td>94</td>
</tr>
<tr>
<td>8.2.1 blkio Parameters</td>
<td>94</td>
</tr>
<tr>
<td>8.2.2 cpu Parameters</td>
<td>96</td>
</tr>
<tr>
<td>8.2.3 cpaucnt Parameters</td>
<td>96</td>
</tr>
<tr>
<td>8.2.4 cpuset Parameters</td>
<td>97</td>
</tr>
<tr>
<td>8.2.5 devices Parameters</td>
<td>98</td>
</tr>
<tr>
<td>8.2.6 freezer Parameter</td>
<td>99</td>
</tr>
<tr>
<td>8.2.7 memory Parameters</td>
<td>99</td>
</tr>
<tr>
<td>8.2.8 net_cls Parameter</td>
<td>102</td>
</tr>
<tr>
<td>8.3 Enabling the cgconfig Service</td>
<td>102</td>
</tr>
<tr>
<td>8.4 Enabling PAM to Work with cgroup Rules</td>
<td>102</td>
</tr>
<tr>
<td>8.5 Restarting the cgconfig Service</td>
<td>103</td>
</tr>
<tr>
<td>8.6 About the cgroups Configuration File</td>
<td>103</td>
</tr>
<tr>
<td>8.7 About the cgroup Rules Configuration File</td>
<td>105</td>
</tr>
<tr>
<td>8.8 Displaying and Setting Subsystem Parameters</td>
<td>105</td>
</tr>
<tr>
<td>8.9 Use Cases for cgroups</td>
<td>106</td>
</tr>
<tr>
<td>8.9.1 Pinning Processes to CPU Cores</td>
<td>106</td>
</tr>
<tr>
<td>8.9.2 Controlling CPU and Memory Usage</td>
<td>106</td>
</tr>
<tr>
<td>8.9.3 Restricting Access to Devices</td>
<td>107</td>
</tr>
<tr>
<td>8.9.4 Throttling I/O Bandwidth</td>
<td>107</td>
</tr>
<tr>
<td>8.10 For More Information About cgroups</td>
<td>108</td>
</tr>
</tbody>
</table>
By manipulating a cgroup’s virtual file system, for example, by adding process IDs to tasks directories under /sys/fs/cgroup.

By editing the cgroup rules file /etc/cgrules.conf so that the rules engine or PAM move processes into cgroups automatically.

By using additional application software such as Linux Containers.

By using the APIs that are provided in libvirt.

Because you might ultimately want to deploy cgroups in a production environment, this chapter demonstrates how to configure cgroups by editing the /etc/cgconfig.conf and /etc/cgrules.conf files, and how to configure PAM to associate processes with cgroups.

Note
To use cgroups, you must install the libcgroup package on your system.

8.2 Subsystems

You control the access that cgroups have to system resources by specifying parameters to various kernel modules known as subsystems (or as resource controllers in some cgroups documentation).

The following table lists the subsystems that are provided with the cgroups package.

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>blkio</td>
<td>Controls and reports block I/O operations. See Section 8.2.1, “blkio Parameters”.</td>
</tr>
<tr>
<td>cpu</td>
<td>Controls access to CPU resources. See Section 8.2.2, “cpu Parameters”.</td>
</tr>
<tr>
<td>cpucore</td>
<td>Reports usage of CPU resources. See Section 8.2.3, “cpucore Parameters”.</td>
</tr>
<tr>
<td>cpuset</td>
<td>Controls access to CPU cores and memory nodes (for systems with NUMA architectures). See Section 8.2.4, “cpuset Parameters”.</td>
</tr>
<tr>
<td>devices</td>
<td>Controls access to system devices. See Section 8.2.5, “devices Parameters”.</td>
</tr>
<tr>
<td>freezer</td>
<td>Suspends or resumes cgroup tasks. See Section 8.2.6, “freezer Parameter”.</td>
</tr>
<tr>
<td>memory</td>
<td>Controls access to memory resources, and reports on memory usage. See Section 8.2.7, “memory Parameters”.</td>
</tr>
<tr>
<td>net_cls</td>
<td>Tags network packets for use by network traffic control. See Section 8.2.8, “net_cls Parameter”.</td>
</tr>
</tbody>
</table>

The following sections describe the parameters that you can set for each subsystem.

8.2.1 blkio Parameters

The following blkio parameters are defined:

**blkio.io_merged**

Reports the number of BIOS requests that have been merged into async, read, sync, or write I/O operations.
**blkio Parameters**

**blkio.io_queued**

Reports the number of requests for `async`, `read`, `sync`, or `write` I/O operations.

**blkio.io_service_bytes**

Reports the number of bytes transferred by `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers as recorded by the completely fair queueing (CFQ) scheduler, but not updated while it is operating on a request queue.

**blkio.io_serviced**

Reports the number of `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers as recorded by the CFQ scheduler, but not updated while it is operating on a request queue.

**blkio.io_service_time**

Reports the time in nanoseconds taken to complete `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers.

**blkio.io_wait_time**

Reports the total time in nanoseconds that a cgroup spent waiting for `async`, `read`, `sync`, or `write` I/O operations to complete to or from the devices specified by their major and minor numbers.

**blkio.reset_stats**

Resets the statistics for a cgroup if an integer is written to this parameter.

**blkio.sectors**

Reports the number of disk sectors written to or read from the devices specified by their major and minor numbers.

**blkio.throttle.io_service_bytes**

Reports the number of bytes transferred by `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers even while the CFQ scheduler is operating on a request queue.

**blkio.throttle.io_serviced**

Reports the number of `async`, `read`, `sync`, or `write` I/O operations to or from the devices specified by their major and minor numbers even while the CFQ scheduler is operating on a request queue.

**blkio.throttle.read_bps_device**

Specifies the maximum number of bytes per second that a cgroup may read from a device specified by its major and minor numbers. For example, the setting `8:1 4194304` specifies that a maximum of 4 MB per second may be read from `/dev/sda1`.

**blkio.throttle.read_iops_device**

Specifies the maximum number of read operations per second that a cgroup may perform on a device specified by its major and minor numbers. For example, the setting `8:1 100` specifies that a maximum of 100 read operations per second may be performed on `/dev/sda1`. 
### blkio.throttle.write_bps_device

Specifies the maximum number of bytes per second that a cgroup may write to a device specified by its major and minor numbers. For example, the setting `8:2 2097152` specifies a maximum of 2 MB per second may be written to `/dev/sda2`.

### blkio.throttle.write_iops_device

Specifies the maximum number of write operations per second that a cgroup may perform on a device specified by its major and minor numbers. For example, the setting `8:2 50` specifies that a maximum of 50 write operations per second may be performed on `/dev/sda2`.

### blkio.time

Reports the time in milliseconds that I/O access was available to a device specified by its major and minor numbers.

### blkio.weight

Specifies a bias value from 100 to 1000 that determines a cgroup's share of access to block I/O. The default value is 1000. The value is overridden by the setting for an individual device (see `blkio.weight_device`).

### blkio.weight_device

Specifies a bias value from 100 to 1000 that determines a cgroup's share of access to block I/O on a device specified by its major and minor numbers. For example, the setting `8:17 100` specifies a bias value of 100 for `/dev/sdb1`.

### 8.2.2 cpu Parameters

The following `cpu` parameters are defined:

#### cpu.rt_period_us

Specifies how often in microseconds that a cgroup's access to a CPU should be rescheduled. The default value is 1000000 (1 second).

#### cpu.rt_runtime.us

Specifies for how long in microseconds that a cgroup has access to a CPU between rescheduling operations. The default value is 950000 (0.95 seconds).

#### cpu.shares

Specifies the bias value that determines a cgroup's share of CPU time. The default value is 1024.

### 8.2.3 cpuacct Parameters

The following `cpuacct` parameters are defined:

#### cpuacct.stat

Reports the total CPU time in nanoseconds spent in user and system mode by all tasks in the cgroup.


**cpuset Parameters**

```plaintext
cpuacct.usage

Reports the total CPU time in nanoseconds for all tasks in the cgroup. Setting this parameter to 0 resets its value, and also resets the value of cpuacct.usage_percpu.

```

```plaintext
cpuacct.usage_percpu

Reports the total CPU time in nanoseconds on each CPU core for all tasks in the cgroup.

```

### 8.2.4 cpuset Parameters

The following cpuset parameters are defined:

```plaintext
cpuacct.cpu_exclusive

Specifies whether the CPUs specified by cpuset.cpus are exclusively allocated to this CPU set and cannot be shared with other CPU sets. The default value of 0 specifies that CPUs are not exclusively allocated. A value of 1 enables exclusive use of the CPUs by a CPU set.

```

```plaintext
cpuset.cpus

Specifies a list of CPU cores to which a cgroup has access. For example, the setting 0,1,5-8 allows access to cores 0, 1, 5, 6, 7, and 8. The default setting includes all the available CPU cores.

```

```

**Note**

If you associate the cpuset subsystem with a cgroup, you must specify a value for the cpuset.cpus parameter.

```plaintext
cpuset.mem_exclusive

Specifies whether the memory nodes specified by cpuset.mems are exclusively allocated to this CPU set and cannot be shared with other CPU sets. The default value of 0 specifies that memory nodes are not exclusively allocated. A value of 1 enables exclusive use of the memory nodes by a CPU set.

```

```plaintext
cpuset.mem_hardwall

Specifies whether the kernel allocates pages and buffers to the memory nodes specified by cpuset.mems exclusively to this CPU set and cannot be shared with other CPU sets. The default value of 0 specifies that memory nodes are not exclusively allocated. A value of 1 allows you to separate the memory nodes that are allocated to different cgroups.

```

```plaintext
cpuset.memory_migrate

Specifies whether memory pages are allowed to migrate between memory nodes if the value of cpuset.mems changes. The default value of 0 specifies that memory nodes are not allowed to migrate. A value of 1 allows pages to migrate between memory nodes, maintaining their relative position on the node list where possible.

```

```plaintext
cpuset.memory_pressure

If cpuset.memory_pressure_enabled has been set to 1, reports the memory pressure, which represents the number of attempts per second by processes to reclaim in-use memory. The reported value scales the actual number of attempts up by a factor of 1000.

```


**cpuset.memory_pressure_enabled**

Specifies whether the memory pressure statistic should be gathered. The default value of 0 disables the counter. A value of 1 enables the counter.

**cpuset.memory_spread_page**

Specifies whether file system buffers are distributed between the allocated memory nodes. The default value of 0 results in the buffers being placed on the same memory node as the process that owns them. A value of 1 allows the buffers to be distributed across the memory nodes of the CPU set.

**cpuset.memory_spread_slab**

Specifies whether I/O slab caches are distributed between the allocated memory nodes. The default value of 0 results in the caches being placed on the same memory node as the process that owns them. A value of 1 allows the caches to be distributed across the memory nodes of the CPU set.

**cpuset.mems**

Specifies the memory nodes to which a cgroup has access. For example, the setting `0-2,4` allows access to memory nodes 0, 1, 2, and 4. The default setting includes all available memory nodes. The parameter has a value of 0 on systems that do not have a NUMA architecture.

*Note*

If you associate the `cpuset` subsystem with a cgroup, you must specify a value for the `cpuset.mems` parameter.

**cpuset.sched_load_balance**

Specifies whether the kernel should attempt to balance CPU load by moving processes between the CPU cores allocated to a CPU set. The default value of 1 turns on load balancing. A value of 0 disables load balancing. Disabling load balancing for a cgroup has no effect if load balancing is enabled in the parent cgroup.

**cpuset.sched_relax_domain_level**

If `cpuset.sched_load_balance` is set to 1, specifies one of the following load-balancing schemes.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Use the system's default load balancing scheme. This is the default behavior.</td>
</tr>
<tr>
<td>0</td>
<td>Perform periodic load balancing. Higher numeric values enable immediate load balancing.</td>
</tr>
<tr>
<td>1</td>
<td>Perform load balancing for threads running on the same core.</td>
</tr>
<tr>
<td>2</td>
<td>Perform load balancing for cores of the same CPU.</td>
</tr>
<tr>
<td>3</td>
<td>Perform load balancing for all CPU cores on the same system.</td>
</tr>
<tr>
<td>4</td>
<td>Perform load balancing for a subset of CPU cores on a system with a NUMA architecture.</td>
</tr>
<tr>
<td>5</td>
<td>Perform load balancing for all CPU cores on a system with a NUMA architecture.</td>
</tr>
</tbody>
</table>

**8.2.5 devices Parameters**

The following `devices` parameters are defined:
**devices.allow**

Specifies a device that a cgroup is allowed to access by its type (a for any, b for block, or c for character), its major and minor numbers, and its access modes (m for create permission, r for read access, and w for write access).

For example, `b 8:17 rw` would allow read and write access to the block device `/dev/sdb1`.

You can use the wildcard `*` to represent any major or minor number. For example, `b 8:* rw` would allow read and write access to any `/dev/sd*` block device.

Each device that you specify is added to the list of allowed devices.

**devices.deny**

Specifies a device that a cgroup is not allowed to access.

Removes each device that you specify from the list of allowed devices.

**devices.list**

Reports those devices for which access control is set. If no devices are controlled, all devices are reported as being available in all access modes: `a *:* rwm`.

### 8.2.6 freezer Parameter

The following `freezer` parameter is defined:

**freezer.state**

Specifies one of the following operations.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROZEN</td>
<td>Suspends all the tasks in a cgroup. You cannot move a process into a frozen cgroup.</td>
</tr>
<tr>
<td>THAWED</td>
<td>Resumes all the tasks in a cgroup.</td>
</tr>
</tbody>
</table>

**Note**

You cannot set the `FREEZING` state. If displayed, this state indicates that the system is currently suspending the tasks in the cgroup.

The `freezer.state` parameter is not available in the root cgroup.

### 8.2.7 memory Parameters

The following `memory` parameters are defined:

**memory.failcnt**

Specifies the number of times that the amount of memory used by a cgroup has risen to `memory.limit_in_bytes`.

**memory.force_empty**

If a cgroup has no tasks, setting the value to 0 removes all pages from memory that were used by tasks in the cgroup. Setting the parameter in this way avoids a parent cgroup from being assigned the defunct page caches when you remove its child cgroup.
**memory.limit_in_bytes**

Specifies the maximum usage permitted for user memory including the file cache. The default units are bytes, but you can also specify a `k` or `K`, `m` or `M`, and `g` or `G` suffix for kilobytes, megabytes, and gigabytes respectively. A value of -1 removes the limit.

To avoid an out-of-memory error, set the value of `memory.limit_in_bytes` lower than `memory.memsw.limit_in_bytes`, and set `memory.memsw.limit_in_bytes` lower than the amount of available swap space.

**memory.max_usage_in_bytes**

Reports the maximum amount of user memory in bytes used by tasks in the cgroup.

**memory.memsw.failcnt**

Specifies the number of times that the amount of memory and swap space used by a cgroup has risen to `memory.memsw.limit_in_bytes`.

**memory.memsw.limit_in_bytes**

Specifies the maximum usage permitted for user memory plus swap space. The default units are bytes, but you can also specify a `k` or `K`, `m` or `M`, and `g` or `G` suffix for kilobytes, megabytes, and gigabytes respectively. A value of -1 removes the limit.

**memory.memsw.max_usage_in_bytes**

Reports the maximum amount of user memory and swap space in bytes used by tasks in the cgroup.

**memory.memsw.usage_in_bytes**

Reports the total size in bytes of the memory and swap space used by tasks in the cgroup.

**memory.move_charge_at_immigrate**

Specifies whether a task's charges are moved when you migrate the task between cgroups. You can specify the following values.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disable moving task charges.</td>
</tr>
<tr>
<td>1</td>
<td>Moves charges for an in-use or swapped-out anonymous page exclusively owned by the task.</td>
</tr>
<tr>
<td>2</td>
<td>Moves charges for file pages that are memory mapped by the task.</td>
</tr>
<tr>
<td>3</td>
<td>Equivalent to specifying both 1 and 2.</td>
</tr>
</tbody>
</table>

**memory.numa_stat**

Reports the NUMA memory usage in bytes for each memory node (N0, N1,...) together with the following statistics.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>anon</td>
<td>The size in bytes of anonymous and swap cache.</td>
</tr>
<tr>
<td>file</td>
<td>The size in bytes of file-backed memory.</td>
</tr>
<tr>
<td>total</td>
<td>The sum of the <code>anon</code>, <code>file</code> and <code>unevictable</code> values.</td>
</tr>
</tbody>
</table>
### memory Parameters

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unevictable</td>
<td>The size in bytes of unreclaimable memory.</td>
</tr>
</tbody>
</table>

#### memory.oom_control

Displays the values of the out-of-memory (OOM) notification control feature.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oom_kill_disable</td>
<td>Whether the OOM killer is enabled (0) or disabled (1).</td>
</tr>
<tr>
<td>under_oom</td>
<td>Whether the cgroup is under OOM control (1) allowing tasks to be stopped, or not under OOM control (0).</td>
</tr>
</tbody>
</table>

#### memory.soft_limit_in_bytes

Specifies a soft, upper limit for user memory including the file cache. The default units are bytes, but you can also specify a k or K, m or M, and g or G suffix for kilobytes, megabytes, and gigabytes respectively. A value of -1 removes the limit.

The soft limit should be lower than the hard-limit value of `memory.limit_in_bytes` as the hard limit always takes precedence.

#### memory.stat

Reports the following memory statistics.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active_anon</td>
<td>The size in bytes of anonymous and swap cache on active least-recently-used (LRU) list (includes tmpfs).</td>
</tr>
<tr>
<td>active_file</td>
<td>The size in bytes of file-backed memory on active LRU list.</td>
</tr>
<tr>
<td>cache</td>
<td>The size in bytes of page cache (includes tmpfs).</td>
</tr>
<tr>
<td>hierarchical_memory_limit</td>
<td>The size in bytes of the limit of memory for the cgroup hierarchy.</td>
</tr>
<tr>
<td>hierarchical_memsw_limit</td>
<td>The size in bytes of the limit of memory plus swap for the cgroup hierarchy.</td>
</tr>
<tr>
<td>inactive_anon</td>
<td>The size in bytes of anonymous and swap cache on inactive LRU list (includes tmpfs).</td>
</tr>
<tr>
<td>inactive_file</td>
<td>The size in bytes of file-backed memory on inactive LRU list.</td>
</tr>
<tr>
<td>mapped_file</td>
<td>The size in bytes of memory-mapped files (includes tmpfs).</td>
</tr>
<tr>
<td>pgfault</td>
<td>The number of page faults, where the kernel has to allocate and initialize physical memory for use in the virtual address space of a process.</td>
</tr>
<tr>
<td>pgmajfault</td>
<td>The number of major page faults, where the kernel has to actively free physical memory before allocation and initialization.</td>
</tr>
<tr>
<td>pgpgin</td>
<td>The number of paged-in pages of memory.</td>
</tr>
<tr>
<td>pgpgout</td>
<td>The number of paged-out pages of memory.</td>
</tr>
<tr>
<td>rss</td>
<td>The size in bytes of anonymous and swap cache (does not include tmpfs). The actual resident set size is given by the sum of rss and mapped_file.</td>
</tr>
<tr>
<td>swap</td>
<td>The size in bytes of used swap space.</td>
</tr>
</tbody>
</table>
### net_cls Parameter

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>total_*</td>
<td>The value of the appended statistic for the cgroup and all of its children.</td>
</tr>
<tr>
<td>unevictable</td>
<td>The size in bytes of memory that is not reclaimable.</td>
</tr>
</tbody>
</table>

**memory.swappiness**

Specifies a bias value for the kernel to swap out memory pages used by processes in the cgroup rather than reclaim pages from the page cache. A value smaller than the default value of 60 reduces the kernel's preference for swapping out. A value greater than 60 increases the preference for swapping out. A value greater than 100 allows the system to swap out pages that fall within the address space of the cgroup's tasks.

**memory.usage_in_bytes**

Reports the total size in bytes of the memory used by all the tasks in the cgroup.

**memory.use_hierarchy**

Specifies whether the kernel should attempt to reclaim memory from a cgroup's hierarchy. The default value of 0 prevents memory from being reclaimed from other tasks in the hierarchy. A value of 1 allows memory to be reclaimed from other tasks in the hierarchy.

#### 8.2.8 net_cls Parameter

The following `net_cls` parameter is defined:

**net_cls.classid**

Specifies the hexadecimal class identifier that the system uses to tag network packets for use with the Linux traffic controller.

### 8.3 Enabling the cgconfig Service

To enable the cgroup services on a system:

1. Install the `libcggroup` package.

    ```bash
    # yum install libcggroup
    ```

2. Start the `cgconfig` service and configure it to start when the system is booted.

    ```bash
    # service cgconfig start
    # chkconfig cgconfig on
    ```

### 8.4 Enabling PAM to Work with cgroup Rules

To configure PAM to use the rules that you configure in the `/etc/cgrules.conf` file:

1. Install the `libcggroup-pam` package.

    ```bash
    # yum install libcggroup-pam
    ```

    The `pam_cgroup.so` module is installed in `/lib64/security` on 64-bit systems, and in `/lib/security` on 32-bit systems.
2. Edit the /etc/pam.d/su configuration file, and add the following line for the pam_cgroup.so module:

```
session optional pam_cgroup.so
```

Note
For a service that has a configuration file in /etc/sysconfig, you can add the following line to the start section of the file to start the service in a specified cgroup:

```
CGROUP_DAEMON="*:cgroup"
```

8.5 Restarting the cgconfig Service

If you make any changes to the cgroups configuration file, /etc/cgconfig.conf, restart the cgconfig service to make it reread the file.

```
# service cgconfig restart
```

8.6 About the cgroups Configuration File

The cgroups configuration file, /etc/cgconfig.conf, contains a mount definition and one or more group definitions.

**mount Definitions**

A mount definition specifies the virtual file systems that you use to mount resource subsystems before you attach them to cgroups. The configuration file can contain only one mount definition.

The mount entry takes the following form:

```
mount {
    subsystem1 = /cgroup/resource_path1;
    [subsystem2 = /cgroup/resource_path2;]
    ...
    ...
}
```

For example, the following mount definition combines the cpu, cpuset, and memory subsystems under the /cgroup/cpumem subsystem hierarchy, and also creates entries for the blkio and devices subsystems under /cgroup/iolimit and /cgroup/devlist. You cannot include a subsystem in more than one subsystem hierarchy.

```
mount {
    cpu = /cgroup/cpumem;
    cpuset = /cgroup/cpumem;
    memory = /cgroup/cpumem;
    blkio = /cgroup/iolimit;
    devices = /cgroup/devlist;
}
```

**group Definitions**

A group definition specifies a cgroup, its access permissions, the resource subsystems that it uses, and the parameter values for those subsystems. The configuration file can contain more than one group definition.
A `group` entry takes the following form:

```
group cgroup_name {
  perm {
    task {
      uid = task_user;
      gid = task_group;
    }
    admin {
      uid = admin_user;
      gid = admin_group;
    }
  }
  subsystem {
    [subsystem.parameter1 = value1];
    [subsystem.parameter2 = value2];
    ...
  }
}
```

The `cgroup_name` argument defines the name of the cgroup. The `task` section of the optional `perm` (permissions) section defines the user and group combination that can add tasks to the cgroup. The `admin` section defines the user and group combination that can modify subsystem parameters and create subgroups. Whatever settings exist under `perm`, the `root` user always has permission to make any `admin` or `task` change.

One or more subsystem sections define the parameter settings for the cgroup. You can associate only one virtual subsystem hierarchy from `/cgroup` with a cgroup. If a several subsystems are grouped in the same hierarchy, you must include definitions for all the subsystems. For example, if the `/cgroup/cpumem` hierarchy includes the `cpu`, `cpuset`, and `memory` subsystems, you must include definitions for all of these subsystems.

For example, the following `group` definition defines the cgroup `dbgrp` for database processes, allows the `oracle` user to add tasks, and sets various parameters for CPU and memory usage:

```
group dbgrp {
  perm {
    task {
      uid = oracle;
      gid = dba;
    }
    admin {
      uid = root;
      gid = root;
    }
  }
  cpu {
    # Reallocate CPU resources once per second
    cpu.rt_period_us="1000000";
    # Allocate 50% of runtime to tasks in the cgroup
    cpu.rt_runtime_us="5000000";
  }
  cpuset {
    cpuset.mems="0";
    # Allocate CPU cores 4 through 7 to tasks in the cgroup
    cpuset.cpus="4-7";
  }
  memory {
    # Allocate at most 4 GB of memory to tasks
```
You can include comments in the file by preceding them with a `#` character, which must be at the start of a line.

### 8.8 Displaying and Setting Subsystem Parameters

To display the value of a subsystem parameter, use the `cgget` command. The following example shows how to display the memory statistics for the cgroup `hipri`.

```
# cgget -r memory.stat hipri
    rss 168132608
```
You can use the `cgset` command to change the value of subsystem parameters for a cgroup. The next example removes input throttling from the device `/dev/sda1` for the cgroup `iocap1` by setting the value of `blkio.throttle.read_bps_device` to 0.

```
# cgset -r blkio.throttle.read_bps_device="8:1 0" iocap1
```

Any change that you make to a parameter is effective only while the `cgconfig` service continues to run. The `cgset` command does not write the new value to the configuration file, `/etc/cgconfig.conf`. You can use the `cgsnapshot` command to display the current cgroup configuration in a form that you can use as the basis for a new `/etc/cgconfig.conf` file.

```
# cgsnapshot -s > current_cgconfig.conf
```

For more information, see the `cgget(1)`, `cgset(1)`, and `cgsnapshot(1)` manual pages.

### 8.9 Use Cases for cgroups

The following sections describe sample `/etc/cgconfig.conf` entries for cgroups that can control the access that processes have to system resources.

#### 8.9.1 Pinning Processes to CPU Cores

Define two cgroups that can be used to assign tasks to run on different sets of CPU cores.

```
mount {
    cpuset = /cgroup/coregrp;
}

group locores {
    cpuset {
        cpuset.mems="0";
        # Run tasks on cores 0 through 3
        cpuset.cpus="0-3";
    }
}

group hicores {
    cpuset {
        cpuset.mems="0";
        # Run tasks on cores 4 through 7
        cpuset.cpus="4-7";
    }
}
```

#### 8.9.2 Controlling CPU and Memory Usage

Define two cgroups with different allocations of available CPU time and memory resources.

```
mount {
    cpu = /cgroup/cpumem;
    cpuset = /cgroup/cpumem;
    memory = /cgroup/cpumem;
}

# High priority group
group hipri {
    cpu {
```
8.9.3 Restricting Access to Devices

Define a cgroup that denies access to the disk devices /dev/sd[bcd].

```
mount {
    devices = /cgroup/devlist;
}

group blkdev {
    devices {
        Deny access to /dev/sdb
        devices.deny="b 8:16 mrw";
        Deny access to /dev/sdc
        devices.deny="b 8:32 mrw";
        Deny access to /dev/sdd
        devices.deny="b 8:48 mrw";
    }
}
```

8.9.4 Throttling I/O Bandwidth

Define a cgroup that limits the I/O bandwidth to 50MB/s when reading from /dev/sda1.

```
mount {
    
```
Define a cgroup that limits the number of read transactions to 100 per second when reading from /dev/sdd.

```bash
mount {
    blkio = /cgroup/iolimit;
}
group iocap2 {
    blkio {
        # Limit read tps from /dev/sdd to 100 per second
        blkio.throttle.read_iops_device="8:48 100";
    }
}
```

Define two cgroups with different shares of I/O access to /dev/sdb.

```bash
mount {
    blkio = /cgroup/iolimit;
}
# Low access share group
group iolo {
    blkio {
        # Set the share of I/O access by /dev/sdb to 25%
        blkio.weight_device="8:16 250";
    }
}
# High access share group
group iohi {
    blkio {
        # Set the share of I/O access by /dev/sdb to 75%
        blkio.weight_device="8:16 750";
    }
}
```

### 8.10 For More Information About cgroups

Chapter 9 Linux Containers

Table of Contents

9.1 About Linux Containers ...................................................................................................................................................... 109
  9.1.1 Supported Oracle Linux Container Versions .............................................................................................................. 111
9.2 Configuring Operating System Containers .......................................................................................................................... 111
  9.2.1 Installing and Configuring the Software .......................................................................................................................... 111
  9.2.2 Setting up the File System for the Containers .................................................................................................................. 112
  9.2.3 Creating and Starting a Container ........................................................................................................................................ 112
  9.2.4 About the lxc-oracle Template Script ............................................................................................................................... 114
  9.2.5 About Veth and Macvlan ...................................................................................................................................................... 116
  9.2.6 Modifying a Container to Use Macvlan ................................................................................................................................ 117
9.3 Logging in to Containers ....................................................................................................................................................... 118
9.4 Creating Additional Containers .............................................................................................................................................. 118
9.5 Monitoring and Shutting Down Containers ............................................................................................................................ 119
9.6 Starting a Command Inside a Running Container .................................................................................................................. 121
9.7 Controlling Container Resources ........................................................................................................................................... 121
9.8 Configuring ulimit Settings for an Oracle Linux Container ...................................................................................................... 122
9.9 Configuring Kernel Parameter Settings for Oracle Linux Containers ......................................................................................... 122
9.10 Deleting Containers ................................................................................................................................................................. 123
9.11 Running Application Containers ........................................................................................................................................... 123
9.12 For More Information About Linux Containers ..................................................................................................................... 125

This chapter describes how to use Linux Containers (LXC) to isolate applications and entire operating system images from the other processes that are running on a host system. The version of LXC described here is 1.0.7 or later running under UEK R3 QU6 or later, which provides some significant enhancements over previous versions.

For information about how to use the Docker Engine to create application containers, see Chapter 10, Docker.

9.1 About Linux Containers

Note

Prior to UEK R3, LXC was a Technology Preview feature that was made available for testing and evaluation purposes, but was not recommended for production systems. LXC is a supported feature with UEK R3.

The Linux Containers (LXC) feature is a lightweight virtualization mechanism that does not require you to set up a virtual machine on an emulation of physical hardware. The Linux Containers feature takes the cgroups resource management facilities as its basis and adds POSIX file capabilities to implement process and network isolation. You can run a single application within a container (an application container) whose name space is isolated from the other processes on the system in a similar manner to a chroot jail.

However, the main use of Linux Containers is to allow you to run a complete copy of the Linux operating system in a container (a system container) without the overhead of running a level-2 hypervisor such as VirtualBox. In fact, the container is sharing the kernel with the host system, so its processes and file system are completely visible from the host. When you are logged into the container, you only see its file system and process space. Because the kernel is shared, you are limited to the modules and drivers that it has loaded.

Typical use cases for Linux Containers are:
• Running Oracle Linux 5 and Oracle Linux 6 containers in parallel. Both versions of the operating system support the Unbreakable Enterprise Kernel Release 2. You can even run an Oracle Linux 5 container on an Oracle Linux 6 system with the UEK R3 kernel, even though UEK R3 is not supported for Oracle Linux 5. You can also run an i386 container on an x86_64 kernel. However, you cannot run an x86_64 container on an i386 kernel. For more information, see Section 9.1.1, “Supported Oracle Linux Container Versions”.

• Running applications that are supported only by Oracle Linux 5 in an Oracle Linux 5 container on an Oracle Linux 6 host. However, incompatibilities might exist in the modules and drivers that are available.

• Running many copies of application configurations on the same system. An example configuration would be a LAMP stack, which combines Linux, Apache server, MySQL, and Perl, PHP, or Python scripts to provide specialised web services.

• Creating sandbox environments for development and testing.

• Providing user environments whose resources can be tightly controlled, but which do not require the hardware resources of full virtualization solutions.

• Creating containers where each container appears to have its own IP address. For example you can use the lxc-sshd template script to create isolated environments for untrusted users. Each container runs an sshd daemon to handle logins. By bridging a container's Virtual Ethernet interface to the host's network interface, each container can appear to have its own IP address on a LAN.

When you use the lxc-start command to start a system container, by default the copy of /sbin/init in the container is started to spawn other processes in the container's process space. Any system calls or device access are handled by the kernel running on the host. If you need to run different kernel versions or different operating systems from the host, use a true virtualization solution such as Oracle VM or Oracle VM VirtualBox instead of Linux Containers.

There are a number of configuration steps that you need to perform on the file system image for a container so that it can run correctly:

• Disable any init scripts that load modules to access hardware directly.

• Disable udev and instead create static device nodes in /dev for any hardware that needs to be accessible from within the container.

• Configure the network interface so that it is bridged to the network interface of the host system.

LXC provides a number of template scripts in /usr/share/lxc/templates that perform much of the required configuration of system containers for you. However, it is likely that you will need to modify the script to allow the container to work correctly as the scripts cannot anticipate the idiosyncrasies of your system's configuration. You use the lxc-create command to create a system container by invoking a template script. For example, the lxc-busybox template script creates a lightweight BusyBox system container.

The example system container in this chapter uses the template script for Oracle Linux (lxc-oracle). The container is created on a btrfs file system (/container) to take advantage of its snapshot feature. A btrfs file system allows you to create a subvolume that contains the root file system (rootfs) of a container, and to quickly create new containers by cloning this subvolume.

You can use control groups to limit the system resources that are available to applications such as web servers or databases that are running in the container.

Application containers are not created by using template scripts. Instead, an application container mounts all or part of the host's root file system to provide access to the binaries and libraries that the application requires. You use the lxc-execute command to invoke lxc-init (a cut-down version of /sbin/
9.1.1 Supported Oracle Linux Container Versions

The following table shows the tested and supported Oracle Linux container versions for Oracle Linux 6 hosts:

<table>
<thead>
<tr>
<th>Host</th>
<th>Container Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Linux 6.5 (kernel-uek-3.8.13-16.2.1 or later)</td>
<td>Oracle Linux 5.9 or later</td>
</tr>
<tr>
<td></td>
<td>Oracle Linux 6.5 or later</td>
</tr>
<tr>
<td>Oracle Linux 6.6 (kernel-uek-3.8.13-44.1.1 or later)</td>
<td>Oracle Linux 5.9 or later</td>
</tr>
<tr>
<td></td>
<td>Oracle Linux 6.5 or later</td>
</tr>
<tr>
<td></td>
<td>Oracle Linux 7.0 or later</td>
</tr>
</tbody>
</table>

9.2 Configuring Operating System Containers

The procedures in the following sections describe how to set up Linux Containers that contain a copy of the root file system installed from packages at Oracle Public Yum.

- Section 9.2.1, “Installing and Configuring the Software”
- Section 9.2.2, “Setting up the File System for the Containers”
- Section 9.2.3, “Creating and Starting a Container”

**Note**
Throughout the following sections in this chapter, the prompts [root@host ~]# and [root@ol6ctr1 ~]# distinguish between commands run by root on the host and in the container.

The software functionality described requires that you boot the system with at least the Unbreakable Enterprise Kernel Release 2 (2.6.39).

9.2.1 Installing and Configuring the Software

To install and configure the software that is required to run Linux Containers:

1. Use `yum` to install the `btrfs-progs` package.

   ```bash
   [root@host ~]# yum install btrfs-progs
   ```

2. Install the `lxc` and `wget` packages.

   ```bash
   [root@host ~]# yum install lxc wget
   ```

   This command installs all of the required packages, such as `libvirt`, `libcg`, and `lxc-libs`. The LXC template scripts are installed in `/usr/share/lxc/templates`. LXC uses `wget` to download packages from Oracle Public Yum.

3. Start the Control Groups (cgroups) service, `cgconfig`, and configure the service to start at boot time.

   ```bash
   [root@host ~]# service cgconfig start
   ```
LXC uses the cgroups service to control the system resources that are available to containers.

4. Start the virtualization management service, `libvirtd`, and configure the service to start at boot time.

```
[root@host ~]# chkconfig libvirtd on
```

LXC uses the virtualization management service to support network bridging for containers.

5. If you are going to compile applications that require the LXC header files and libraries, install the `lxc-devel` package.

```
[root@host ~]# yum install lxc-devel
```

### 9.2.2 Setting up the File System for the Containers

**Note**

The LXC template scripts assume that containers are created in `/container`. You must edit the script if your system's configuration differs from this assumption.

To set up the `/container` file system:

1. Create a btrfs file system on a suitably sized device such as `/dev/sdb`, and create the `/container` mount point.

```
[root@host ~]# mkfs.btrfs /dev/sdb
```

```
[root@host ~]# mkdir /container
```

2. Mount the `/container` file system.

```
[root@host ~]# mount /dev/sdb /container
```

3. Add an entry for `/container` to the `/etc/fstab` file.

```
/dev/sdb      /container    btrfs    defaults   0 0
```

For more information, see Chapter 5, *The Btrfs File System*.

### 9.2.3 Creating and Starting a Container

**Note**

The procedure in this section uses the LXC template script for Oracle Linux (`lxc-oracle`), which is located in `/usr/share/lxc/templates`.

An Oracle Linux container requires a minimum of 400 MB of disk space.

To create and start a container:

1. Create an Oracle Linux 6 container named `ol6ctr1` using the `lxc-oracle` template script.

```
[root@host ~]# lxc-create -n ol6ctr1 -B btrfs -t oracle -- --release=6.latest
```

```
lxc-create: No config file specified, using the default config /etc/lxc/default.conf
Host is OracleServer 6.4
Creating configuration file /container/ol6ctr1/config
Downloading release 6.0.0.0.0 for x86_64

...''
Creating and Starting a Container

For LXC version 1.0 and later, you must specify the `-B btrfs` option if you want to use the snapshot features of btrfs. For more information, see the `lxc-create(1)` manual page.

The `lxc-create` command runs the template script `lxc-oracle` to create the container in `/container/ol6ctr1` with the btrfs subvolume `/container/ol6ctr1/rootfs` as its root file system. The command then uses `yum` to install the latest available update of Oracle Linux 6 from Oracle Public Yum. It also writes the container's configuration settings to the file `/container/ol6ctr1/config` and its `fstab` file to `/container/ol6ctr1/fstab`. The default log file for the container is `/container/ol6ctr1/ol6ctr1.log`.

You can specify the following template options after the `--` option to `lxc-create`:

- `-a` | `--arch=i386|x86_64` Specifies the architecture. The default value is the architecture of the host.

- `--baseurl=PKG_REPO` Specifies the file URI of a package repository. You must also use the `--arch` and `--release` options to specify the architecture and the release, for example:

```
# mount -o loop OracleLinux-R7-GA-Everything-x86_64-dvd.iso /mnt
# lxc-create -n ol70beta -B btrfs -t oracle -- -R 7.0 -a x86_64 \
  --baseurl=file://mnt/Server
```

- `-P` | `--patch=PATH` Patches the rootfs at the specified path.

- `--privileged=[rt]` Allows you to adjust the values of certain kernel parameters under the `/proc` hierarchy.

The container uses a privilege configuration file, which mounts `/proc` read-only with some exceptions. See Section 9.9, "Configuring Kernel Parameter Settings for Oracle Linux Containers".

This option also enables the `CAP_SYS_NICE` capability, which allows you to set negative `nice` values (that is, more favored for scheduling) for processes from within the container.

If you specify the `=rt` (real-time) modifier, you can configure the `lxc.cgroup.cpu.rt_runtime_us` setting in the container's configuration file or when you start the container. This setting specifies the maximum continuous period in microseconds for which the container has access to CPU resources from the base period set by the system-wide value of `cpu.rt_period_us`. Otherwise, a container uses the system-wide value of `cpu.rt_runtime_us`, which might cause it to consume too many CPU resources. In addition, this modifier ensures that rebooting a container terminates all of its processes and boots it to a clean state.
About the lxc-oracle Template Script

- `R` | `--release=.major.minor` Specifies the major release number and minor update number of the Oracle release to install. The value of `major` can be set to 4, 5, 6, or 7. If you specify `latest` for `minor`, the latest available release packages for the major release are installed. If the host is running Oracle Linux, the default release is the same as the release installed on the host. Otherwise, the default release is the latest update of Oracle Linux 6.

- `-r` | `--rpms=rpm_name` Installs the specified RPM in the container.

- `-t` | `--templatefs=rootfs` Specifies the path to the root file system of an existing system, container, or Oracle VM template that you want to copy. Do not specify this option with any other template option. See Section 9.4, “Creating Additional Containers”.

- `-u` | `--url=repo_URL` Specifies a yum repository other than Oracle Public Yum. For example, you might want to perform the installation from a local yum server. The repository file is configured in `/etc/yum.repos.d` in the container's root file system. The default URL is `http://public-yum.oracle.com`.

2. If you want to create additional copies of the container in its initial state, create a snapshot of the container's root file system, for example:

```
# btrfs subvolume snapshot /container/ol6ctr1/rootfs /container/ol6ctr1/rootfs_snap
```

See Chapter 5, *The Btrfs File System* and Section 9.4, “Creating Additional Containers”.

3. Start the container `ol6ctr1` as a daemon that writes its diagnostic output to a log file other than the default log file.

```
[root@host ~]# lxc-start -n ol6ctr1 -d -o /container/ol6ctrl_debug.log -l DEBUG
```

Note

If you omit the `-d` option, the container's console opens in the current shell.

The following logging levels are available: `FATAL`, `CRIT`, `WARN`, `ERROR`, `NOTICE`, `INFO`, and `DEBUG`. You can set a logging level for all `lxc-*` commands.

If you run the `ps -ef --forest` command on the host system and the process tree below the `lxc-start` process shows that the `/usr/sbin/sshd` and `/sbin/mingetty` processes have started in the container, you can log in to the container from the host. See Section 9.3, “Logging in to Containers”.

9.2.4 About the lxc-oracle Template Script

Note

If you amend a template script, you alter the configuration files of all containers that you subsequently create from that script. If you amend the `config` file for a container, you alter the configuration of that container and all containers that you subsequently clone from it.

The `lxc-oracle` template script defines system settings and resources that are assigned to a running container, including:
About the lxc-oracle Template Script

- the default passwords for the oracle and root users, which are set to oracle and root respectively
- the host name (lxc.utsname), which is set to the name of the container
- the number of available terminals (lxc.tty), which is set to 4
- the location of the container's root file system on the host (lxc.rootfs)
- the location of the fstab mount configuration file (lxc.mount)
- all system capabilities that are not available to the container (lxc.cap.drop)
- the local network interface configuration (lxc.network)
- all whitelisted cgroup devices (lxc.cgroup.devices.allow)

The template script sets the virtual network type (lxc.network.type) and bridge (lxc.network.link) to veth and virbr0. If you want to use a macvlan bridge or Virtual Ethernet Port Aggregator that allows external systems to access your container via the network, you must modify the container's configuration file. See Section 9.2.5, “About Veth and Macvlan” and Section 9.2.6, “Modifying a Container to Use Macvlan”.

To enhance security, you can uncomment lxc.cap.drop capabilities to prevent root in the container from performing certain actions. For example, dropping the sys_admin capability prevents root from remounting the container's fstab entries as writable. However, dropping sys_admin also prevents the container from mounting any file system and disables the hostname command. By default, the template script drops the following capabilities: mac_admin, mac_override, setfcap, setpcap, sys_module, sys_nice, sys_pacct, sys_rawio, and sys_time.

For more information, see Chapter 8, Control Groups and the capabilities(7) and lxc.conf(5) manual pages.

When you create a container, the template script writes the container's configuration settings and mount configuration to /container/name/config and /container/name/fstab, and sets up the container's root file system under /container/name/rootfs.

Unless you specify to clone an existing root file system, the template script installs the following packages under rootfs (by default, from Oracle Public Yum at http://public-yum.oracle.com):

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chkconfig</td>
<td>chkconfig utility for maintaining the /etc/rc*.d hierarchy.</td>
</tr>
<tr>
<td>dhclient</td>
<td>DHCP client daemon (dhclient) and dhclient-script.</td>
</tr>
<tr>
<td>initscripts</td>
<td>/etc/inittab file and /etc/init.d scripts.</td>
</tr>
<tr>
<td>openssh-server</td>
<td>Open source SSH server daemon, /usr/sbin/sshd.</td>
</tr>
<tr>
<td>oraclelinux-release</td>
<td>Oracle Linux 6 release and information files.</td>
</tr>
<tr>
<td>passwd</td>
<td>passwd utility for setting or changing passwords using PAM.</td>
</tr>
<tr>
<td>policycoreutils</td>
<td>SELinux policy core utilities.</td>
</tr>
<tr>
<td>rootfiles</td>
<td>Basic files required by the root user.</td>
</tr>
<tr>
<td>rsyslog</td>
<td>Enhanced system logging and kernel message trapping daemons.</td>
</tr>
<tr>
<td>vim-minimal</td>
<td>Minimal version of the VIM editor.</td>
</tr>
<tr>
<td>yum</td>
<td>yum utility for installing, updating and managing RPM packages.</td>
</tr>
</tbody>
</table>
The template script edits the system configuration files under `rootfs` to set up networking in the container and to disable unnecessary services including volume management (LVM), device management (`udev`), the hardware clock, `readahead`, and the Plymouth boot system.

### 9.2.5 About Veth and Macvlan

By default, the `lxc-oracle` template script sets up networking by setting up a veth bridge. In this mode, a container obtains its IP address from the `dnsmasq` server that `libvirtd` runs on the private virtual bridge network (`virbr0`) between the container and the host. The host allows a container to connect to the rest of the network by using NAT rules in `iptables`, but these rules do not allow incoming connections to the container. Both the host and other containers on the veth bridge have network access to the container via the bridge.

Figure 9.1 illustrates a host system with two containers that are connected via the veth bridge `virbr0`.

**Figure 9.1 Network Configuration of Containers Using a Veth Bridge**

If you want to allow network connections from outside the host to be able to connect to the container, the container needs to have an IP address on the same network as the host. One way to achieve this configuration is to use a macvlan bridge to create an independent logical network for the container. This network is effectively an extension of the local network that is connected the host's network interface. External systems can access the container as though it were an independent system on the network, and the container has network access to other containers that are configured on the bridge and to external systems. The container can also obtain its IP address from an external DHCP server on your local network. However, unlike a veth bridge, the host system does not have network access to the container.

Figure 9.2 illustrates a host system with two containers that are connected via a macvlan bridge.

**Figure 9.2 Network Configuration of Containers Using a Macvlan Bridge**
If you do not want containers to be able to see each other on the network, you can configure the Virtual Ethernet Port Aggregator (VEPA) mode of macvlan. Figure 9.3 illustrates a host system with two containers that are separately connected to a network by a macvlan VEPA. In effect, each container is connected directly to the network, but neither container can access the other container nor the host via the network.

Figure 9.3 Network Configuration of Containers Using a Macvlan VEPA

For information about configuring macvlan, see Section 9.2.6, “Modifying a Container to Use Macvlan” and the lxc.conf(5) manual page.

9.2.6 Modifying a Container to Use Macvlan

To modify a container so that it uses the bridge or VEPA mode of macvlan, edit /container/name/config and replace the following lines:

```
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = virbr0
```

with these lines for bridge mode:

```
lxc.network.type = macvlan
lxc.network.macvlan.mode = bridge
lxc.network.flags = up
lxc.network.link = eth0
```

or these lines for VEPA mode:

```
lxc.network.type = macvlan
lxc.network.macvlan.mode = vepa
lxc.network.flags = up
lxc.network.link = eth0
```

In these sample configurations, the setting for lxc.network.link assumes that you want the container’s network interface to be visible on the network that is accessible via the host’s eth0 interface.

9.2.6.1 Modifying a Container to Use a Static IP Address

By default, a container connected by macvlan relies on the DHCP server on your local network to obtain its IP address. If you want the container to act as a server, you would usually configure it with a static IP address. You can configure DHCP to serve a static IP address for a container or you can define the address in the container’s config file.

To configure a static IP address that a container does not obtain using DHCP:
1. Edit `/container/name/rootfs/etc/sysconfig/network-scripts/ifcfg-iface`, where `iface` is the name of the network interface, and change the following line:

```plaintext
BOOTPROTO= dhcp
```

to read:

```plaintext
BOOTPROTO= none
```

2. Add the following line to `/container/name/config`:

```plaintext
lxc.network.ipv4 = xxx.xxx.xxx.xxx/prefix_length
```

where `xxx.xxx.xxx.xxx/prefix_length` is the IP address of the container in CIDR format, for example: `192.168.56.100/24`.

**Note**

The address must not already be in use on the network or potentially be assignable by a DHCP server to another system.

To configure DNS, edit the `hosts` and `resolv.conf` files under `/container/name/rootfs/etc`.

You might also need to configure the firewall on the host to allow access to a network service that is provided by a container.

### 9.3 Logging in to Containers

You can use the `lxc-console` command to log in to a running container.

```
[root@host ~]# lxc-console -n name [-t tty_number]
```

If you do not specify a tty number, you log in to the first available terminal.

For example, log in to a terminal on `ol6ctr1`:

```
[root@host ~]# lxc-console -n ol6ctr1
```

To exit an `lxc-console` session, type `Ctrl-A` followed by `Q`.

Alternatively, you can use `ssh` to log in to a container if you install the `lxc-0.9.0-2.0.5` package (or later version of this package).

**Note**

To be able to log in using `lxc-console`, the container must be running an `/sbin/mingetty` process for the terminal. Similarly, using `ssh` requires that the container is running the SSH daemon (`/usr/sbin/sshd`).

### 9.4 Creating Additional Containers

To clone an existing container, use the `lxc-clone` command, as shown in this example:

```
[root@host ~]# lxc-clone -o ol6ctr1 -n ol6ctr2
```

Alternatively, you can use the `lxc-create` command to create a container by copying the root file system from an existing system, container, or Oracle VM template. Specify the path of the root file system as the argument to the `--templatefs` template option:
This example copies the new container's `rootfs` from a snapshot of the `rootfs` that belongs to container `ol6ctr1`. The additional container is created in `/container/ol6ctr3` and a new `rootfs` snapshot is created in `/container/ol6ctr3/rootfs`.

For LXC version 1.0 and later, you must specify the `-B btrfs` option if you want to use the snapshot features of btrfs. For more information, see the `lxc-create(1)` manual page.

To change the host name of the container, edit the `HOSTNAME` settings in `/container/name/rootfs/etc/sysconfig/network` and `/container/name/rootfs/etc/sysconfig/network-scripts/ifcfg-iface`, where `iface` is the name of the network interface, such as `eth0`.

9.5 Monitoring and Shutting Down Containers

To display the containers that are configured, use the `lxc-ls` command on the host.

```
[root@host ~]# lxc-ls
ol6ctr1
ol6ctr2
```

To display the containers that are running on the host system, specify the `--active` option.

```
[root@host ~]# lxc-ls --active
ol6ctr1
```

To display the state of a container, use the `lxc-info` command on the host.

```
[root@host ~]# lxc-info -n ol6ctr1
state:  RUNNING
pid:    10171
```

A container can be in one of the following states: ABORTING, RUNNING, STARTING, STOPPED, or STOPPING. Although `lxc-info` might show your container to be in the RUNNING state, you cannot log in to it unless the `/usr/sbin/sshd` or `/sbin/mingetty` processes have started running in the container. You must allow time for the `/sbin/init` process in the container to first start networking and the various other services that you have configured.

To view the state of the processes in the container from the host, either run `ps -ef --forest` and look for the process tree below the `lxc-start` process or use the `lxc-attach` command to run the `ps` command in the container.

```
[root@host ~]# ps -ef --forest
```

```
UID   PID   PPID  C   STIME TTY     TIME  CMD
...  
root  3171 1   0 09:57 ?   00:00:00 lxc-start -n ol6ctr1 -d
root  3182 3171 0   09:57 ?   00:00:00  \
   /sbin/init
root  3441 3182 0   09:57 ?   00:00:00  \
   /sbin/dhclient -H ol6ctr1 ...
root  3464 3182 0   09:57 ?   00:00:00  \
   /sbin/rsyslogd ...
root  3493 3182 0   09:57 ?   00:00:00  \
   /usr/sbin/sshd
root  3500 3182 0   09:57 pts/5 00:00:00  \
   /sbin/mingetty ... /dev/console
root  3504 3182 0   09:57 pts/1 00:00:00  \
   /sbin/mingetty ... /dev/tty1
root  3506 3182 0   09:57 pts/2 00:00:00  \
   /sbin/mingetty ... /dev/tty2
root  3508 3182 0   09:57 pts/3 00:00:00  \
   /sbin/mingetty ... /dev/tty3
root  3510 3182 0   09:57 pts/4 00:00:00  \
   /sbin/mingetty ... /dev/tty4
...  
```
Monitoring and Shutting Down Containers

If a container appears not to be starting correctly, examining its process tree from the host will often reveal where the problem might lie.

Note that the process numbers differ from those of the same processes on the host, and that they all descend from the process 1, /sbin/init, in the container.

To suspend or resume the execution of a container, use the `lxc-freeze` and `lxc-unfreeze` commands on the host.

Alternatively, you can run a command such as `halt` or `init 0` while logged in to the container.

As shown in the example, you are returned to the shell prompt on the host.

To shut down a container by terminating its processes immediately, use `lxc-stop` with the `-k` option.
If you are debugging the operation of a container, using `lxc-stop` is the quickest method as you would usually destroy the container and create a new version after modifying the template script.

To monitor the state of a container, use the `lxc-monitor` command.

```
[root@host ~]# lxc-monitor -n ol6ctr1
'ol6ctr1' changed state to [STARTING]
'ol6ctr1' changed state to [RUNNING]
'ol6ctr1' changed state to [STOPPING]
'ol6ctr1' changed state to [STOPPED]
```

To wait for a container to change to a specified state, use the `lxc-wait` command.

```
lxc-wait -n $CTR -s ABORTING && lxc-wait -n $CTR -s STOPPED && 
    echo "Container $CTR terminated with an error."
```

### 9.6 Starting a Command Inside a Running Container

**Note**
The `lxc-attach` command is supported by UEK R3 with the `lxc-0.9.0-2.0.4` package or later.

You can use `lxc-attach` to execute an arbitrary command inside a container that is already running from outside the container, for example:

```
[root@host ~]# lxc-attach -n ol6ctr1 -- ps aux
```

For more information, see the `lxc-attach(1)` manual page.

### 9.7 Controlling Container Resources

Linux containers use cgroups in their implementation, and you can use the `lxc-cgroup` command to control the access that a container has to system resources relative to other containers. For example, to display the CPU cores to which a container can run on, enter:

```
[root@host ~]# lxc-cgroup -n ol6ctr1 cpuset.cpus
0-7
```

To restrict a container to cores 0 and 1, you would enter a command such as the following:

```
[root@host ~]# lxc-cgroup -n ol6ctr1 cpuset.cpus 0,1
```

To change a container’s share of CPU time and block I/O access, you would enter:

```
[root@host ~]# lxc-cgroup -n ol6ctr2 cpu.shares 256
[root@host ~]# lxc-cgroup -n ol6ctr2 blkio.weight 500
```

Limit a container to 256 MB of memory when the system detects memory contention or low memory; otherwise, set a hard limit of 512 MB:

```
[root@host ~]# lxc-cgroup -n ol6ctr2 memory.soft_limit_in_bytes 268435456
[root@host ~]# lxc-cgroup -n ol6ctr2 memory.limit_in_bytes 53687091
```

To make the changes to a container’s configuration permanent, add the settings to the file `/container/name/config`, for example:

```
# Permanently tweaked resource settings
```
9.8 Configuring ulimit Settings for an Oracle Linux Container

The `ulimit` setting of an Oracle Linux container created using the `lxc-oracle` template script honors the values of `ulimit` settings such as `memlock` and `nofile` in the container's version of `/etc/security/limits.conf` provided that these values are lower than or equal to the values on the host system.

The values of `memlock` and `nofile` determine the maximum amount of address space in kilobytes that can be locked into memory by a user process and the maximum number of file descriptors that a user process can have open at the same time.

If you require a higher `ulimit` value for a container, increase the value of the settings in `/etc/security/limits.conf` on the host, for example:

```
#<domain>      <type>  <item>         <value>
*              soft    memlock       1048576
*              hard    memlock       2097152
*              soft    nofile        5120
*              hard    nofile        10240
```

A process can use the `ulimit` built-in shell command or the `setrlimit()` system call to raise the current limit for a shell above the soft limit. However, the new value cannot exceed the hard limit unless the process is owned by `root`.

You can use `ulimit` to set or display the current soft and hard values on the host or from inside the container, for example:

```
[root@host ~]# echo "host: nofile = $(ulimit -n)"
host: nofile = 1024
[root@host ~]# echo "host: nofile = $(ulimit -H -n)"
host: nofile = 4096
[root@host ~]# ulimit -n 2048
[root@host ~]# echo "host: nofile = $(ulimit -n)"
host: nofile = 2048
[root@host ~]# lxc-attach -n ol6ctr1 -- echo "container: nofile = $(ulimit -n)"
container: nofile = 1024
```

Note

Log out and log in again or, if possible, reboot the host before starting the container in a shell that uses the new soft and hard values for `ulimit`.

9.9 Configuring Kernel Parameter Settings for Oracle Linux Containers

If you specify the `--privileged` option with the `lxc-oracle` template script, you can adjust the values of certain kernel parameters for a container under the `/proc` hierarchy.

The container mounts `/proc` read-only with the following exceptions, which are writable:

- `/proc/sys/kernel/msgmax`
- `/proc/sys/kernel/msgmnb`
Deleting Containers

- /proc/sys/kernel/msgmni
- /proc/sys/kernel/sem
- /proc/sys/kernel/shmall
- /proc/sys/kernel/shmmax
- /proc/sys/kernel/shmmni
- /proc/sys/net/ipv4/conf/default/accept_source_route
- /proc/sys/net/ipv4/conf/default/rp_filter
- /proc/sys/net/ipv4/ip_forward

Each of these parameters can have a different value than that configured for the host system and for other containers running on the host system. The default value is derived from the template when you create the container. Oracle recommends that you change a setting only if an application requires a value other than the default value.

Note
Prior to UEK R3 QU6, the following host-only parameters were not visible within the container due to kernel limitations:
- /proc/sys/net/core/rmem_default
- /proc/sys/net/core/rmem_max
- /proc/sys/net/core/wmem_default
- /proc/sys/net/core/wmem_max
- /proc/sys/net/ipv4/ip_local_port_range
- /proc/sys/net/ipv4/tcp_syncookies

With UEK R3 QU6 and later, these parameters are read-only within the container to allow Oracle Database and other applications to be installed. You can change the values of these parameters only from the host. Any changes that you make to host-only parameters apply to all containers on the host.

9.10 Deleting Containers

To delete a container and its snapshot, use the lxc-destroy command as shown in the following example.

```
[root@host ~]# lxc-destroy -n ol6ctr2
Delete subvolume '/container/ol6ctr2/rootfs'
```

This command also deletes the rootfs subvolume.

9.11 Running Application Containers

You can use the lxc-execute command to create a temporary application container in which you can run a command that is effectively isolated from the rest of the system. For example, the following command creates an application container named guest that runs sleep for 100 seconds.
Running Application Containers

While the container is active, you can monitor it by running commands such as `lxc-ls --active` and `lxc-info -n guest` from another window.

<table>
<thead>
<tr>
<th>[root@host ~]# lxc-execute -n guest -- sleep 100</th>
</tr>
</thead>
</table>

If you need to customize an application container, you can use a configuration file. For example, you might want to change the container's network configuration or the system directories that it mounts.

The following example shows settings from a sample configuration file where the rootfs is mostly not shared except for mount entries to ensure that `lxc-init` and certain library and binary directory paths are available.

```
lxc.utsname = guest
lxc.tty = 1
lxc.pts = 1
lxc.rootfs = /tmp/guest/rootfs
lxc.mount.entry=/lib /tmp/guest/rootfs/lib none ro,bind 0 0
lxc.mount.entry=/usr/libexec /tmp/guest/rootfs/usr/lib none ro,bind 0 0
lxc.mount.entry=/lib64 /tmp/guest/rootfs/lib64 none ro,bind 0 0
lxc.mount.entry=/usr/lib64 /tmp/guest/rootfs/usr/lib64 none ro,bind 0 0
lxc.mount.entry=/bin /tmp/guest/rootfs/bin none ro,bind 0 0
lxc.mount.entry=/usr/bin /tmp/guest/rootfs/usr/bin none ro,bind 0 0
lxc.cgroup.cpuset.cpus=1
```

The mount entry for `/usr/libexec` is required so that the container can access `/usr/libexec/lxc/lxc-init` on the host system.

The example configuration file mounts both `/bin` and `/usr/bin`. In practice, you should limit the host system directories that an application container mounts to only those directories that the container needs to run the application.

```
<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>To avoid potential conflict with system containers, do not use the /container directory for application containers.</td>
</tr>
</tbody>
</table>
```

You must also configure the required directories under the rootfs directory:

```
<table>
<thead>
<tr>
<th>[root@host ~]# TMPDIR=/tmp/guest/rootfs</th>
</tr>
</thead>
<tbody>
<tr>
<td>[root@host ~]# mkdir -p $TMPDIR/lib $TMPDIR/usr/lib $TMPDIR/lib64 $TMPDIR/usr/lib64 $TMPDIR/bin $TMPDIR/usr/bin $TMPDIR/dev/pts $TMPDIR/dev/shm $TMPDIR/proc</td>
</tr>
</tbody>
</table>
```

In this example, the directories include `/dev/pts`, `/dev/shm`, and `/proc` in addition to the mount point entries defined in the configuration file.

You can then use the `-f` option to specify the configuration file (config) to `lxc-execute`:

```
<table>
<thead>
<tr>
<th>[root@host ~]# lxc-execute -n guest -f config -- ps -ef</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>UID</th>
<th>PID</th>
<th>PPID</th>
<th>C</th>
<th>STIME</th>
<th>TTY</th>
<th>TIME</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>08:56</td>
<td>?</td>
<td>00:00:00</td>
<td>/usr/lib/lxc/lxc-init -- ps -ef</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>08:56</td>
<td>?</td>
<td>00:00:00</td>
<td>ps -ef</td>
</tr>
</tbody>
</table>
```

This example shows that the `ps` command runs as a child of `lxc-init`.

As for system containers, you can set cgroup entries in the configuration file and use the `lxc-cgroup` command to control the system resources to which an application container has access.
Note

`lxc-execute` is intended to run application containers that share the host's root file system, and not to run system containers that you create using `lxc-create`. Use `lxc-start` to run system containers.

For more information, see the `lxc-execute(1)` and `lxc.conf(5)` manual pages.

9.12 For More Information About Linux Containers

For more information about LXC, see [https://wiki.archlinux.org/index.php/Linux_Containers](https://wiki.archlinux.org/index.php/Linux_Containers) and the LXC manual pages.
Chapter 10 Docker

Table of Contents

10.1 About Docker .................................................................................................................. 127
10.2 Installing and Configuring the Docker Engine ......................................................... 128
10.3 Upgrading to Docker Version 1.8.1 and Later .......................................................... 130
10.4 Restarting the Docker Engine ....................................................................................... 130
10.5 Enabling Non-root Users to Run Docker Commands .................................................. 130
10.6 Pulling Oracle Linux Images from the Docker Hub Registry ...................................... 131
  10.6.1 Enabling or Disabling Docker Content Trust ......................................................... 131
10.7 Creating and Running Docker Containers ................................................................. 132
  10.7.1 Configuring How Docker Restarts Containers ...................................................... 134
  10.7.2 Controlling Capabilities and Making Host Devices Available to Containers ........ 134
  10.7.3 Accessing the Host's Process ID Namespace ....................................................... 135
  10.7.4 Mounting a Host's root File System in Read-Only Mode ........................................ 135
10.8 Creating a Docker Image from an Existing Container .................................................. 135
10.9 Creating a Docker Image from a Dockerfile ............................................................... 137
10.10 Communicating Between Docker Containers ........................................................... 139
  10.10.1 Example of Linking Database and HTTP Server Containers ............................... 141
10.11 Accessing External Files from Docker Containers .................................................... 145
10.12 Creating and Using Data Volume Containers ............................................................. 146
10.13 Moving Data Between Docker Containers and the Host ............................................ 147
10.14 Using Labels to Define Metadata ................................................................................. 148
10.15 Defining the Logging Driver ........................................................................................ 149
10.16 About Image Digests ..................................................................................................... 149
10.17 Specifying Control Groups for Containers ............................................................... 150
10.18 Limiting CPU Usage by Containers .......................................................................... 150
10.19 Making a Container Use the Host's UTS Namespace .................................................. 150
10.20 Setting ulimit Values on Containers ......................................................................... 150
10.21 Building Images with Resource Constraints ............................................................ 150
10.22 Committing, Exporting and Importing Images ............................................................ 151
10.23 For More Information About Docker .......................................................................... 151

This chapter describes how to use Docker, which is an open-source, distributed-application platform that is based on LXC.

10.1 About Docker

Docker allows you to create and distribute applications across Oracle Linux systems and other operating systems that support Docker. Docker consists of the Docker Engine, which packages and runs the applications, and the Docker Hub Registry, which shares the applications in a Software-as-a-Service (SaaS) cloud.

The Docker Engine is designed primarily to run single applications in a similar manner to LXC application containers that provide a degree of isolation from other processes running on a system. The Docker Engine is available for Oracle Linux 6 (x86_64) and Oracle Linux 7 (x86_64).

The Docker Hub Registry hosts applications as Docker images and provides services that allow you to create and manage a Docker environment. You must register with the Docker Hub Registry to be able to access its resources and services.
Note
The Docker Hub Registry is owned and maintained by Docker, Inc. Oracle makes Docker images available on the Docker Hub Registry that you can download and use with the Docker Engine. Oracle does not have any control otherwise over the content of the Docker Hub Registry site or its repositories.

For more information, see https://docs.docker.com.

10.2 Installing and Configuring the Docker Engine

To install and configure the Docker Engine on an Oracle Linux 6 system:

1. If your system is registered with ULN, enable the ol6_x86_64_addons channel.
   
   If you use Oracle Public Yum, enable the ol6_addons repository in the /etc/yum.repos.d/public-yum-ol6.repo file, for example:
   
   ```
   [ol6_addons]
   name=Oracle Linux $releasever Add ons ($basearch)
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
   enabled=1
   ```
   
   You can download an up-to-date version of this file from http://public-yum.oracle.com/public-yum-ol6.repo.

2. Install the docker-engine package.
   
   ```
   # yum install docker-engine
   ```

3. By default, the Docker Engine uses the device mapper to manage Docker containers. As with LXC, there are benefits to using the snapshot features of btrfs instead.

   To configure the Docker Engine to use btrfs instead of the device mapper:
   
   a. Use yum to install the btrfs-progs package.
      
      ```
      # yum install btrfs-progs
      ```
   
   b. Create a btrfs file system on a suitable device such as /dev/sdb in this example:
      
      ```
      # mkfs.btrfs /dev/sdb
      ```
   
   c. Mount the file system on /var/lib/docker.
      
      ```
      # mount /dev/sdb /var/lib/docker
      ```
   
   d. Add an entry for /var/lib/docker to the /etc/fstab file.
      
      ```
      /dev/sdb /var/lib/docker btrfs defaults 0 0
      ```

4. Edit /etc/sysconfig/docker to configure global networking options, for example:
   
   - If your system needs to use a web proxy to access the Docker Hub Registry, add the following lines:
     
     ```
     export HTTP_PROXY="proxy_URL:port"
     export HTTPS_PROXY="proxy_URL:port"
     ```
     
     Replace proxy_URL and port with the appropriate URL and port number for your web proxy.
• To configure IPv6 support in version 1.5 and later of Docker, add the `--ipv6` option to OPTIONS, for example:

```
OPTIONS="--ipv6"
```

With IPv6 enabled, Docker assigns the link-local IPv6 address `fe80::1` to the bridge `docker0`.

If you want Docker to assign global IPv6 addresses to containers, additionally specify the IPv6 subnet to the `--fixed-cidr-v6` option, for example:

```
OPTIONS="--ipv6 --fixed-cidr-v6='2001:db8:1::/64'"
```

For more information about configuring Docker networking, see [https://docs.docker.com/articles/networking/](https://docs.docker.com/articles/networking/).

5. In version 1.5 and later of Docker, the docker service unshares its mount namespace to resolve `device busy` issues with the device mapper storage driver. However, this configuration breaks `autofs` in the host system and prevents you from accessing subsequently mounted volumes in Docker containers. The workaround is to stop the Docker service from unsharing its mount namespace.

Edit `/etc/init.d/docker` and change the following line:

```
"$unshare" -m -- $exec -d $other_args &>> $logfile &
```

so that it reads:

```
$exec -d $other_args &>> $logfile &
```

**Note**

You might need to reapply this workaround if you update the `docker` package and the change to `/etc/init.d/docker` is overwritten.

6. Start the `docker` service and configure it to start at boot time:

```
# service docker start
# chkconfig docker on
```

**Note**

If you have installed the `mlocate` package, it is recommended that you modify the `PRUNEPATHS` entry in `/etc/updatedb.conf` to prevent `updatedb` from indexing directories below `/var/lib/docker`, for example:

```
PRUNEPATHS="/media /tmp /var/lib/docker /var/spool /var/tmp"
```

This entry prevents `locate` from reporting files that belong to Docker containers.

To check that the `docker` service is running, use the following command:

```
# service docker status
```

You can also use the `docker` command to display information about the configuration and version of the Docker Engine, for example:

```
# docker info
Containers: 0
Images: 6
Storage Driver: btrfs
```
Upgrading to Docker Version 1.8.1 and Later

The Docker package for version 1.8.1 and later is `docker-engine`, which conflicts with the `docker` package used by previous versions of Docker.

To upgrade Docker to version 1.8.1 or later:

1. Stop the `docker` service.

   ```
   # service docker stop
   ```

2. Remove the `docker` package and install the `docker-engine` package.

   ```
   # yum remove docker
   # yum install docker-engine
   ```

3. Start the `docker` service.

   ```
   # service docker start
   ```

10.4 Restarting the Docker Engine

If you edit the `/etc/sysconfig/docker` configuration file while the `docker` service is running, you must restart the service to make the changes take effect.

To restart the `docker` service, enter the following command:

```
# service docker restart
```
2. Restart the `docker` service:

```
# service docker restart
```

The UNIX socket `var/run/docker.sock` is now readable and writable by members of the `docker` group.

3. Add the users that should have Docker access to the `docker` group:

```
# usermod -a -G docker user1 ...
```

## 10.6 Pulling Oracle Linux Images from the Docker Hub Registry

### Note

An Internet connection is required to pull images from the Docker Hub Registry.

You can obtain images for Oracle Linux for use with the Docker Engine from the `oraclelinux` repository at the Docker Hub Registry. For a list of the Oracle Linux images that are available, see [https://registry.hub.docker.com/_/oraclelinux/](https://registry.hub.docker.com/_/oraclelinux/).

To download an Oracle Linux image, use the `docker pull` command, for example:

```
# docker pull oraclelinux:6.6
```

Pulling repository oraclelinux
9ac13076d2b5: Download complete
511136ea3c5a: Download complete
ad98bd7101f2: Download complete
3e24531dbb17: Download complete
Status: Downloaded newer image for oraclelinux:6.6

To display a list of the images that you have downloaded to a system, use the `docker images` command, for example:

```
[root@host ~]# docker images
REPOSITORY    TAG         IMAGE ID       CREATED       VIRTUAL SIZE
oraclelinux   6           9ac13076d2b5   5 days ago    319.4 MB
oraclelinux   6.6         9ac13076d2b5   5 days ago    319.4 MB
```

Each image in a repository is distinguished by its `tag` value and its unique ID. In the following example, the tags `6` and `6.6` refer to the same image ID for Oracle Linux 6 as do the tags `7`, `7.0`, and `latest` for Oracle Linux 7.

```
[root@host ~]# docker images
REPOSITORY    TAG         IMAGE ID       CREATED       VIRTUAL SIZE
oraclelinux   6           9ac13076d2b5   5 days ago    319.4 MB
oraclelinux   6.6         9ac13076d2b5   5 days ago    319.4 MB
oraclelinux   latest      073ded22ac0f   5 days ago    265.2 MB
oraclelinux   7           073ded22ac0f   5 days ago    265.2 MB
oraclelinux   7.0         073ded22ac0f   5 days ago    265.2 MB
```

When new images are made available for Oracle Linux updates, the tags `6, 7`, and `latest` are updated in the `oraclelinux` repository to refer to the appropriate newest version.

### 10.6.1 Enabling or Disabling Docker Content Trust

Docker version 1.8 introduces Content Trust, which allows you to verify the authenticity, integrity, and publication date of Docker images that are made available on the Docker Hub Registry.
Creating and Running Docker Containers

By default, Content Trust is disabled. To enable Content Trust for signing and verifying Docker images that you build, push to, or pull from the Docker Hub Registry, set the `DOCKER_CONTENT_TRUST` environment variable, for example:

```bash
# export DOCKER_CONTENT_TRUST=1
```

If you use `sudo` to run Docker commands, specify the `-E` option to preserve the environment or use `visudo` to add the following line to `/etc/sudoers`:

```bash
Defaults        env_keep += "DOCKER_CONTENT_TRUST"
```

For individual `docker build`, `docker push`, or `docker pull` commands, you can specify the `--disable-content-trust=false` and `--disable-content-trust=true` options to enable or disable Content Trust.

For more information, see [https://blog.docker.com/2015/08/content-trust-docker-1-8/](https://blog.docker.com/2015/08/content-trust-docker-1-8/) and [https://docs.docker.com/security/trust/content_trust/](https://docs.docker.com/security/trust/content_trust/).

10.7 Creating and Running Docker Containers

You use the `docker run` command to run an application inside a container, for example:

```
[root@host ~]# docker run -i -t --name guest oraclelinux:6.6 /bin/bash
[root@guest ~]# cat /etc/oracle-release
Oracle Linux Server release 6.6
[root@guest ~]#
```

This example runs an interactive `bash` shell using the Oracle Linux 6 image named `oraclelinux:6.6` to provide the container. The `-t` and `-i` options allow you to use a pseudo-terminal to run the container interactively. `[root@host ~]` and `[root@guest ~]` represent the prompts shown by the host and by the container respectively. The actual prompt displayed by the container might be different.

The `--name` option specifies the name `guest` for the container instance. Docker does not remove the container when it exits and we can restart it at a later time.

If an image does not already exist on your system, the Docker Engine performs a `docker pull` operation to download the image from the Docker Hub Registry (or from another repository that you specify) as shown in the following example:

```
[root@host ~]# docker run -i -t --rm oraclelinux:7.0
Unable to find image 'oraclelinux:7.0' locally
Pulling repository oraclelinux
073ded22ac0f: Download complete
511336ea3c5a: Download complete
ad98bd7101f2: Download complete
cbb192d7f4cf: Download complete
Status: Downloaded newer image for oraclelinux:7.0
[root@guest /]# cat /etc/oracle-release
Oracle Linux Server release 7.0
[root@guest /]# exit
exit
[root@host ~]#
```

Because we specified the `--rm` option instead of naming the container, Docker removes the container when it exits and we cannot restart it.

From another shell window, you can use the `docker ps` command to display information about the containers that are currently running, for example:

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
77bacba845e2        oraclelinux:6.6    /bin/bash               11 minutes ago      Up 11 minutes       guest
```
The container named `guest` with the ID `77bacba845e2` is currently running the command `/bin/bash`. It is more convenient to manage a container by using its name than by its ID.

To display the processes that a container is running, use the `docker top` command:

```
[root@host ~]# docker top guest
UID    PID    PPID   C   STIME   TTY     TIME       CMD
root   7474   1958   1   15:40   pts/2   00:00:00   /bin/bash
```

In version 1.3.0 and later of Docker, you can use the `docker exec` command to run additional processes in a container that is already running, for example:

```
[root@host ~]# docker exec -i -t guest bash
[...]
```

You can exit a container by typing `Ctrl-D` or `exit` at the `bash` command prompt inside the container or by using the `docker stop` command:

```
[root@host ~]# docker stop guest
```

The `-a` option to `docker ps` displays all containers that are currently running or that have exited.

```
[root@host ~]# docker ps -a
CONTAINER ID  IMAGE              COMMAND    CREATED  STATUS             PORTS  NAMES
77bacba845e2  oraclelinux:6.6    ...        ...      Exited (0) 9 seconds ago  guest
8a1b9b19bb70  oraclelinux:6.6    ...        ...      Up 38 seconds   ...       newguest
```

You can use `docker start` to restart a stopped container. After reattaching to it, the contents remain unchanged from the last time that you used the container.

```
[root@host ~]# docker start -a -i guest
[...]
```

Because the container preserves any changes that you make to it, you can reconfigure files and install packages in the container without worrying that your changes will disappear.

If you need to remove a container permanently so that you can create a new container with the same name, use the `docker rm` command:

```
[root@host ~]# docker rm guest
```

**Note**

If you specify the `--rm` option when you run a container, Docker removes the container when the container exits. You cannot combine the `--rm` option with the `-d` option.
In version 1.2.0 and later of Docker, specifying the `-f` option to `docker rm` kills a running container before removing it. In previous versions, the same command stops the container before removing it. If you want to stop a container safely, use `docker stop`.

You can use the `docker logs` command to watch what is happening inside a container, for example:

```
[root@host ~]# docker logs -f guest
...
bash-4.x# touch /tmp/foobar
bash-4.x# exit
bash-4.x# bash-4.x# ls -l /tmp/foobar
-rw-r--r--. 1 root root 0 Aug 29 05:23 /tmp/foobar
```

The `-f` option causes the command to update its output as events happen in the container. Type `Ctrl-C` to exit the command.

You can obtain full information about a container in JSON format by using the `docker inspect` command. This command also allows you to retrieve specified elements of the configuration, for example:

```
[root@host ~]# docker inspect --format='{{ .State.Running }}' guest
true
```

### 10.7.1 Configuring How Docker Restarts Containers

To specify how you want Docker to handle a container when it exits, you can use the `--restart` option with `docker run` in version 1.2.0 and later of Docker and with `docker create` in version 1.3.0 and later:

- `--restart=always`: Docker always attempts to restart the container when the container exits.
- `--restart=no`: Docker does not attempt to restart the container when the container exits. This is the default policy.
- `--restart=on-failure[=max-retry]`: Docker attempts to restart the container if the container returns a non-zero exit code. You can optionally specify the maximum number of times that Docker will try to restart the container.

### 10.7.2 Controlling Capabilities and Making Host Devices Available to Containers

If you specify the `--privileged=true` option to `docker create` or `docker run`, the container has access to all the devices on the host, which can present a security risk. For more precise control, you can use the `--cap-add` and `--cap-drop` options in version 1.2.0 and later of Docker to restrict the capabilities of a container, for example:

```
[root@host ~]# docker run --cap-add=ALL --cap-drop=NET_ADMIN -i -t --rm oraclelinux:6.6 /bin/bash
```

This example grants all capabilities except `NET_ADMIN` to the container so that it is not able to perform network-administration operations. For more information, see the `capabilities(7)` manual page.

To make only individual devices on the host available to a container, you can use the `--device` option with `docker run` in version 1.2.0 and later of Docker and with `docker create` in version 1.3.0 and later:
Accessing the Host's Process ID Namespace

--device=host_devname[;container_devname[;permissions]]

--device

host_devname is the name of the host device.
container_devname is an optional name for the device in the container.
permissions optionally specifies the permissions that the container has on the device, which is a combination of the following codes:

- m Grants mknod permission. For example, you can use mknod to set permission bits or the SELinux context for the device file.
- r Grants read permission.
- w Grants write permission. For example, you can use a command such as mkfs to format the device.

For example, --device=/dev/sdd:/dev/xvdd:r would make the host device /dev/sdd available to the container as the device /dev/xvdd with read-only permission.

Warning
Do not make block devices that can easily be removed from the system available to untrusted containers.

10.7.3 Accessing the Host's Process ID Namespace

In version 1.5 and later of Docker, you can make the host's process ID namespace visible from inside a container by specifying the --pid=host option to docker run. A suggested use of this mode is to debug host processes by using containerized debugging tools.

Warning
Host mode is inherently insecure as it gives a container full access to D-Bus and other system services on the host.

10.7.4 Mounting a Host's root File System in Read-Only Mode

In version 1.5 and later of Docker, you can mount the host's root file system in read-only mode from a container by specifying the --read-only=true option to docker create or docker run. You can use this mode to restrict write access by a containerized application.

10.8 Creating a Docker Image from an Existing Container

If you modify the contents of a container, you can use the docker commit command to save the current state of the container as an image.

The following example demonstrates how to modify an container based on the oraclelinux:6.6 image so that it can run an Apache HTTP server. After stopping the container, the image mymod/httpd:v1 is created from it.

To create an Apache server image from an oraclelinux:6.6 container:

1. Run the bash shell inside a container named guest:

```bash
[guest@host ~]# docker run -i -t --name guest oraclelinux:6.6 /bin/bash
[guest@guest ~]#
```
Creating a Docker Image from an Existing Container

2. If you use a web proxy, edit the yum configuration on the guest as described in Section 2.2.1, “Configuring Use of a Proxy Server”.

3. Install the httpd package:

   ```bash
   [root@guest ~]\# yum install httpd
   ```

4. If required, create the web content to be displayed under the /var/www/html directory hierarchy on the guest.

5. Exit the guest by using the `docker stop` command on the host:

   ```bash
   [root@host ~]\# docker stop guest
   ```

6. Create the image mymod/httpd with the tag v1 using the ID of the container that you stopped:

   ```bash
   [root@host ~]\# docker commit -m "ol6 + httpd" -a "A N Other" \\
   "docker ps -l -q" mymod/httpd:v1
   ```

   ```text
   8594abec905e6374db51bed1bf8b208804cb60d96b285efb897db581a01676e9
   ```

   Use the `--m` and `--a` options to document the image and its author. The command returns the full version of the new image's ID.

   If you use the `docker images` command, the new image now appears in the list:

   ```text
   REPOSITORY    TAG         IMAGE ID       CREATED       VIRTUAL SIZE
   mymod/httpd   v1          8594abec905e   2 minutes ago 938.5 MB
   oraclelinux   6           9ac13076d2b5   5 days ago    319.4 MB
   oraclelinux   6.6         9ac13076d2b5   5 days ago    319.4 MB
   oraclelinux   latest      073ded22ac0f   5 days ago    265.2 MB
   oraclelinux   7           073ded22ac0f   5 days ago    265.2 MB
   oraclelinux   7.0         073ded22ac0f   5 days ago    265.2 MB
   ```

7. Remove the container named `guest`.

   ```bash
   # docker rm guest
   ```

   You can now use the new image to create a container that works as a web server, for example:

   ```bash
   # docker run -d --name newguest -p 8080:80 mymod/httpd:v1 /usr/sbin/httpd -D FOREGROUND
   7afbbefec5191f6363e149f385ae10ed0ba88f1c5455a5fcb93e575ef6a3e63
   ```

   The `--d` option runs the command non-interactively in the background and displays the full version of the unique container ID. The `-p 8080:80` option maps port 80 in the guest to port 8080 on the host. You can view the port mapping by running `docker ps` or `docker port`, for example:

   ```text
   CONTAINER ID  IMAGE         COMMAND    CREATED    STATUS    NAMES
   7afbbefec519  mymod/httpd:v1           ...        ...      newguest
   ```

   ```bash
   [root@host ~]\# docker port newguest 80
   0.0.0.0:8080->80/tcp
   ```

### Note

The `docker ps` command displays the short version of the container ID. You can use the `--no-trunc` option to display the long version.

The default IP address value of 0.0.0.0 means that the port mapping applies to all network interfaces on the host. You can restrict the IP addresses to which the remapping applies by using multiple `-p` options, for example:
Creating a Docker Image from a Dockerfile

You can view the web content served by the guest by pointing a browser at port 8080 on the host. If you access the content from a different system, you might need to allow incoming connections to the port on the host, for example:

```
[root@host ~]# iptables -I INPUT -p tcp --dport 8080 -j ACCEPT
[root@host ~]# service iptables save
```

If you need to remove an image, use the `docker rmi` command:

```
[root@host ~]# docker rmi mymod/httpd:v1
Untagged: mymod/httpd:v1
Deleted: 7afbbefec519f632e149f85ae10ed0ba88f1c545daad18cb930e575ef6a3e63
```

Note
From version 1.8 of Docker, you cannot remove the image of a running container.

In a production environment, using the `docker commit` command to create an image does not provide a convenient record of how you created the image so you might find it difficult to recreate an image that has been lost or become corrupted. The preferred method for creating an image is to set up a `Dockerfile`, in which you define instructions that allow Docker to build the image for you. See Section 10.9, “Creating a Docker Image from a Dockerfile”.

10.9 Creating a Docker Image from a Dockerfile

You use the `docker build` command to create a Docker image from the definition contained in a `Dockerfile`.

The following example demonstrates how to build an image named `mymod/httpd` with the tag `v2` based on the `oraclelinux:6.6` image so that it can run an Apache HTTP server.

To create a Docker image from a Dockerfile:

1. Make a directory where you can create the Dockerfile, for example:

   ```
   # mkdir -p /var/docker_projects/mymod/httpd
   ```

   Note
   You do not need to create the Dockerfile on the same system on which you want to deploy containers that you create from the image. The only requirement is that the Docker Engine can access the Dockerfile.

2. In the new directory, create the Dockerfile, which is usually named `Dockerfile`. The following Dockerfile contents are specific to the example:

   ```
   # Dockerfile that modifies oraclelinux:6.6 to include an Apache HTTP server
   FROM oraclelinux:6.6
   MAINTAINER A N Other <another@mydom.com>
   RUN sed -i -e '/^[\main\]/aproxy=http://proxy.mydom.com:80' /etc/yum.conf
   RUN yum -y install httpd
   RUN echo "HTTP server running on guest" > /var/www/html/index.html
   EXPOSE 80
   ENTRYPOINT /usr/sbin/httpd -D FOREGROUND
   ```

   The `#` prefix in the first line indicates that the line is a comment. The remaining lines start with the following instruction keywords that define how Docker creates the image:
Creating a Docker Image from a Dockerfile

**ENTRYPOINT** Specifies the command that a container created from the image always runs. In this example, the command is `/usr/sbin/httpd -D FOREGROUND`, which starts the HTTP server process.

**EXPOSE** Defines that the specified port is available to service incoming requests. You can use the `-p` or `-P` options with `docker run` to map this port to another port on the host. Alternatively, you can use the `--link` option with `docker run` to allow another container to access the port over Docker’s internal network (see Section 10.10, “Communicating Between Docker Containers”).

**FROM** Defines the image that Docker uses as a basis for the new image.

**MAINTAINER** Defines who is responsible for the Dockerfile.

**RUN** Defines the commands that Docker runs to modify the new image. In the example, the RUN lines set up the web proxy, install the `httpd` package, and create a simple home page for the server.

For more information about other instructions that you can use in a Dockerfile, see http://docs.docker.com/reference/builder/.

3. Use the `docker build` command to create the image:

```
# docker build --tag=mymod/httpd:v2 --file=/var/docker_projects/mymod/httpd/Dockerfile
Uploading context  2.56 kB
Uploading context
Step 0 : FROM oraclelinux:6.6
    ---> 3e4b5e722ab9
Step 1 : MAINTAINER A N Other <another@mymod.com>
    ---> Using cache
    ---> debe47cef9b8
Step 2 : RUN sed -i -e '/^\[main\]/aproxy=http://proxy.mymod.com:80' /etc/yum.conf
    ---> Using cache
    ---> 7719ba64938e
Step 3 : RUN yum -y install httpd
    ---> Running in 14bdeddf332
Loaded plugins: fastestmirror
Determining fastest mirrors
Setting up Install Process
Resolving Dependencies
    --> Running transaction check
    ---> Package httpd.x86_64 0:2.2.15-31.0.1.el6_5 will be installed
    .
    .
    Installed:
    httpd.x86_64 0:2.2.15-31.0.1.el6_5
Dependency Installed:
    apr.x86_64 0:1.3.9-5.el6_2
    apr-util.x86_64 0:1.3.9-3.el6_0.1
    apr-util-ldap.x86_64 0:1.3.9-3.el6_0.1
    httpd-tools.x86_64 0:2.2.15-31.0.1.el6_5
    mailcap.noarch 0:2.1.31-2.el6
Complete!
    ---> 76ef62c00e49
Removing intermediate container e02588a269b9
Step 4 : RUN echo "HTTP server running on guest" > /var/www/html/index.html
    ---> Running in c8081a1d0c96
    ---> 75e22b04a1ad
```
Having built the image, you can test it by creating a container instance named `newguest2`:

```
[root@host ~]# docker run -d --name newguest2 -P mymod/httpd:v2
31b334b993cfbdc714b7c4f723c352c8de842823505b6f11a08bf960e0398e7
```

You do not need to specify `/usr/sbin/httpd -D FOREGROUND` as this command is now built into the container.

The `-P` option specifies that Docker should map the ports exposed by the guest to available ports in the range 49000 through 49900 on the host.

You can use `docker inspect` to return the host port that Docker maps to TCP port 80:

```
[root@host ~]# docker inspect --format='{{ .NetworkSettings.Ports }}' newguest2
map[80/tcp:[map[HostIp:0.0.0.0 HostPort:49153]]]
```

In this example, TCP port 80 in the guest is mapped to TCP port 49153 on the host.

You can view the web content served by the guest by pointing a browser at port 49153 on the host. If you access the content from a different system, you might need to allow incoming connections to the port on the host, for example:

```
[root@host ~]# iptables -I INPUT -p tcp --state NEW -m tcp --dport 49153 -j ACCEPT
[root@host ~]# service iptables save
```

You can also use `curl` to test that the server is working:

```
[root@host ~]# curl http://localhost:49153
HTTP server running on guest
[root@host ~]# ssh auser@10.0.0.23
auser@10.0.0.23's password: password
Last login: Fri Aug 29 13:48:58 2014 from 192.168.0.1
[auser@10.0.0.23 ~]# curl 192.168.0.2:49153
HTTP server running on guest
```

### 10.10 Communicating Between Docker Containers

You can use the `-link` option with `docker run` to make network connection information about a server container available to a client container. The client container uses a private networking interface to access the exposed port in the server container. Docker sets environment variables about the server container in the client container that describe the interface and the ports that are available.

The following example demonstrates how to link an `oraclelinux:6.6`-based client container with an HTTP server container based on the `mymod/httpd:v2` image that you created in Section 10.9, “Creating a Docker Image from a Dockerfile”.

---

**Glossary**

- **Container**: A lightweight, standalone, isolated process that runs in its own address space.
- **Dockerfile**: A small text file that contains a set of instructions used to build an image.
- **Image**: A Docker image is a container runtime that contains a full software stack, including OS, libraries, and apps.
- **Network**:
  - **Network namespaces**: A method for isolating a process or container from other processes or containers on the same host.
  - **Network containers**: Containers that are part of a network namespace and share network connectivity with other containers on the host.
  - **Private network**: A network that is isolated from the host network and only visible to the container.
  - **Public network**: A network that is visible to all applications running on the host.
- **Port**: A specific route through the network that allows a service to be accessed by others on the network.
- **Run**: A command that runs the Docker daemon and starts a new container.
- **Service**: A Docker service is a set of containers that are managed as a single logical unit.
- **Volume**: A directory on the host that is mounted into a container as a volume. Volumes can be used to persist data across container restarts.

---

**Further Reading**

- [Docker Documentation](https://docs.docker.com)
- [Docker Community](https://community.docker.com)
- [Docker Conferences and Events](https://events.docker.com)

---

**About the Author**

Alice Smith is a software engineer at Docker, Inc. She specializes in container networking and has a background in distributed systems and cloud computing. Alice holds a degree in computer science from Stanford University and has been working in the tech industry since 2005.
To create an HTTP server and client containers that are linked:

1. Create an HTTP server container named `http_server`:

   ```
   [root@host ~]# docker run -d --name http_server mymod/httpd:v2
   a47169154222329eed6672128755cd9fdd24d0f227f8e01678ef136bb66d03
   ```

2. Create a client container named `client1` that runs the `bash` shell and is linked to the `http_server` container:

   ```
   [root@host httpd]# docker run --rm -t -i --name client1 --link http_server:server \
   oraclelinux:6.6 /bin/bash
   [root@client1 ~]#
   ```

   The argument `http_server:server` to the `--link` option aliases the name `http_server` as `server`. Docker converts the alias to upper case (`SERVER`) and uses this string when setting up the names of the environment variables on the client.

You can now view the environment variables in the `client1` container. You can also use `ping` to detect the server container by name or IP address, and use `curl` to access the web server running on the server:

   ```
   [root@client1 ~]# env
   HOSTNAME=10815c22e5b4
   TERM=xterm
   SERVER_PORT=tcp://172.17.0.16:80
   PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
   PWD=/
   SERVER_PORT_80_TCP_PORT=80
   SERVER_PORT_80_TCP_ADDR=172.17.0.16
   SERVER_PORT_80_TCP=tcp://172.17.0.16:80
   SERVER_PORT_80_TCP_PROTO=tcp
   SHLVL=1
   SERVER_NAME=/client1/server
   HOME=/
   _=/usr/bin/env
   [root@client1 ~]# ping -c 1 server
   PING server (172.17.0.16) 56(84) bytes of data.
   64 bytes from server (172.17.0.16): icmp_seq=1 ttl=64 time=0.105 ms

   --- server ping statistics ---
   1 packets transmitted, 1 received, 0% packet loss, time 0ms
   rtt min/avg/max/mdev = 0.105/0.105/0.105/0.000 ms
   [root@client1 ~]# ping -c 1 172.17.0.16
   PING 172.17.0.16 (172.17.0.16) 56(84) bytes of data.
   64 bytes from 172.17.0.16: icmp_seq=1 ttl=64 time=0.171 ms

   --- 172.17.0.16 ping statistics ---
   1 packets transmitted, 1 received, 0% packet loss, time 0ms
   rtt min/avg/max/mdev = 0.171/0.171/0.171/0.000 ms
   [root@client1 ~]# curl http://server
   HTTP server running on guest
   [root@client1 ~]# curl http://172.17.0.16
   HTTP server running on guest
   ```

You can start multiple client container instances with different names, each of which can access port 80 on the server container. Docker assigns a different IP address to each client. As shown in the following example output, Docker creates an entry for the server in the `/etc/hosts` files on each client but it does not create entries for the names of the client containers themselves:

   ```
   [root@client1 ~]# cat /etc/hosts
   172.17.0.17  10815c22e5b4
   127.0.0.1  localhost
   ::1  localhost ip6-localhost ip6-loopback
   fe00::0  ip6-localnet
   ```
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.16 server

[root@client1 ~]# ping -c 1 client2
ping: unknown host client2

[root@client1 ~]# ping -c 1 172.17.0.18
PING 172.17.0.18 (172.17.0.18) 56(84) bytes of data.
64 bytes from 172.17.0.18: icmp_seq=1 ttl=64 time=0.268 ms

--- 172.17.0.18 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.268/0.268/0.268/0.000 ms

By default, the clients are visible to each other on the private network only by their IP addresses.

The `docker ps` command shows the containers that are running:

```
CONTAINER ID  IMAGE            COMMAND          CREATED  STATUS        PORTS  NAMES
449abeac3041  oraclelinux:6.6  /bin/bash        ...      Up 1 minutes         client2
10815c22e5b4  oraclelinux:6.6  /bin/bash        ...      Up 2 minutes         client1
a47169154222  mymod/httpd:v2   /usr/sbin/httpd  ...      Up 3 minutes  80/tcp
```

The NAMES column shows that http_server is linked to client1 and client2 as server. The PORTS column shows that Docker has not remapped TCP port 80 on http_server to another port on the host.

### 10.10.1 Example of Linking Database and HTTP Server Containers

**Note**

This simple example demonstrates how to link containers. You should not use it as the basis of a production application.

The following example demonstrates how to link a container that is running a MySQL server with a container running an HTTP server.

First of all, we define a Dockerfile for the MySQL server, which we place in the `/var/docker_projects/mymod/mysql` directory:

```
FROM oraclelinux:6.6
ENV http_proxy http://proxy.mydom.com:80
RUN yum install -y mysql-server
ADD my.cnf /etc/my.cnf
ADD run.sh /opt/run.sh
RUN chmod 744 /opt/run.sh
ENTRYPOINT /opt/run.sh
```

The instruction keywords define how to create the image:

- **ADD**: Copy the files `my.cnf` and `run.sh` from the `/var/docker_projects/mymod/mysql` directory to `/etc/my.cnf` and `/opt/run.sh` in the container.
- **ENTRYPOINT**: Specify that the container always runs `/opt/run.sh`.
- **ENV**: Define the web proxy in the build environment (as an alternative to modifying `/etc/yum.conf`).
- **FROM**: Define `oraclelinux:6.6` as a basis for the new image.
- **RUN**: Install the `mysql-server` package and make the `/opt/run.sh` script executable.
The `my.cnf` file in `/var/docker_projects/mymod/mysql` contains the database configuration:

```ini
[mysqld]
bind-address=0.0.0.0
console=1
general_log=1
general_log_file=/dev/stdout
log_error=/dev/stderr
collation-server=utf8_unicode_ci
character-set-server=utf8
datadir=/var/lib/mysql
```

The `run.sh` file in `/var/docker_projects/mymod/mysql` contains the shell script for starting the database:

```bash
#!/bin/bash
chown -R mysql:mysql /var/lib/mysql
mysql_install_db --user=mysql > /dev/null
/usr/libexec/mysqld --user mysql --bootstrap << SQL
   FLUSH PRIVILEGES;
   GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;
   CREATE USER dbuser IDENTIFIED BY 'secret';
   CREATE DATABASE MYDB;
   USE MYDB;
   GRANT ALL ON MYDB.* to 'dbuser'@'%';
SQL
/usr/bin/mysqld_safe --user mysql
```

Having set up the Dockerfile and the files `my.cnf` and `run.sh`, we can now build the image `mymod/mysql:v1` and create an instance of this container named `db` that uses the standard MySQL connection port (3306):

```
# docker build --tag="mymod/mysql:v1" --file="/var/docker_projects/mymod/mysql/Dockerfile"
```

We next define a Dockerfile for the HTTP server, which we place in the `/var/docker_projects/mymod/httpd2` directory:

```
FROM oraclelinux:6.6
ENV http_proxy http://proxy.mydom.com:80
RUN yum install -y httpd perl perl-DBI.x86_64 libdbi-dbd-mysql.x86_64 perl-DBD-MySQL.x86_64
ADD version.pl /var/www/cgi-bin/version.pl
RUN chmod 755 /var/www/cgi-bin/version.pl
```

```
# docker run -d --name db -p 3306:3306 mymod/mysql:v1
ba8816c594051ca892aa89828b5cf33464a7b4616b56177266033d3311d0e00d
```

```
FROM oraclelinux:6.6
ENV http_proxy http://proxy.mydom.com:80
RUN yum install -y httpd perl perl-DBI.x86_64 libdbi-dbd-mysql.x86_64 perl-DBD-MySQL.x86_64
ADD version.pl /var/www/cgi-bin/version.pl
RUN chmod 755 /var/www/cgi-bin/version.pl
```

142
Example of Linking Database and HTTP Server Containers

This Dockerfile modifies the container's HTTP server configuration file (/etc/httpd/conf/httpd.conf) to allow the use of CGI scripts and installs the following Perl scripts from the /var/docker_projects/mymod/httpd2 directory:

**version.pl** Connects to the database and returns its version.

```perl
#!/usr/bin/perl
use DBI;

print "Content-type: text/html\n\n";
my $dbh = DBI->connect(
    "dbi:mysql:dbname=MYDB:host=db",
    "dbuser",
    "secret",
    { RaiseError => 1 },
) or die $DBI::errstr;

my $sth = $dbh->prepare("SELECT VERSION()");
$sth->execute();

my $ver = $sth->fetch();
print "Version = ", @$ver, "\n";

$sth->finish();
$dbh->disconnect();
```

**initdb.pl** Sets up the database and populates a table with several entries.

```perl
#!/usr/bin/perl
use strict;
use DBI;

print "Content-type: text/html\n\n";
my $dbh = DBI->connect(
    "dbi:mysql:dbname=MYDB:host=db",
    "dbuser",
    "secret",
    { RaiseError => 1 }
) or die $DBI::errstr;

$dbh->do("DROP TABLE IF EXISTS PEOPLE");
$dbh->do("CREATE TABLE People(Id INT PRIMARY KEY, Name TEXT, Age INT) ENGINE=InnoDB");
$dbh->do("INSERT INTO People VALUES(1,'Alice',42)");
$dbh->do("INSERT INTO People VALUES(2,'Bobby',27)");
$dbh->do("INSERT INTO People VALUES(3,'Carol',29)");
$dbh->do("INSERT INTO People VALUES(4,'Daisy',20)");
$dbh->do("INSERT INTO People VALUES(5,'Eddie',35)");
$dbh->do("INSERT INTO People VALUES(6,'Frank',21)");

my @noerr = ('Rows inserted in People table');
print @noerr;
print "\n";

my $sth = $dbh->prepare("SELECT * FROM People");
```

---

143
Example of Linking Database and HTTP Server Containers

```perl
$sth->execute();
for ( 1 .. $sth->rows() ) {
    my ($id, $name, $age) = $sth->fetchrow();
    print "$id $name $age\n";
}
$sth->finish();
$dbh->disconnect();
```

**doquery.pl** Performs a simple query on the database, using the command argument as data for the query.

```perl
#!/usr/bin/perl
use strict;
use DBI;

print "Content-type: text/html\n\n";

my $dbh = DBI->connect(
    "dbi:mysql:dbname=MYDB;host=db",
    "dbuser",
    "secret",
    { RaiseError => 1 },
) or die $DBI::errstr;

my $sth = $dbh->prepare("SELECT * FROM People WHERE Age > $ARGV[0]" );
$sth->execute();

my $fields = $sth->{NUM_OF_FIELDS};
my $rows = $sth->rows();
print "Selected $rows row(s) with $fields field(s)\n";
for ( 1 .. $rows ) {
    my ($id, $name, $age) = $sth->fetchrow();
    print "$id $name $age\n";
}
$sth->finish();
$dbh->disconnect();
```

Having set up the Dockerfile and the Perl scripts, we can now build the image `mymod/httpd:v3` and create an instance of this container named `web`, which is linked to the `db` container and which uses the standard HTTP server port (80) on the host:

```bash
# docker build --tag="mymod/httpd:v3" --file="/var/docker_projects/mymod/httpd2/Dockerfile"

```

```bash
Uploading context 142.8 kB
Uploading context
Step 0 : FROM oraclelinux:6.6    ---> d56e767abb61
Step 1 : ENV http_proxy http://proxy.mydom.com:80    ---> Using cache
    ---> f92df8c449eb
...  
Step 11 : ENTRYPOINT /usr/sbin/httpd -D FOREGROUND
    ---> Running in 3203c57a7204
    ---> 10dc2d7624d3
Removing intermediate container 3203c57a7204
Successfully built 10dc2d7624d3
# docker run -d --name web -p 80:80 --link db:db mymod/httpd:v3
ba8816c340513a892aa8982b5c33464a7bba616b56177266033d3311d0e00d
```

Finally, we can use `curl` to test the operation of the CGI scripts with the database:

```bash
$ curl http://10.0.0.2/cgi-bin/version.pl
Version = 5.1.73-log
$ curl http://10.0.0.2/cgi-bin/initdb.pl
```
10.11 Accessing External Files from Docker Containers

You can use the `-v` option with `docker run` to make a file or file system available inside a container. The following example demonstrates how to make web pages on the host available to an HTTP server running in a container.

Create the file `/var/www/html/index.html` on the host and run an HTTP server container that mounts this file:

```
[root@host ~]# echo "This text was created in a file on the host" > /var/www/html/index.html
[root@host ~]# docker run -d --name newguest3 -P \\
```

The `:ro` modifier specifies that a container mounts a file or file system read-only. To mount a file or file system read-writable, specify the `:rw` modifier instead or omit the modifier altogether.

Check that the HTTP server is not running on the host:

```
[root@host ~]# curl http://localhost
curl: (7) couldn't connect to host
[root@host ~]# service httpd status
httpd is stopped
```

Even though an HTTP server is not running directly on the host, you can display the new web page served by the `newguest3` container:

```
[root@host ~]# docker inspect --format='{{ .NetworkSettings.Ports }}' newguest3
map[80/tcp:[map[HostIp:0.0.0.0 HostPort:49153]]]
[root@host ~]# curl http://localhost:49153
This text was created in a file on the host
```

Any changes that you make to the `/var/www/html/index.html` file on the host are reflected in the mounted file in the container:

```
[root@host ~]# echo "Change the file on the host" > /var/www/html/index.html
[root@host ~]# curl http://localhost:49153
Change the file on the host
```

Even if you delete the file on the host, it is still visible in the container:

```
[root@host ~]# rm /var/www/html/index.html
rm: remove regular file `/var/www/html/index.html'? y
[root@host ~]# ls -l /var/www/html/index.html
```
Creating and Using Data Volume Containers

It is not possible to use a Dockerfile to define how to mount a file or file system from a host. Docker applications are intended to be portable and it is unlikely that a file or file system that exists on the original host would be available on another system. If you want external file data to be portable, you can encapsulate it in a data volume container. See Section 10.12, “Creating and Using Data Volume Containers”.

10.12 Creating and Using Data Volume Containers

If you specify a single directory argument to the -v option of docker run, Docker creates the directory in the container and marks it as a data volume that other containers can mount. You can also use the VOLUME instruction in a Dockerfile to create this data volume in an image. A container that contains such a data volume is called a data volume container. After populating the data volume with files, you can use the --volumes-from option of docker run to have other containers mount the volume and access its data.

The following example creates a data volume container that an HTTP server container can use as the source of its web content.

To create a data volume container image and an instance of a data volume container from this image:

1. Make a directory where you can create the Dockerfile for the data volume container image, for example:

   ```bash
   # mkdir -p /var/docker_projects/mymod/dvc
   ```

2. In the new directory, create a Dockerfile that defines the image for a data volume container:

   ```bash
   # Dockerfile that modifies oraclelinux:6.6 to create a data volume container
   FROM oraclelinux:6.6
   MAINTAINER A N Other <another@mydom.com>
   RUN mkdir -p /var/www/html
   RUN echo "This is the content for file1.html" > /var/www/html/file1.html
   RUN echo "This is the content for file2.html" > /var/www/html/file2.html
   RUN echo "This is the content for index.html" > /var/www/html/index.html
   VOLUME /var/www/html
   ENTRYPOINT /usr/bin/tail -f /dev/null
   ```

   The RUN instructions create a /var/www/html directory that contains three simple files.

   The VOLUME instruction makes the directory available as a volume that other containers can mount by using the --volumes-from option to docker run.

   The ENTRYPOINT instruction specifies the command that a container created from the image always runs. To prevent the container from exiting, the /usr/bin/tail -f /dev/null command blocks until you use a command such as docker stop dvc1 to stop the container.

3. Use the docker build command to create the image:

   ```bash
   [root@host ~]# docker build --tag="mymod/dvc:v1" --file="/var/docker_projects/mymod/dvc/Dockerfile"
   ```

The following example creates a data volume container that an HTTP server container can use as the source of its web content.

To create a data volume container image and an instance of a data volume container from this image:

1. Make a directory where you can create the Dockerfile for the data volume container image, for example:

   ```bash
   # mkdir -p /var/docker_projects/mymod/dvc
   ```

2. In the new directory, create a Dockerfile that defines the image for a data volume container:

   ```bash
   # Dockerfile that modifies oraclelinux:6.6 to create a data volume container
   FROM oraclelinux:6.6
   MAINTAINER A N Other <another@mydom.com>
   RUN mkdir -p /var/www/html
   RUN echo "This is the content for file1.html" > /var/www/html/file1.html
   RUN echo "This is the content for file2.html" > /var/www/html/file2.html
   RUN echo "This is the content for index.html" > /var/www/html/index.html
   VOLUME /var/www/html
   ENTRYPOINT /usr/bin/tail -f /dev/null
   ```

   The RUN instructions create a /var/www/html directory that contains three simple files.

   The VOLUME instruction makes the directory available as a volume that other containers can mount by using the --volumes-from option to docker run.

   The ENTRYPOINT instruction specifies the command that a container created from the image always runs. To prevent the container from exiting, the /usr/bin/tail -f /dev/null command blocks until you use a command such as docker stop dvc1 to stop the container.

3. Use the docker build command to create the image:

   ```bash
   [root@host ~]# docker build --tag="mymod/dvc:v1" --file="/var/docker_projects/mymod/dvc/Dockerfile"
   ```
Step 2: RUN mkdir -p /var/www/html
    --> Running in fa94df7dd3af
    ---> 503132e87939
Removing intermediate container fa94df7dd3af
Step 3: RUN echo "This is the content for file1.html" > /var/www/html/file1.html
    --> Running in f98a14371e72
    ---> e63ba0d36d88
Removing intermediate container f98a14371e72
Step 4: RUN echo "This is the content for file2.html" > /var/www/html/file2.html
    --> Running in d0dc96ad53c
    ---> 2f2e2b3d207
Removing intermediate container d0dc96ad53c
Step 5: RUN echo "This is the content for index.html" > /var/www/html/index.html
    --> Running in fe39aa35b577
    ---> 89f3c1db1c3
Removing intermediate container fe39aa35b577
Step 6: VOLUME /var/www/html
    --> Using cache
    ---> 91d394fd412e
Step 7: ENTRYPOINT /usr/bin/tail -f /dev/null
    --> Running in 91b872b93b35
    ---> c6e934249b9f
Removing intermediate container 91b872b93b35
Successfully built 91d394fd412e

4. Create an instance of the data volume container, for example dvc1:

```bash
[root@host ~]# docker run -d --name dvc1 mymod/dvc:v1 tail -f /dev/null
1c8973e3c24e4f195e2b90ba5cb44af930121897c0e697407a8f83270589c6f1
```

To test that other containers can mount the data volume (/var/www/html) from dvc1, create a container named websvr that runs an HTTP server and mounts its data volume from dvc1.

```bash
[root@host ~]# docker run -d --volumes-from dvc1 --name websvr -P mymod/httpd:v2
008ce3decbf98ce50f6e3f3cfc7618d48ce9dcfca8c291d90d4d179118d4c1b3
```

After finding out the correct port to use on the host, use curl to test that websvr correctly serves the content of all three files that were set up in the image.

```bash
[root@host ~]# docker port websvr 80
0.0.0.0:49154
[root@host ~]# curl http://localhost:49154
This is the content for index.html
[root@host ~]# curl http://localhost:49154/file1.html
This is the content for file1.html
[root@host ~]# curl http://localhost:49154/file2.html
This is the content for file2.html
```

10.13 Moving Data Between Docker Containers and the Host

You can use the -v option of docker run to copy volume data between a data volume container and the host. For example, you might want to back up the data so that you can restore it to the same data volume container or to copy it to a different data volume container.

The examples in this section assume that Docker is running two instances of the data volume container image mymod/dvc:v1 that is described in Section 10.12, “Creating and Using Data Volume Containers”. You can use the following commands to start these containers:

```bash
# docker run -d --name dvc1 mymod/dvc:v1
# docker run -d --name dvc2 mymod/dvc:v1
```

To copy the data from a data volume to the host, mount the volume from another container and use the cp command to copy the data to the host, for example:
Using Labels to Define Metadata

From version 1.6.0 of Docker, you can use labels to add metadata to the Docker daemon and to Docker containers and images.

In the Dockerfile, a `LABEL` instruction defines an image label that can contain one or more key-value pairs, for example:

```
LABEL com.mydom.dept="ITGROUP" \
     com.mydom.version="1.0.0-ga" \
     com.mydom.is-final \
     com.mydom.released="June 6, 2015"
```

**10.14 Using Labels to Define Metadata**

This example uses a transient, interactive container named `guest` to extract the contents of the archive to `dvc2`.
In this example, each key name is prefixed by the domain name in reverse DNS form (com.mydom.) to guard against name-space conflicts. Key values are always expressed as strings and are not interpreted by Docker. If you omit the value, you can use the presence or absence of the key in the metadata to encode information such as the release status. The backslash characters allow you to extend the label definition across several lines.

You can use the `docker inspect` command to display the labels that are associated with an image, for example:

```
$ docker inspect 7ac15076dcc1
...
"Labels": {
  "com.mydom.dept": "ITGROUP",
  "com.mydom.version": "1.0.0-ga",
  "com.mydom.is-final": "",
  "com.mydom.release-date": "June 6, 2015"
}
...
```

You can use the `--filter "label=key=value"` option with the `docker images` and `docker ps` commands to list the images and running containers on which a metadata value has been set, for example:

```
$ docker images --filter "label=com.mydom.dept='DEVGROUP'"
$ docker ps --filter "label=com.mydom.is-beta2"
$ docker ps --filter "label=env=Oracle\ Linux\ 6"
```

For containers, you can use `--label key=value` options with the `docker create` and `docker run` commands to define key-value pairs, for example:

```
$ docker run -i -t --rm testapp:1.0 --label run="11" --label platform="Oracle Linux 6"
```

For the Docker engine, you can use `--label key=value` options if you start `docker` from the command line or you can add the options to `OPTIONS` in `/etc/sysconfig/docker`, for example:

```
OPTIONS="--label com.mydom.dept='DEVGROUP'"
```

As containers and the Docker daemon are transitory and run in a known environment, it is not usually necessary to apply reverse domain name prefixes to key names.

### 10.15 Defining the Logging Driver

From version 1.6.0 of Docker, you can use the `--log-driver` option with the `docker create` and `docker run` commands to specify the logging driver that a container should use:

- **json-file** Write log messages to a JSON file that you can examine by using the `docker logs` command, for example:

  ```
  $ docker logs --follow --timestamps=false container_name
  ```

  This is the default logging driver.

- **none** Disable logging.

- **syslog** Write log messages to `syslog`.

### 10.16 About Image Digests

From version 1.6.0 of Docker, registry version 2 or later images can be identified by their digest (for example, `sha256:digest_value_in_hexadecimal`). You can list the digest by specifying the `--
Specifying Control Groups for Containers

You can use a digest with the `docker create`, `docker pull`, `docker rmi`, and `docker run` commands and with the `FROM` instruction in a Dockerfile.

10.17 Specifying Control Groups for Containers

From version 1.6.0 of Docker, you can use the `--cgroup-parent` option with the `docker create` command to specify the control group (`cgroup`) in which a container should run.

10.18 Limiting CPU Usage by Containers

To control a container's CPU usage, you can use the `--cpu-period` and `--cpu-quota` options with the `docker create` and `docker run` commands from version 1.7.0 of Docker onward.

The `--cpu-quota` option specifies the number of microseconds that a container has access to CPU resources during a period specified by `--cpu-period`. As the default value of `--cpu-period` is 100000, setting the value of `--cpu-quota` to 25000 limits a container to 25% of the CPU resources. By default, a container can use all available CPU resources, which corresponds to a `--cpu-quota` value of -1.

10.19 Making a Container Use the Host's UTS Namespace

By default, a container runs with a UTS namespace (which defines the system name and domain) that is different from the UTS namespace of the host. To make a container use the same UTS namespace as the host, you can use the `--uts=host` option with the `docker create` and `docker run` commands from version 1.7.0 of Docker onward. This setting allows the container to track the UTS namespace of the host or to set the host name and domain from the container.

10.20 Setting ulimit Values on Containers

The `--ulimit` option to `docker run` allows you to specify `ulimit` values for a container, for example:

```
$ docker run -i -t --rm myapp:2.0 --ulimit nofile=128:256 --ulimit nproc=32:64
```

This example sets a soft limit of 128 open files and 32 child processes and a hard limit of 256 open files and 64 child processes on the container.

From version 1.6.0 of Docker, you can set default `ulimit` values for all containers by specifying `--default-ulimit` options to the Docker daemon. For example, you can add the options to OPTIONS in `/etc/sysconfig/docker`:

```
OPTIONS="--ulimit nofile=1280:2560 --ulimit nproc=256:512"
```

Any `ulimit` values that you specify for a container override the default values that you set for the daemon.

10.21 Building Images with Resource Constraints

From version 1.6.0 of Docker, you can specify cgroup resource constraints to `docker build`, for example:

```
# docker build --cpu-shares=100 --memory=1024m \
--tag="mymod/myapp:1.0" --file="/var/docker_projects/mymod/myapp/Dockerfile"
```

Any containers that you generate from the image inherit these resource constraints.

You can use the `docker stats` command to display a container's resource usage, for example:
You can use the `docker commit` command to save the current state of a container to an image.

```bash
docker commit [--author="name"] \ 
[--change="instructions"]... \ 
[--message="text"] \ 
[--pause=false] container [repository[:tag]]
```

You can use this image to create new containers, for example to debug the container independently of the existing container.

You can use the `docker export` command to export a container to another system as an image tar file.

```bash
docker export [--output="filename"] container
```

**Note**

You need to export separately any data volumes that the container uses. See Section 10.13, "Moving Data Between Docker Containers and the Host".

To import the image tar file, use `docker import` and specify the image URL or read the file from the standard input.

```bash
docker import [--change="instructions"]... URL [repository[:tag]]
docker import [--change="instructions"]... - [repository[:tag]] < filename
```

From version 1.6.0 of Docker, you can use `--change` options with `docker commit` and `docker import` to specify Dockerfile instructions that modify the configuration of the image, for example:

```bash
docker commit --change "LABEL com.mydom.status='Debug'" 7ac15076dcc1 mymod/debugimage:v1
```

For `docker commit`, you can specify the following instructions: ADD, CMD, COPY, ENTRYPOINT, ENV, EXPOSE, FROM, LABEL, MAINTAINER, RUN, USER, VOLUME, and WORKDIR.

For `docker import`, you can specify the following instructions: CMD, ENTRYPOINT, ENV, EXPOSE, ONBUILD, USER, VOLUME, and WORKDIR.

**10.23 For More Information About Docker**

For more information about Docker, see https://www.docker.com/ and the Docker manual pages.
Chapter 11 HugePages

Table of Contents

11.1 About HugePages .................................................................................................................. 153
11.2 Configuring HugePages for Oracle Database ........................................................................ 153
11.3 For More Information About HugePages ............................................................................. 155

This chapter describes how to set up the HugePages feature on a system that is running several Oracle Database instances.

11.1 About HugePages

The HugePages feature enables the Linux kernel to manage large pages of memory in addition to the standard 4KB (on x86 and x86_64) or 16KB (on IA64) page size. If you have a system with more than 16GB of memory running Oracle databases with a total System Global Area (SGA) larger than 8GB, you should enable the HugePages feature to improve database performance.

Note

The Automatic Memory Management (AMM) and HugePages features are not compatible in Oracle Database 11g and later. You must disable AMM to be able to use HugePages.

The memory allocated to huge pages is pinned to primary storage, and is never paged nor swapped to secondary storage. You reserve memory for huge pages during system startup, and this memory remains allocated until you change the configuration.

In a virtual memory system, the tables store the mappings between virtual addresses and physical addresses. When the system needs to access a virtual memory location, it uses the page tables to translate the virtual address to a physical address. Using huge pages means that the system needs to load fewer such mappings into the Translation Lookaside Buffer (TLB), which is the cache of page tables on a CPU that speeds up the translation of virtual addresses to physical addresses. Enabling the HugePages feature allows the kernel to use \texttt{hugetlb} entries in the TLB that point to huge pages. The \texttt{hugetlb} entries mean that the TLB entries can cover a larger address space, requiring many fewer entries to map the SGA, and releasing entries that can map other portions of the address space.

With HugePages enabled, the system uses fewer page tables, reducing the overhead for maintaining and accessing them. Huges pages remain pinned in memory and are not replaced, so the kernel swap daemon has no work to do in managing them, and the kernel does not need to perform page table lookups for them. The smaller number of pages reduces the overhead involved in performing memory operations, and also reduces the likelihood of a bottleneck when accessing page tables.

Huge pages are 4MB in size on x86, 2MB on x86_64, and 256MB on IA64.

11.2 Configuring HugePages for Oracle Database

The steps in this section are for configuring HugePages on a 64-bit Oracle Linux system running one or more Oracle Database instances.

To configure HugePages:

1. Verify that the \texttt{soft} and \texttt{hard} values in kilobytes of \texttt{memlock} that are configured in \texttt{/etc/security/limits.conf} are slightly smaller than the amount of installed memory. For example, if the system has 64GB of RAM, the values shown here would be appropriate:
Configuring HugePages for Oracle Database

2. Log in as the Oracle account owner (usually `oracle`) and use the following command to verify the value of `memlock`:

$$ulimit -l$$

3. If your system is running Oracle Database 11g or later, disable AMM by setting the values of both of the initialization parameters `memory_target` and `memory_max_target` to 0.

If you start the Oracle Database instances with a server parameter file, which is the default if you created the database with the Database Configuration Assistant (DBCA), enter the following commands at the SQL prompt:

```
SQL> alter system set memory_target=0;
System altered.
SQL> alter system set memory_max_target=0;
System altered.
```

If you start the Oracle Database instances with a text initialization parameter file, manually edit the file so that it contains the following entries:

```
memory_target = 0
memory_max_target = 0
```

4. Verify that all the Oracle Database instances are running (including any Automatic Storage Management (ASM) instances) as they would run on the production system.

5. Create the file `hugepages_settings.sh` with the following content (taken from the My Oracle Support (MOS) note 401749.1).

```
#!/bin/bash
#
# hugepages_settings.sh
#
# Linux bash script to compute values for the recommended HugePages/HugeTLB configuration
#
# Note: This script does calculation for all shared memory segments available when the script is run, no matter it is an Oracle RDBMS shared memory segment or not.
# Check for the kernel version
KERN=`uname -r | awk -F. '{ printf("%d.%d",$1,$2); }'`
# Find out the HugePage size
HPG_SZ=`grep Hugepagesize /proc/meminfo | awk {'print $2'}`
# Start from 1 pages to be on the safe side and guarantee 1 free HugePage
NUM_PG=1
# Cumulative number of pages required to handle the running shared memory segments
for SEG_BYTES in `ipcs -m | awk {'print $5'} | grep "^[0-9]"`;
  do
    MIN_PG=`echo "\$SEG_BYTES/\$HPG_SZ" | bc -q`;
    if [ $MIN_PG -gt 0 ]; then
      NUM_PG=`echo "\$NUM_PG+$MIN_PG" | bc -q`;
    fi
done
# Finish with results
case $KERN in
  '2.4') HUGETLB_POOL=`echo "\$NUM_PG*\$HPG_SZ" | bc -q`;
    echo "Recommended setting: vm.hugetlb_pool = \$HUGETLB_POOL" ;
    '2.6') echo "Recommended setting: vm.nr_hugepages = \$NUM_PG" ;
    *) echo "Unrecognized kernel version $KERN. Exiting." ;
```
6. Make the file executable, and run it to calculate the recommended value for the `vm.nr_hugepages` kernel parameter.

```bash
$ chmod u+x ./hugepages_settings.sh
$ ./hugepages_settings.sh

Recommended setting: `vm.nr_hugepages` = 22960
```

7. As `root`, edit the file `/etc/sysctl.conf` and set the value of the `vm.nr_hugepages` parameter to the recommended value.

```bash
vm.nr_hugepages = 22960
```

8. Stop all the database instances and reboot the system.

After rebooting the system, verify that the database instances (including any ASM instances) have started, and use the following command to display the state of the huge pages.

```bash
# grep ^Huge /proc/meminfo
HugePages_Total:   22960
HugePages_Free:     2056
HugePages_Rsvd:     2016
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

The value of `HugePages_Free` should be smaller than that of `HugePages_Total`, and the value of `HugePages_Rsvd` should be greater than zero. As the database instances allocate pages dynamically and proactively as required, the sum of the `Hugepages_Free` and `HugePages_Rsvd` values is likely to be smaller than the total SGA size.

If you subsequenty change the amount of system memory, add or remove any database instances, or change the size of the SGA for a database instance, use `hugepages_settings.sh` to recalculate the value of `vm.nr_hugepages`, readjust the setting in `/etc/sysctl.conf`, and reboot the system.

### 11.3 For More Information About HugePages

For more information about using HugePages with Oracle Database, see [http://docs.oracle.com/cd/E11882_01/server.112/e10839/appi_vlm.htm#CACDCGAH](http://docs.oracle.com/cd/E11882_01/server.112/e10839/appi_vlm.htm#CACDCGAH).
Chapter 12 Using kexec for Fast Rebooting

Table of Contents

12.1 About kexec ................................................................. 157
12.2 Setting up Fast Reboots of the Current Kernel ..................................................... 157
12.3 Controlling Fast Reboots ....................................................................................... 158
12.4 For More Information About kexec ....................................................................... 158

This chapter describes how to configure the kexec to enable fast rebooting of a system.

12.1 About kexec

kexec is a fast-boot mechanism that allows a kernel to boot from inside the context of a kernel that is already running without initializing the BIOS or firmware, performing memory and device discovery, or passing through the boot-loader stage.

When you reboot a system, the init process goes to run-level 6 and runs the /etc/init.d/halt script. If you have configured kexec on the system, the script will execute the kexec -e command, and cause the system to bypass the standard boot sequence.

The total amount of time saved when rebooting is highly dependent on your server, and can range from several tens of seconds to several minutes.

Caution

As fast reboots bypass device initialization, some devices might fail to work correctly, or a device driver might malfunction if it sees a device in an unexpected state. Before enabling this feature on your systems, test it to ensure that the hardware devices and their drivers continue to behave correctly across fast reboots.

12.2 Setting up Fast Reboots of the Current Kernel

To set up your system so that you can enable fast reboots of the current kernel:

1. Create the file /etc/init.d/runkexec with the following contents:

```bash
#!/bin/sh
#
# runkexec
#
### BEGIN INIT INFO
# Provides: runkexec
# Required-Start: 
# Required-Stop: 
# Default-Stop: 
# Description: Enable or disable fast system rebooting 
# Short-Description: enable or disable fast system rebooting
### END INIT INFO

KV=`uname -r`

case "$1" in
  start|restart|load|reload)
    kexec -l --append="`cat /proc/cmdline`" --initrd=/boot/initramfs-$KV.img \
    /boot/vmlinuz-$KV
```
Controlling Fast Reboots

```bash
if [ $1 ]; then
  case $1 in
    start|restart|load|reload|stop|unload|status)
      echo "$1"
      ;;
  esac
  exit 0
fi

if [ $2 = -u ]; then
  kexec -u && echo "Target kexec kernel unloaded."
  ;;
fi

status)
  echo "Status not available for kexec."
  ;;
*)
  echo "Usage: runkexec {start|restart|load|reload|stop|unload|status}"
  exit 2
esac
exit 0
```

2. Set the ownership and mode of the file.

```bash
# chown root:root /etc/init.d/runkexec
# chmod 755 /etc/init.d/runkexec
```

3. Create the symbolic link `S00kexec` to the file from the `/etc/rc1.d` directory.

```bash
# ln -s /etc/init.d/runkexec /etc/rc1.d/S00kexec
```

4. To enable fast reboots without needing to reboot the system, enter:

```bash
# service runkexec start
```

### 12.3 Controlling Fast Reboots

Once you have enabled fast reboots, running `reboot` will cause the system to shut down all services and then directly execute the kernel image.

If you want to execute the new kernel immediately without shutting down any services, use the following commands.

```bash
# sync; umount -a; kexec -e
```

To re-enable fast reboots of the current kernel at any time, enter:

```bash
# service runkexec restart
```

Alternatively, specify a different kernel that you want the system to reboot into by entering the following command:

```bash
# kexec -l --append="kernel_options" --initrd=initial_ramdisk_image kernel_path
```

where `kernel_options` are the options that you want to specify to the kernel, and `initial_ramdisk_image` and `kernel_path` are the paths to the initial ramdisk image and the kernel that you want to use.

To unload a target kernel, enter:

```bash
# service runkexec stop
```

Alternatively, you can enter:

```bash
# kexec -u
```

### 12.4 For More Information About kexec

For more information, see the `kexec(8)` manual page.
Chapter 13 DTrace

Table of Contents

13.1 About DTrace ................................................................. 159
13.2 Installing and Configuring DTrace ........................................ 159
  13.2.1 Changing the Mode of the DTrace Helper Device .................. 161
  13.2.2 Loading DTrace Kernel Modules .................................... 161
13.3 Differences Between DTrace on Oracle Linux and Oracle Solaris .......... 162
13.4 Calling DTrace from the Command Line .................................. 163
13.5 About Programming for DTrace ......................................... 166
13.6 Introducing the D Programming Language ................................ 167
  13.6.1 Probe Clauses ......................................................... 168
  13.6.2 Pragmas .............................................................. 169
  13.6.3 Global Variables .................................................. 169
  13.6.4 Predicates .......................................................... 170
  13.6.5 Scalar Arrays and Associative Arrays ............................... 171
  13.6.6 Pointers and External Variables ................................. 172
  13.6.7 Address Spaces .................................................... 173
  13.6.8 Thread-local Variables ............................................. 174
  13.6.9 Speculations ....................................................... 174
  13.6.10 Aggregations ....................................................... 176
13.7 DTrace Command Examples .............................................. 177
13.8 Tracing User-Space Applications ....................................... 180
  13.8.1 Examining the Stack Trace of a User-Space Application .......... 181
13.9 For More Information About DTrace .................................... 182

This chapter introduces the dynamic tracing (DTrace) facility that you can use to examine the behavior of the operating system and the operating system kernel. Version 0.4 of DTrace is described, which is supported for use with UEK R3.

13.1 About DTrace

DTrace is a comprehensive dynamic tracing facility that was first developed for the Oracle Solaris operating system, and subsequently ported to Oracle Linux. DTrace allows you to explore your system to understand how it works, to track down performance problems across many layers of software, or to locate the causes of aberrant behavior.

Using DTrace, you can record data at locations of interest in the kernel, called probes. A probe is a location to which DTrace can bind a request to perform a set of actions, such as recording a stack trace, a timestamp, or the argument to a function. Probes function like programmable sensors that record information. When a probe is triggered, DTrace gathers data from it and reports the data back to you.

Using DTrace’s D programming language, you can query the system probes to provide immediate, concise answers to arbitrary questions that you formulate.

13.2 Installing and Configuring DTrace

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The DTrace dtrace-utils package is available from ULN. Your system must be registered with ULN and be installed with or be updated to Oracle Linux 6 Update 4 or later.</td>
</tr>
</tbody>
</table>
Installing and Configuring DTrace

To install and configure DTrace, perform the following steps:

1. On ULN, subscribe your system to the following channels:
   
   - Oracle Linux 6 Latest (x86_64) \( (\text{ol6\_x86\_64\_latest}) \)
   - Unbreakable Enterprise Kernel Release 3 for Oracle Linux 6 (x86_64) - Latest  
     \( (\text{ol6\_x86\_64\_UEKR3\_latest}) \)
   - Oracle Linux 6 Dtrace Userspace Tools (x86_64) - Latest  
     \( (\text{ol6\_x86\_64\_Dtrace\_userspace\_latest}) \)

   Make sure that your system is \textit{not} subscribed to the following channels:

   - Latest Unbreakable Enterprise Kernel for Oracle Linux 6 (x86_64) \( (\text{ol6\_x86\_64\_UEK\_latest}) \)
   - Dtrace for Oracle Linux 6 (x86_64) - Latest \( (\text{ol6\_x86\_64\_Dtrace\_latest}) \)
   - Dtrace for Oracle Linux 6 (x86_64) - Beta release \( (\text{ol6\_x86\_64\_Dtrace\_BETA}) \)
   - Unbreakable Enterprise Kernel Release 3 (3.8 based) for Oracle Linux 6 (x86_64) - Beta release  
     \( (\text{ol6\_x86\_64\_UEK\_BETA}) \)

   These channels are applicable to UEK R2, DTrace for UEK R2, the beta release of DTrace for UEK R2,  
   and the beta release of UEK R3.

2. If your system is not already running the latest version of the Unbreakable Enterprise Kernel Release 3  
   (UEK R3):
   
   a. Use \texttt{yum} to update your system to use UEK R3:

   
   \begin{verbatim}
   # yum update
   \end{verbatim}

   b. Reboot the system, selecting the Oracle Linux Server (3.8.13) kernel in the GRUB menu if it is not  
      the default kernel.

3. Use \texttt{yum} to install the DTrace utilities package:

   \begin{verbatim}
   # yum install dtrace-utils
   \end{verbatim}

   If you subsequently use \texttt{yum update} to install a new kernel, \texttt{yum} does not automatically install  
   the matching \texttt{dtrace-modules} package that the kernel requires. If the appropriate \texttt{dtrace-modules}  
   package for the running kernel is not present on the system, the \texttt{dtrace} command downloads and installs  
   the package from ULN. To invoke this action without performing a trace, use a command such as the  
   following:

   \begin{verbatim}
   # dtrace -l
   \end{verbatim}

   Alternatively, run the following command to install the DTrace module that is appropriate to the running  
   kernel:

   \begin{verbatim}
   # yum install dtrace-modules-`uname -r`
   \end{verbatim}

   If you want to implement a \texttt{libdtrace} consumer or develop a DTrace provider, use \texttt{yum} to install the  
   \texttt{dtrace-utils-devel} or \texttt{dtrace-modules-provider-headers} package respectively.

   To be able to trace user-space processes that are run by users other than \texttt{root}, change the mode of the  
   DTrace helper device as described in Section 13.2.1, “Changing the Mode of the DTrace Helper Device”. 

160
13.2.1 Changing the Mode of the DTrace Helper Device

The DTrace helper device (/dev/dtrace/helper) allows a user-space application that contains DTrace probes to send probe provider information to DTrace.

To trace user-space processes that are run by users other than root, you must change the mode of the DTrace helper device to allow the user to record tracing information, for example:

```bash
# chmod 666 /dev/dtrace/helper
```

Alternatively, if the acl package is installed on your system, you can use an ACL rule to limit access to a specific user, for example:

```bash
# setfacl -m u:guest:rw /dev/dtrace/helper
```

**Note**

You must change the mode on the device before the user runs the program.

You can create a udev rules file such as /etc/udev/rules.d/10-dtrace.rules to change the permissions on the device file when the system starts.

To change the mode of the device file, the udev rules file should contain the following line:

```bash
kernel="dtrace/helper", MODE="0666"
```

To change the ACL settings for the device file, use a line such as the following in the udev rules file:

```bash
kernel="dtrace/helper", RUN="/usr/bin/setfacl -m u:guest:rw /dev/dtrace/helper"
```

To apply the udev rule without needing to restart the system, run the start_udev command.

13.2.2 Loading DTrace Kernel Modules

Use the modprobe command to load the modules that support the DTrace probes that you want to use. For example, if you wanted to use the probes that the proc provider publishes, you would load the sdt module.

```bash
# modprobe sdt
```

**Note**

The fasttrap, profile, sdt, and systrace modules automatically load the dtrace module, and the dtrace module automatically loads the ctf module.

To list the probes that a specific provider publishes, use the following command:

```bash
# dtrace -l -P provider
```

To verify that a probe is available:

```bash
# dtrace -l -n probe_name
```

To display the probes that are available for a specific module:

```bash
# dtrace -l -m module_name
```
For example, display the probes that are provided by the `libphp5.so` and `mysqld` modules for DTrace-enabled PHP and MySQL:

```
# dtrace -l -m libphp5.so -m mysqld
```

<table>
<thead>
<tr>
<th>ID</th>
<th>PROVIDER</th>
<th>MODULE</th>
<th>FUNCTION NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>dtrace_compile_file compile-file-entry</td>
</tr>
<tr>
<td>5</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>dtrace_compile_file compile-file-return</td>
</tr>
<tr>
<td>6</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>zend_error error</td>
</tr>
<tr>
<td>7</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>ZEND_CATCH_SPEC_CONST_CV_HANDLER exception-caught</td>
</tr>
<tr>
<td>8</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>zend_throw_exception_internal exception-thrown</td>
</tr>
<tr>
<td>9</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>dtrace_execute_ex execute-entry</td>
</tr>
<tr>
<td>10</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>dtrace_execute_internal execute-entry</td>
</tr>
<tr>
<td>11</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>dtrace_execute_ex execute-return</td>
</tr>
<tr>
<td>12</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>dtrace_execute_internal execute-return</td>
</tr>
<tr>
<td>13</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>dtraceExecuteEx function-entry</td>
</tr>
<tr>
<td>14</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>dtrace_execute_ex function-return</td>
</tr>
<tr>
<td>15</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>php_request_shutdown request-shutdown</td>
</tr>
<tr>
<td>16</td>
<td>php3566</td>
<td>libphp5.so</td>
<td>php_request_startup request-startup</td>
</tr>
<tr>
<td>121</td>
<td>mysql3684</td>
<td>mysqld</td>
<td>_Z16dispatch_command19enum_server_commandP3THDPcj command-done</td>
</tr>
<tr>
<td>122</td>
<td>mysql3684</td>
<td>mysqld</td>
<td>_Z16dispatch_command19enum_server_commandP3THDPcj command-start</td>
</tr>
<tr>
<td>123</td>
<td>mysql3684</td>
<td>mysqld</td>
<td>_Z16close_connectionP3THDj connection-done</td>
</tr>
<tr>
<td>124</td>
<td>mysql3684</td>
<td>mysqld</td>
<td>_Z22thd_prepare_connectionP3THD connection-start</td>
</tr>
<tr>
<td>125</td>
<td>mysql3684</td>
<td>mysqld</td>
<td>_ZN7handler13ha_delete_rowEPKh delete-row-done</td>
</tr>
<tr>
<td>126</td>
<td>mysql3684</td>
<td>mysqld</td>
<td>_ZN7handler13ha_delete_rowEPKh delete-row-start</td>
</tr>
<tr>
<td>127</td>
<td>mysql3684</td>
<td>mysqld</td>
<td>_ZN7handler13ha_delete_rowEPKh delete-row-start</td>
</tr>
<tr>
<td>128</td>
<td>mysql3684</td>
<td>mysqld</td>
<td>_Z21mysql_execute_commandP3THD delete-start</td>
</tr>
<tr>
<td>129</td>
<td>mysql3684</td>
<td>mysqld</td>
<td><em>Z28filesortP3THDP5TABLEP8FilesortbPyS5</em> filesort-done</td>
</tr>
<tr>
<td>130</td>
<td>mysql3684</td>
<td>mysqld</td>
<td><em>Z28filesortP3THDP5TABLEP8FilesortbPyS5</em> filesort-start</td>
</tr>
</tbody>
</table>

**Note**

For DTrace-enabled, user-space programs, this command requires the `fasttrap` module to have been loaded before the program was started, and it does not return any probes if no instance of the program is running. `dtrace` appends the PID of the process to the DTrace provider name that was defined for the program when it was built.

### 13.3 Differences Between DTrace on Oracle Linux and Oracle Solaris

Note the following main differences that exist in the implementation of DTrace on Oracle Linux relative to Oracle Solaris.

- The following providers are available in the Oracle Linux implementation of DTrace.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Kernel Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtrace</td>
<td>dtrace</td>
<td>Provides probes that relate to DTrace itself, such as <code>BEGIN</code>, <code>ERROR</code>, and <code>END</code>. You can use these probes to initialize DTrace's state before tracing begins, process its state after tracing has completed, and handle unexpected execution errors in other probes.</td>
</tr>
<tr>
<td>fasttrap</td>
<td>fasttrap</td>
<td>Supports user-space tracing of DTrace-enabled applications.</td>
</tr>
<tr>
<td>io</td>
<td>sdt</td>
<td>Provides probes that relate to data input and output. The <code>io</code> provider enables quick exploration of behavior observed through I/O monitoring tools such as <code>iostat</code>.</td>
</tr>
</tbody>
</table>
### Calling DTrace from the Command Line

The `dtrace` command accepts the following options:

```
  dtrace [-CeFGhHlqSvVwZ] [-b bufsize] [-c command] [-D name[=value]] [-I path] [-L ...]
  [-o path] [-p PID] [-s source_path] [-U name] [-x option[=value]][-Xa|c|s|t]]
  [-m [[provider:]module[=value]] action]  
  [-f [[provider:]module:]function[=value]]
```

<table>
<thead>
<tr>
<th>Provider</th>
<th>Kernel Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc</td>
<td>sdt</td>
<td>Provides probes for monitoring process creation and termination, LWP creation and termination, execution of new programs, and signal handling.</td>
</tr>
<tr>
<td>profile</td>
<td>profile</td>
<td>Provides probes associated with an interrupt that fires at a fixed, specified time interval. These probes are associated with the asynchronous interrupt event rather than with any particular point of execution. You can use these probes to sample some aspect of a system's state.</td>
</tr>
<tr>
<td>sched</td>
<td>sdt</td>
<td>Provides probes related to CPU scheduling. Because CPUs are the one resource that all threads must consume, the <code>sched</code> provider is very useful for understanding systemic behavior.</td>
</tr>
<tr>
<td>syscall</td>
<td>systrace</td>
<td>Provides probes at the entry to and return from every system call. Because system calls are the primary interface between user-level applications and the operating system kernel, these probes can offer you an insight into the interaction between applications and the system.</td>
</tr>
</tbody>
</table>

Other providers, such as the `pid` provider, the Function Boundary Tracing (`fbt`) provider, and the providers for the network protocols (`ip`, `iscsi`, `nfsv3`, `nfsv4`, `srp`, `tcp`, and `udp`), have not yet been implemented.

- Solaris-specific features such as projects, zones, tasks, contracts, and message queues are not supported.
- The names of kernel probes are specific to the Linux kernel.
- The `-Xa`, `-Xc`, and `-Xt` options to `dtrace` all include the option `-std=gnu99` (conformance with 1999 C standard including GNU extensions) when invoking the C preprocessor (`cpp`) on D programs. The `-Xs` option includes the option `-traditional-cpp` (conformance with K&R C).
- Anonymous tracing is not supported (`-a` and `-A` options to `dtrace`).
- The 32-bit data model is not supported (`-32` option to `dtrace`).
- Various definitions in the `<dtrace.h>` header file for flags, types, structures, and function prototypes reflect intrinsic differences between the implementation of Oracle Solaris and Oracle Linux.
- SDT probes do not work in IRQ context. As a result, the `proc:::signal-discard` probe does not fire if a signal that is sent as event notification for a POSIX timer expiration should be discarded.

See the `INCOMPATIBILITIES` file in `/usr/share/doc/dtrace-DTrace_version` for more information.

## 13.4 Calling DTrace from the Command Line

The `dtrace` command accepts the following options:
Calling DTrace from the Command Line

[-n [[provider:]module:]function:name[[predicate]action]]
[-i probe-id[[predicate]action]]

where predicate is any D predicate enclosed in slashes // and action is any D statement list enclosed in braces () according to the D language syntax. If D program code is provided as an argument to the -P, -m, -f, -n, or -i options, this text must be appropriately quoted to avoid interpretation by the shell.

The options are as follows:

- **-b bufsize**
  Set the principal trace buffer size, which can include any of the size suffixes k (kilobyte), m (megabyte), g (gigabyte), or t (terabyte). If the buffer space cannot be allocated, dtrace attempts to reduce the buffer size or exit depending on the setting of the bufresize property.

- **-c command**
  Run the specified command and exit upon its completion. If you specify more than one -c option, dtrace exits when all the commands have exited, and reports the exit status for each child process as it terminates. dtrace makes the process ID of the first command available to D programs as the $target macro variable.

- **-C**
  Run the C preprocessor (cpp) on D programs before compiling them. You can pass options to the C preprocessor by using the -D, -H, -I, and -U options. You can use the -X option to select the degree of conformance with the C standard.

- **-D name[=value]**
  Define the specified macro name and optional value when invoking cpp using the -C option. You can specify the -D option multiple times to the command.

- **-e**
  Exit after compiling any requests and before enabling any probes. You can combine this option with the -D option to verify that your D programs compile without executing them or enabling the corresponding instrumentation.

- **-f [[provider:]module:]function[\'D-probe_clause\']**
  Specify a function (optionally specifying the provider and module) that you want to trace or list. You can append an optional D-probe clause. You can specify the -f option multiple times to the command.

- **-F**
  Reduce trace output by combining the output for function and system call entry and return points. dtrace indents entry probe reports and leaves return probe reports unindented. dtrace prefixes the output from function entry probe reports with -> and the output from function return probe reports with <-. dtrace prefixes the output from system call entry probe reports with => and the output from system call return probe reports with <=.

- **-G**
  Generate an ELF file that contains an embedded D program. dtrace saves the DTrace probes that are specified in the program using a relocatable ELF object that can be linked with another program. If you specify the -o option, dtrace saves the ELF file to the specified path name. If you do not specify the -o option, the ELF file is given the same name as the source file for the D program, except with a .o extension instead of .s. Otherwise, the ELF file is saved with the name d.out.
-h

Create a header file based on probe definitions in the file that is specified as the argument to the `-s` option. If you specify the `-o` option, `dtrace` saves the header file to the specified path name. If you do not specify the `-o` option, the header file is given the same name as the source file for the D program, except with a `.h` extension instead of `.d`. You should amend the source file of the program to be traced so that it includes this header file.

-H

Print the path names of included files on `stderr` when you invoke `cpp` using the `-C` option.

-i probe_ID[\`D-probe_clause\`]

Specify a probe identifier that you want to trace or list. You must specify the probe ID as a decimal integer (as displayed by `dtrace -l`). You can append an optional D-probe clause. You can specify the `-i` option multiple times to the command.

-I pathname

Add the specified directory path to the search path for `#include` files when you invoke `cpp` using the `-C` option. The specified directory is inserted at the head of the default directory list.

-l

List probes instead of enabling them. `dtrace` filters the list of probes based on the arguments to the `-f`, `-i`, `-m`, `-n`, `-P`, and `-s` options. If no options are specified, `dtrace` lists all probes.

-L pathname

Add the specified directory path to the end of the library search path. Use this option to specify the path to DTrace libraries, which contain common definitions for D programs.

-m [provider:]module[\`D-probe_clause\`]

Specify a module (optionally specifying the provider) that you want to trace or list. You can append an optional D-probe clause. You can specify the `-m` option multiple times to the command.

-n [\[[provider:]\] \[module:]\] \[function\]]prob[\`D-probe_clause\`]

Specify a probe name (optionally specifying the provider, module, and function) that you want to trace or list. You can append an optional D-probe clause. You can specify the `-n` option multiple times to the command.

-o pathname

Specify the output file for the `-G` and `-l` options, or for traced data.

-p PID

Grab a process specified by its process ID, cache its symbol tables, and exit upon its completion. If you specify more than one `-p` option, `dtrace` exits when all the processes have exited, and reports the exit status for each process as it terminates. `dtrace` makes the first process ID specified available to D programs as the `$target` macro variable.

-P provider[\`D-probe_clause\`]

Specify a provider that you want to trace or list. You can append an optional D-probe clause. You can specify the `-P` option multiple times to the command.

-q

Set quiet mode. `dtrace` suppresses informational messages, column headers, the CPU ID, the probe ID, and additional newlines. Only data that is traced and formatted by the `printa()`, `printf()`, and `trace()` D program statements is displayed on `stdout`. This option is equivalent to specifying `#pragma D option quiet` in a D program.
### 13.5 About Programming for DTrace

When you use the `dtrace` command, you invoke the compiler for the D language. Once DTrace has compiled your program, it sends it to the operating system kernel for execution, where it activates the probes that your program uses.

DTrace enables probes only when you are using them. No instrumented code is present for inactive probes, so your system does not experience performance degradation when you are not using DTrace. Once your D program exits, all of the probes it used are automatically disabled and their instrumentation is

---

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-s source_pathname</code></td>
<td>Specifies a D program source file to be compiled by <code>dtrace</code>. If you specify the <code>-h</code> option, <code>dtrace</code> creates a header file using the probe definitions in the file. If you specify the <code>-G</code> option, <code>dtrace</code> generates a relocatable ELF object that can be linked with another program. If you specify the <code>-e</code> option, <code>dtrace</code> compiles the program, but it does not enable any instrumentation. If you specify the <code>-l</code> option, <code>dtrace</code> compiles the program and lists the set of matching probes, but it does not enable any instrumentation. If you do not specify an option, <code>dtrace</code> enables the instrumentation specified by the D program and begins tracing.</td>
</tr>
<tr>
<td><code>-S</code></td>
<td>Show the D compiler intermediate code. The D compiler writes a report of the intermediate code that it generated for each D program to <code>stderr</code>.</td>
</tr>
<tr>
<td><code>-U name</code></td>
<td>Undefine the specified name when invoking <code>cpp</code> using the <code>-C</code> option. You can specify the <code>-U</code> option multiple times to the command.</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>Set verbose mode. <code>dtrace</code> produces a program stability report showing the minimum interface stability and dependency level for any specified D programs.</td>
</tr>
<tr>
<td><code>-V</code></td>
<td>Write the highest D programming interface version supported by <code>dtrace</code> to <code>stdout</code>.</td>
</tr>
<tr>
<td><code>-w</code></td>
<td>Permit destructive actions by D programs. If you do not specify this option, <code>dtrace</code> does not compile or enable a D program that contains destructive actions. This option is equivalent to specifying <code>#pragma D option destructive</code> in a D program.</td>
</tr>
<tr>
<td><code>-x option[=value]</code></td>
<td>Enable or modify a DTrace runtime option or D compiler option.</td>
</tr>
<tr>
<td>`-X[a</td>
<td>c</td>
</tr>
<tr>
<td><code>-xs</code></td>
<td>Include the option <code>-traditional-cpp</code> (conformance with K&amp;R C) when invoking <code>cpp</code> using the <code>-C</code> option.</td>
</tr>
<tr>
<td><code>-Z</code></td>
<td>Permit probe descriptions that do not match any probes. If you do not specify this option, <code>dtrace</code> reports an error and exits if a probe description does not match a known probe.</td>
</tr>
</tbody>
</table>
removed, returning your system to its original state. No effective difference exists between a system where DTrace is not active and one where the DTrace software is not installed.

DTrace implements the instrumentation for each probe dynamically on the live, running operating system. DTrace neither quiesces nor pauses the system in any way, and it adds instrumentation code only for the probes that you enable. As a result, the effect of using DTrace probes is limited to exactly what you ask DTrace to do. DTrace instrumentation is designed to be as efficient as possible, and enables you to use it in production to solve real problems in real time.

The DTrace framework provides support for an arbitrary number of virtual clients. You can run as many simultaneous D programs as you like, limited only by your system's memory capacity, and all the programs operate independently using the same underlying instrumentation. This same capability also permits any number of distinct users on the system to take advantage of DTrace simultaneously on the same system without interfering with one another.

Unlike a C or C++ program, but similar to a Java program, DTrace compiles your D program into a safe intermediate form that it executes when a probe fires. DTrace validates whether this intermediate form can run safely, reporting any run-time errors that might occur during the execution of your D program, such as dividing by zero or dereferencing invalid memory. As a result, you cannot construct an unsafe D program. You can use DTrace in a production environment without worrying about crashing or corrupting your system. If you make a programming mistake, DTrace disables the instrumentation and reports the error to you.

Figure 13.1 illustrates the different components of the DTrace architecture, including probe providers, the DTrace driver, the DTrace library, and the `dtrace` command.

**Figure 13.1 Components of the DTrace Architecture**

![Diagram of DTrace Architecture]

13.6 Introducing the D Programming Language

D programs describe the probes that are to be enabled together with the predicates and actions that are bound to the probes. D programs can also declare variables and define new types. This section provides an introduction to the important features that you are likely to encounter in simple D programs.
13.6.1 Probe Clauses

D programs consist of a set of one or more probe clauses. Each probe clause takes the general form shown here:

```
probe_description_1 [, probe_description_2]...
[ // predicate_statement /]
{ [ action_statement; ] . . . }
```

Every probe clause begins with a list of one or more probe descriptions in this form:

```
provider:module:function:probe_name
```

where the fields are as follows:

- **provider** The name of the DTrace provider that is publishing this probe. For kernel probes, the provider name typically corresponds to the name of the DTrace kernel module that performs the instrumentation to enable the probe, for example, proc. When tracing a DTrace-enabled, user-space application or library, this field takes the form `namePID`, where `name` is the name of the provider as defined in the provider definition file that was used to build the application or library and `PID` is the process ID of the running executable.

- **module** The name of the kernel module, library, or user-space program in which the probe is located, if any, for example, vmlinux. This module is not the same as the kernel module that implements a provider.

- **function** The name of the function in which the probe is located, for example, do_fork.

- **probe_name** The name of the probe usually describes its location within a function, for example, create, entry, or return.

The compiler interprets the fields from right to left. For example, the probe description `settimeofday:entry` would match a probe with function `settimeofday` and name `entry` regardless of the value of the probe's provider and module fields. You can regard a probe description as a pattern that matches one or more probes based on their names. You can omit the leading colons before a probe name if the probe that you want to use has a unique name. If several providers publish probes with the same name, use the available fields to obtain the correct probe. If you do not specify a provider, you might obtain unexpected results if multiple probes have the same name. Specifying a provider but leaving the module, function, and probe name fields blank, matches all probes in a provider. For example, `syscall:::` matches every probe published by the syscall provider.

The optional predicate statement uses criteria such as process ID, command name, or timestamp to determine whether the associated actions should take place. If you omit the predicate, any associated actions always run if the probe is triggered.

You can use the ?, *, and [] shell wildcards with probe clauses. For example, `syscall::[gs]et*:` matches all syscall probes for function names that begin with get or set. If necessary, use the \ character to escape wildcard characters that form part of a name.

You can enable the same actions for more than one probe description. For example, the following D program uses the `trace()` function to record a timestamp each time that any process invokes a system call containing the string `mem` or `soc`:

```
syscall:*mem*:entry, syscall:*soc*:entry
```
By default, the `trace()` function writes the result to the principal buffer, which is accessible by other probe clauses within a D program, and whose contents dtrace displays when the program exits.

### 13.6.2 Pragmas

You can use compiler directives called pragmas in a D program. Pragma lines begin with a `#` character, and are usually placed at the beginning of a D program. The primary use of pragmas is to set run-time DTrace options. For example, the following pragma statements suppress all output except for traced data and permit destructive operations.

```d
#pragma D option quiet
#pragma D option destructive
```

### 13.6.3 Global Variables

D provides fundamental data types for integers and floating-point constants. You can perform arithmetic only on integers in D programs. D does not support floating-point operations. D provides floating-point types for compatibility with ANSI-C declarations and types. You can trace floating-point data objects and use the `printf()` function to format them for output. In the current implementation, DTrace supports only the 64-bit data model for writing D programs.

You can use declarations to introduce D variables and external C symbols, or to define new types for use in D. The following example program, `tick.d`, declares and initializes the variable `i` when the D program starts, displays its initial value, increments the variable and prints its value once every second, and displays the final value when the program exits.

```d
BEGIN
{
    i = 0;
    trace(i);
}
profile:::tick-1sec
{
    printf("i=%d\n",++i);
}
END
{
    trace(i);
}
```

When run, the program produces output such as the following until you type `Ctrl-C`:

```d
# dtrace -s tick.d
dtrace: script 'tick.d' matched 3 probes
CPU     ID               FUNCTION:NAME
  1      1                       :BEGIN       0
  1    618                       :tick-1sec i=1
  1    618                       :tick-1sec i=2
  1    618                       :tick-1sec i=3
  1    618                       :tick-1sec i=4
  1    618                       :tick-1sec i=5
```
Whenever a probe is triggered, \texttt{dtrace} displays the number of the CPU core on which the process indicated by its ID is running, and the name of the function and the probe. \texttt{BEGIN} and \texttt{END} are DTrace probes that trigger when the \texttt{dtrace} program starts and finishes.

To suppress all output except that from \texttt{printa()}, \texttt{printf()}, and \texttt{trace()}, specify \texttt{#pragma D option quiet} in the program or the \texttt{-q} option to \texttt{dtrace}.

```
# dtrace -q -s tick.d
0i=1
  i=2
  i=3
  i=4
  i=5
^C
5
```

### 13.6.4 Predicates

Predicates are logic statements that select whether DTrace invokes the actions that are associated with a probe. For example, the predicates in the following program \texttt{sc1000.d} examine the value of the variable \texttt{i}. This program also demonstrates how to include C-style comments.

```
#pragma D option quiet
BEGIN
{ /* Initialize i */
  i = 1000;
}
syscall:::entry
 /i > 0/  
 { /* Decrement i */
   i--;
  }
syscall:::entry
 /%(i \% 100) == 0/  
 { /* Print i after every 100 system calls */
   printf("i = \%d\n",i);
  }
syscall:::entry
 /i == 0/  
 { printf("i = 0; 1000 system calls invoked\n");
   exit(0); /* Exit with a value of 0 */
  }
```

The program initializes \texttt{i} with a value of 1000, decrements its value by 1 whenever a process invokes a system call, prints its value after every 100 system calls, and exits when the value of \texttt{i} reaches 0. Running the program in quite mode produces output similar to the following:

```
# dtrace -s sc1000.d
i = 900
i = 700
i = 800
i = 600
i = 500
```

Scalar Arrays and Associative Arrays

Note that the order of the countdown sequence is not as expected. The output for \texttt{i=800} appears after the output for \texttt{i=700}. If you turn off quiet mode, it becomes apparent that the reason is that \texttt{dtrace} is collecting information from probes that can be triggered on all the CPU cores. You cannot expect runtime output from DTrace to be sequential in a multithreaded environment.

```bash
# dtrace -s sc1000.d
Dtrace: script 'sc1000.d' matched 889 probes

CPU     ID                FUNCTION:NAME
0    457           clock_gettime:entry i = 900
0    413                   futex:entry i = 700
1     41                   lseek:entry i = 800
1     25                    read:entry i = 600
1     25                    read:entry i = 500
1     25                    read:entry i = 400
1     71                  select:entry i = 300
1     71                  select:entry i = 200
1     25                    read:entry i = 100
1     25                    read:entry i = 0
1     25                    read:entry i = 0; 1000 system calls invoked
```

The next example is an executable DTrace script that displays the file descriptor, output string, and string length specified to the \texttt{write()} system call whenever the \texttt{date} command is run on the system.

```bash
#!/usr/sbin/dtrace -s
#pragma D option quiet
syscall::write:entry
/execname == "date"/
{
    printf("%s(%d, %s, %4d)\n", probefunc, arg0, copyinstr(arg1), arg2);
}
```

If you run the script from one window, while typing the \texttt{date} command in another, you see output such as the following in the first window:

```
write(1, Wed Aug 15 10:42:34 BST 2012 , 29)
```

13.6.5 Scalar Arrays and Associative Arrays

The D language supports \textit{scalar arrays}, which correspond directly in concept and syntax with arrays in C. A scalar array is a fixed-length group of consecutive memory locations that each store a value of the same type. You access scalar arrays by referring to each location with an integer starting from zero. In D programs, you would usually use scalar arrays to access array data within the operating system.

For example, you would use the following statement to declare a scalar array \texttt{sa} of 5 integers:
int sa[5];

As in C, `sa[0]` refers to the first array element, `sa[1]` refers to the second, and so on up to `sa[4]` for the fifth element.

The D language also supports a special kind of variable called an **associative array**. An associative array is similar to a scalar array in that it associates a set of keys with a set of values, but in an associative array the keys are not limited to integers of a fixed range. In the D language, you can index associative arrays by a list of one or more values of any type. Together the individual key values form a **tuple** that you use to index into the array and access or modify the value that corresponds to that key. Each tuple key must be of the same length and must have the same key types in the same order. The value associated with each element of an associative array is also of a single fixed type for the entire array.

For example, the following statement defines a new associative array `aa` of value type `int` with the tuple signature `string, int`, and stores the integer value 828 in the array:

```
aa["foo", 271] = 828;
```

Once you have defined an array, you can access its elements in the same way as any other variable. For example, the following statement modifies the array element previously stored in `a` by incrementing the value from 828 to 829:

```
a["foo", 271]++;
```

You can define additional elements for the array by specifying a different tuple with the same tuple signature, as shown here:

```
aa["bar", 314] = 159;
aa["foo", 577] = 216;
```

The array elements `aa["foo", 271]` and `aa["foo", 577]` are distinct because the values of their tuples differ in the value of their second key.

Syntactically, scalar arrays and associative arrays are very similar. You can declare an associative array of integers referenced by an integer key as follows:

```
int ai[int];
```

You could reference an element of this array using the expression such as `ai[0]`. However, from a storage and implementation perspective, the two kinds of array are very different. The scalar array `sa` consists of five consecutive memory locations numbered from zero, and the index refers to an offset in the storage allocated for the array. An associative array such as `ai` has no predefined size and it does not store elements in consecutive memory locations. In addition, associative array keys have no relationship to the storage location of the corresponding value. If you access the associative array elements `a[0]` and `a[-5]`, DTrace allocates only two words of storage, which are not necessarily consecutive in memory. The tuple keys that you use to index associative arrays are abstract names for the corresponding value, and they bear no relationship to the location of the value in memory.

If you create an array using an initial assignment and use a single integer expression as the array index, for example, `a[0] = 2;`, the D compiler always creates a new associative array, even though `a` could also be interpreted as an assignment to a scalar array. If you want to use a scalar array, you must explicitly declare its type and size.

### 13.6.6 Pointers and External Variables

The implementation of pointers in the D language gives you the ability to create and manipulate the memory addresses of data objects in the operating system kernel, and to store the contents of those data
Address Spaces

objects in variables and associative arrays. The syntax of D pointers is the same as the syntax of pointers in ANSI-C. For example, the following statement declares a D global variable named \( p \) that is a pointer to an integer.

\[
\text{int } *p;
\]

This declaration means that \( p \) itself is a 64-bit integer whose value is the address in memory of another integer.

If you want to create a pointer to a data object inside the kernel, you can compute its address by using the & reference operator. For example, the kernel source code declares an `unsigned long max_pfn` variable. You can access the value of such an external variable in the D language by prefixing it with the `(backquote) scope operator:

\[
\text{value} = `\text{max_pfn};
\]

If more than one kernel module declares a variable with the same name, prefix the scoped external variable with the name of the module. For example, `foo`bar` would refer to the address of the `bar()` function provided by the module `foo`.

You can extract the address of an external variable by applying the & operator and store it as a pointer:

\[
p = &`\text{max_pfn};
\]

You can use the * dereference operator to refer to the object that a pointer addresses:

\[
\text{value} = *p;
\]

You cannot apply the & operator to DTrace objects such as associative arrays, built-in functions, and variables. If you create composite structures, it is possible to construct expressions that retrieve the kernel addresses of DTrace objects. However, DTrace does not guarantee to preserve the addresses of such objects across probe firings.

You cannot use the * dereference operator on the left-hand side of an assignment expression. You may only assign values directly to D variables by name or by applying the array index operator \([]\) to a scalar array or an associative array.

You cannot use pointers to perform indirect function calls. You may only call DTrace functions directly by name.

13.6.7 Address Spaces

DTrace executes D programs within the address space of the operating system kernel. Your entire Oracle Linux system manages one address space for the operating system kernel, and one for each user process. As each address space provides the illusion that it can access all of the memory on the system, the same virtual address might be used in different address spaces, but it would translate to different locations in physical memory. If your D programs use pointers, you need to be aware which address space corresponds to those pointers.

For example, if you use the `syscall` provider to instrument entry to a system call such as `pipe()` that takes a pointer to an integer or to an array of integers as an argument, it is not valid to use the * or \[] operators to dereference that pointer or array. The address is in the address space of the user process that performed the system call, and not in the address space of the kernel. Dereferencing the address in D accesses the kernel's address space, which would result in an invalid address error or return unexpected data to your D program.

To access user process memory from a DTrace probe, use one of the `copyin()`, `copyinstr()`, or `copyinto()` functions with an address in user space.
Thread-local Variables

The following D programs show two alternate and equivalent ways to print the file descriptor, string, and string length arguments that a process passed to the `write()` system call:

```d
syscall::write:entry
{
    printf("fd=%d buf=%s count=%d", arg0, stringof(copyin(arg1, arg2)), arg2);
}
syscall::write:entry
{
    printf("fd=%d buf=%s count=%d", arg0, copyinstr(arg1, arg2), arg2);
}
```

The `arg0`, `arg1`, and `arg2` variables contain the value of the `fd`, `buf`, and `count` arguments to the system call. Note that the value of `arg1` is an address in the address space of the process, and not in the address space of the kernel.

In this example, it is necessary to use the `stringof()` function with `copyin()` so that DTrace converts the retrieved user data to a string. The `copyinstr()` function always returns a string.

To avoid confusion, you should name and comment variables that store user addresses appropriately. You should also store user addresses as variables of type `uintptr_t` so that you do not accidentally compile D code that dereferences them.

### 13.6.8 Thread-local Variables

Thread-local variables are defined within the scope of execution of a thread on the system. To indicate that a variable is thread-local, you prefix it with `self->` as shown in the following example.

```d
#pragma D option quiet
syscall::read:entry
{
    self->t = timestamp; /* Initialize a thread-local variable */
}
syscall::read:return
/\self->t != 0/ 
{
    printf("%s (pid:tid=%d:%d) spent %d microseconds in read()\n", 
        execname, pid, tid, ((timestamp - self->t)/1000)); /* Divide by 1000 -> microseconds */
    self->t = 0; /* Reset the variable */
}
```

This D program (`dtrace.d`) displays the command name, process ID, thread ID, and expired time in microseconds whenever a process invokes the `read()` system call.

```d
# dtrace -s readtrace.d
nome-terminal (pid:tid=2774:2774) spent 27 microseconds in read() 
gnome-terminal (pid:tid=2774:2774) spent 16 microseconds in read() 
hald-addon-inpu (pid:tid=1662:1662) spent 26 microseconds in read() 
hald-addon-inpu (pid:tid=1662:1662) spent 17 microseconds in read() 
Xorg (pid:tid=2046:2046) spent 18 microseconds in read() 
...
```

### 13.6.9 Speculations

The speculative tracing facility in DTrace allows you to tentatively trace data and then later decide whether to commit the data to a tracing buffer or discard the data. Predicates are the primary mechanism for filtering out uninteresting events. Predicates are useful when you know at the time that a probe fires
whether or not the probe event is of interest. However, in some situations, you might not know whether a probe event is of interest until after the probe fires.

For example, if a system call is occasionally failing with an error code in `errno`, you might want to examine the code path leading to the error condition. You can write trace data at one or more probe locations to speculative buffers, and then choose which data to commit to the principal buffer at another probe location. As a result, your trace data contains only the output of interest, no post-processing is required, and the DTrace overhead is minimized.

To create a speculative buffer, use the `speculation()` function. This function returns a speculation identifier, which you use in subsequent calls to the `speculate()` function.

Call the `speculate()` function before performing any data-recording actions in a clause. DTrace directs all subsequent data that you record in a clause to the speculative buffer. You can create only one speculation in any given clause.

Typically, you assign a speculation identifier to a thread-local variable, and then use that variable as a predicate to other probes as well as an argument to `speculate()`. For example:

```c
#!/usr/sbin/dtrace -Fs
syscall::open:entry
{
  /*
   * The call to speculation() creates a new speculation. If this fails,
   * dtrace will generate an error message indicating the reason for
   * the failed speculation(), but subsequent speculative tracing will be
   * silently discarded.
   */
  self->spec = speculation();
  speculate(self->spec);

  /*
   * Because this printf() follows the speculate(), it is being
   * speculatively traced; it will only appear in the data buffer if the
   * speculation is subsequently committed.
   */
  printf("%s", copyinstr(arg0));
}
syscall::open:return
/self->spec/
{
  /*
   * To balance the output with the -F option, we want to be sure that
   * every entry has a matching return. Because we speculated the
   * open entry above, we want to also speculate the open return.
   * This is also a convenient time to trace the errno value.
   */
  speculate(self->spec);
  trace(errno);
}
```

If a speculative buffer contains data that you want to retain, use the `commit()` function to copy its contents to the principal buffer. If you want to delete the contents of a speculative buffer, use the `discard()` function. The following example clauses commit or discard the speculative buffer based on the value of the `errno` variable:

```c
syscall::open:return
/self->spec && errno != 0/
{
  /*
   * If errno is non-zero, we want to commit the speculation.
   */
  commit(self->spec);
}
```
Aggregations

DTrace provides the following built-in functions for aggregating the data that individual probes gather.

<table>
<thead>
<tr>
<th>Aggregating Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>avg(scalar_expression)</code></td>
<td>Returns the arithmetic mean of the expressions that are specified as arguments.</td>
</tr>
<tr>
<td><code>count()</code></td>
<td>Returns the number of times that the function has been called.</td>
</tr>
<tr>
<td><code>lquantize(scalar_expression, lower_bound, upper_bound, step_interval)</code></td>
<td>Returns a linear frequency distribution of the expressions that are specified as arguments, scaled to the specified lower bound, upper bound, and step interval. Increments the value in the highest bucket that is smaller than the specified expression.</td>
</tr>
<tr>
<td><code>max(scalar_expression)</code></td>
<td>Returns the maximum value of the expressions that are specified as arguments.</td>
</tr>
<tr>
<td><code>min(scalar_expression)</code></td>
<td>Returns the minimum value of the expressions that are specified as arguments.</td>
</tr>
<tr>
<td><code>quantize(scalar_expression)</code></td>
<td>Returns a power-of-two frequency distribution of the expressions that are specified as arguments. Increments the value of the highest power-of-two bucket that is smaller than the specified expression.</td>
</tr>
<tr>
<td><code>stddev(scalar_expression)</code></td>
<td>Returns the standard deviation of the expressions that are specified as arguments.</td>
</tr>
<tr>
<td><code>sum(scalar_expression)</code></td>
<td>Returns the sum of the expressions that are specified as arguments.</td>
</tr>
</tbody>
</table>

DTrace indexes the results of an aggregation using a tuple expression similar to that used for an associative array:
The name of the aggregation is prefixed with an @ character. All aggregations are global. If you do not specify a name, the aggregation is anonymous. The keys describe the data that the aggregating function is collecting.

For example, the following command counts the number of write() system calls invoked by processes until you type Ctrl-C.

```
# dtrace -n syscall::write:entry '{ @["write() calls"] = count(); }'
dtrace: description 'syscall:::' matched 1 probe
^C
write() calls                                              9
```

The next example counts the number of both read() and write() system calls:

```
# dtrace -n syscall::write:entry,syscall::read:entry
  '{ @[strjoin(probefunc,"() calls")] = count(); }'
dtrace: description 'syscall::write:entry,syscall::read:entry' matched 2 probes
^C
write() calls                                            150
read() calls                                            1555
```

### Note

If you specify the -q option to dtrace or #pragma D option quiet in a D program, DTrace suppresses the automatic printing of aggregations. In this case, you must use a printa() statement to display the information.

## 13.7 DTrace Command Examples

Display the probes that are available with the proc provider.

```
# dtrace -l -P proc
ID   PROVIDER            MODULE                          FUNCTION NAME
4066       proc           vmlinux                     schedule_tail start
4067       proc           vmlinux                     schedule_tail lwp-start
4069       proc           vmlinux             get_signal_to_deliver signal-handle
4074       proc           vmlinux                   do_sigtimedwait signal-clear
4075       proc           vmlinux                           do_fork lwp-create
4076       proc           vmlinux                           do_fork create
4077       proc           vmlinux                           do_exit lwp-exit
4078       proc           vmlinux                           do_exit exit
4079       proc           vmlinux                  do_execve_common exec-failure
4080       proc           vmlinux                  do_execve_common exec
4081       proc           vmlinux                  do_execve_common exec-success
4085       proc           vmlinux           __send_signal signal-send
4086       proc           vmlinux           __send_signal signal-discard
```

Monitor the system as it loads and executes process images.

```
# dtrace -n 'proc::do_execve_common:exec { trace(stringof(arg0)); }'
dtrace: description 'proc:::exec' matched 1 probe
CPU     ID                FUNCTION:NAME
0    600        do_execve_common:exec   /bin/uname
0    600        do_execve_common:exec   /bin/mkdir
0    600        do_execve_common:exec   /bin/sed
0    600        do_execve_common:exec   /usr/bin/dirname
1    600        do_execve_common:exec   /usr/lib64/qt-3.3/bin/firefox
1    600        do_execve_common:exec   /usr/local/bin/firefox
```
DTrace Command Examples

<table>
<thead>
<tr>
<th>PID</th>
<th>Argument 1</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>/usr/bin/firefox</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/bin/basename</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/bin/uname</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/usr/bin/mozilla-plugin-config</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/usr/lib64/nspluginwrapper/plugin-config</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/usr/lib64/xulrunner-1.9.2/mozilla-xremote-client</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/bin/sed</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/usr/lib64/firefox-3.6/run-mozilla.sh</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/bin/basename</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/bin/uname</td>
<td>do_execve_common:exec</td>
</tr>
<tr>
<td>600</td>
<td>/usr/lib64/firefox-3.6/firefox</td>
<td>do_execve_common:exec</td>
</tr>
</tbody>
</table>

Display the names of commands that invoke the `open()` system call and the name of the file being opened.

```
# dtrace -q -n 'syscall::open:entry { printf("%-16s %-16s\n",execname,copyinstr(arg0)); }'
udisks-daemon           /dev/sr0
devkit-power-da          /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/present
devkit-power-da          /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/energy_now
devkit-power-da          /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/voltage_max_design
devkit-power-da          /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/voltage_min_design
devkit-power-da          /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/status
devkit-power-da          /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/current_now
devkit-power-da          /sys/devices/LNXSYSTM:00/.../PNP0C0A:00/power_supply/BAT0/voltage_now
VBoxService             /var/run/utmp
firefox                 /home/guest/.mozilla/firefox/qeaojiol.default/sessionstore.js
firefox                 /home/guest/.mozilla/firefox/qeaojiol.default/sessionstore-1.js
firefox                 /home/guest/.mozilla/firefox/qeaojiol.default/sessionstore-1.js
```

Display the system calls invoked by the process with ID 3007 and the number of times that it invoked each system call.

```
# dtrace -p 3007 -n 'syscall:::entry { @num[probefunc] = count(); }'
dtrace: description 'syscall:::entry' matched 296 probes
```

```bash
getuid                1
ptrace               1
socket               1
waitid               1
lseek                3
statfs               3
access               4
write                6
munmap              15
newfstat             16
newstat             17
mmap                19
fcntl               20
close               24
alarm               30
inotify_add_watch   30
open               32
rt_sigaction         50
nanosleep            52
rt_sigprocmask       64
ioctl               117
futex               311
clock_gettime       579
rt_sigreturn         744
gettimeofday         1461
setitimer             2093
select               2530
writev              3162
poll                4720
```
Display the distribution of the sizes specified to `read()` calls invoked by running `firefox`.

```bash
# dtrace -n 'syscall::read:entry /execname="firefox"/@dist["firefox"]=quantize(arg2);'
dtrace: description 'syscall::read:entry ' matched 1 probe
^C
```

<table>
<thead>
<tr>
<th>firefox value</th>
<th>Distribution</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>566</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>128</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>256</td>
<td>@</td>
<td>436</td>
</tr>
<tr>
<td>512</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>1024</td>
<td>@</td>
<td>959</td>
</tr>
<tr>
<td>2048</td>
<td>@</td>
<td>230</td>
</tr>
</tbody>
</table>
| 4096          | @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
13.8 Tracing User-Space Applications

A number of DTrace-enabled applications will be made available following the release of DTrace 0.4, including MySQL and PHP. These applications have been instrumented to contain statically defined DTrace probes. You can find details about the probes for MySQL at http://dev.mysql.com/doc/refman/5.5/en/dba-dtrace-mysqld-ref.html and about the probes for PHP at http://php.net/manual/features.dtrace.php.

The MySQL query-probes `query-start(query, connectionid, database, user, host)` and `query-done(status)` are triggered when the MySQL server receives a query is received by the server and when the query has been completed and the server has successfully sent the information to the client.

For example, the following script reports the execution time for each database query:

```
#!/usr/sbin/dtrace -qs

dtrace:::BEGIN
{
  printf("%-20s %-10s %-40s %-9s\n", "Who", "Database", "Query", "Time(microseconds)");
}

mysql*:::query-start
{
  self->query = copyinstr(arg0);
  self->connid = arg1;
  self->db = copyinstr(arg2);
  self->who = strjoin(copyinstr(arg3),strjoin("@",copyinstr(arg4)));
  self->querystart = timestamp;
}

mysql*:::query-done
{
  printf("%-20s %-10s %-40s %-9d\n", self->who,self->db,self->query,
    (timestamp - self->querystart) / 1000);
}
```

The following is sample output from this script:

```
Who                 Database   Query                                   Time(microseconds)
webuser@localhost   namedb     select * from table1 order by n ASC     1135
webuser@localhost   namedb     delete from table1 where n='Bill'       10383
```

The MySQL query-parsing probes `query-parse-start(query)` and `query-parse-done(status)` are triggered immediately before and after MySQL parses a SQL statement. For example, you could use the following script to monitor the execution time for parsing queries:

```
#!/usr/sbin/dtrace -qs

mysql*:::query-parse-start
{
  self->parsestart = timestamp;
  self->parsequery = copyinstr(arg0);
}

mysql*:::query-parse-done
/arg0 == 0/ 
{
  printf("Parsing %s: %d microseconds\n", self->parsequery,
    ((timestamp - self->parsestart)/1000));
}
```
Examining the Stack Trace of a User-Space Application

```c
mysql***:::query-parse-done
/arg0 != 0/
{
  printf("Error parsing %s: %d microseconds\n", self->parsequery,
    ((timestamp - self->parsestart)/1000));
}
```

The following is sample output from this script:

```bash
# ./query-parse.d
Parsing select * from table1 where n like 'B%' order by n ASC: 29 microseconds
Error parsing select from table1 join (table2) on (table1.i = table2.i)
  order by table1.s,table1.i limit 10: 36 microseconds
```

The following script uses the PHP probe `error(error_message, request_file, line_number)`, to report PHP errors:

```bash
#!/usr/sbin/dtrace -qs
php***:::error
{
  printf("PHP error\n");
  printf("  error message             %s\n", copyinstr(arg0));
  printf("  request file              %s\n", copyinstr(arg1));
  printf("  line number               %d\n\n", (int)arg2);
}
```

For example, you can use the PHP `trigger_error()` function to trigger a PHP error if a MySQL function returns an error:

```php
<?php
ini_set('error_reporting', E_ALL); /* Report all errors */
ini_set('display_errors', 'Off');  /* but do not display them */
...
  $mysqli->query($QUERY) or trigger_error($mysqli->error."[$QUERY]",E_USER_ERROR);
...
?>
```

You could use the script to report errors that might indicate incorrectly queries or attempted SQL injection attacks, for example:

```bash
# ./php_error.d
...
PHP error
  error message You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '='1'; --'' at line 1[select * from table1 where n like 'B%' or '1'='1'; --']
  request file /var/www/html/example.php
  line number 61
...
```

### 13.8.1 Examining the Stack Trace of a User-Space Application

You can use the `ustack()` function to perform a stack trace of any user-space application, for example:
DTrace can translate the stack frames into symbols for shared libraries (such as libc) and unstripped executables. As ls is a stripped executable, the addresses remain unconverted. dtrace can translate stack frames for stripped executables if the --export-dynamic option was specified when the program was linked. This option causes the linker to add all symbols to the dynamic symbol table.

**13.9 For More Information About DTrace**

For more information, see the [Oracle Linux Dynamic Tracing Guide](https://docs.oracle.com/cd/E19775-01/html/E17753/dtrace.html).
Chapter 14 Support Diagnostic Tools

Table of Contents

14.1 About sosreport ................................................................. 183
   14.1.1 Configuring and Using sosreport ........................................ 183
14.2 About Kdump ................................................................. 184
   14.2.1 Configuring and Using Kdump .......................................... 184
   14.2.2 Files Used by Kdump ..................................................... 186
14.3 About OSWatcher Black Box ............................................. 186
   14.3.1 Installing OSWbb .......................................................... 186
   14.3.2 Running OSWbb ........................................................... 187
14.4 For More Information About the Diagnostic Tools .................... 188

This chapter describes the sosreport, Kdump, and OSWbb tools that you can use to help diagnose problems with a system.

14.1 About sosreport

The sosreport utility collects information about a system such as hardware configuration, software configuration, and operational state. You can also use sosreport to enable diagnostics and analytical functions. To assist in troubleshooting a problem, sosreport records the information in a compressed file that you can send to a support representative.

14.1.1 Configuring and Using sosreport

If the sos package is not already installed on your system, use yum to install it.

Use the following command to list the available plugins and plugin options.

```
# sosreport -l
The following plugins are currently enabled:
  acpid           acpid related information
  anaconda       Anaconda / Installation information
  .              ...
The following plugins are currently disabled:
  amd            Amd automounter information
  cluster        cluster suite and GFS related information
  .              ...
The following plugin options are available:
  apache.log     off gathers all apache logs
  auditd.syslogsize  15 max size (MiB) to collect per syslog file
  .              ...
```

See the sosreport(1) manual page for information about how to enable or disable plugins, and how to set values for plugin options.

To run sosreport:
1. Enter the command, specifying any options that you need to tailor the report to report information about a problem area.

```bash
# sosreport [options ...]
```

For example, to record only information about Apache and Tomcat, and to gather all the Apache logs:

```bash
# sosreport -o apache,tomcat -k apache.log=on
```

To enable all boolean options for all loaded plugins except the `rpm.rpmva` plugin that verifies all packages, and which takes a considerable time to run:

```bash
# sosreport -a -k rpm.rpmva=off
```

2. Type Enter, and enter additional information when prompted.

```
Please enter your first initial and last name [email_address]: AName
Please enter the case number that you are generating this report for: case#

Running plugins. Please wait ...

Completed [55/55] ...
Creating compressed archive...

Your sosreport has been generated and saved in:
  /tmp/sosreport-AName.case#-datestamp-ID.tar.xz

The md5sum is: checksum

Please send this file to your support representative.
```

`sosreport` saves the report as an `xz`-compressed `tar` file in `/tmp`.

14.2 About Kdump

Kdump is the Linux kernel crash-dump mechanism. Oracle recommends that you enable the Kdump feature. In the event of a system crash, Kdump creates a memory image (`vmcore`) that can help in determining the cause of the crash. Enabling Kdump requires you to reserve a portion of system memory for exclusive use by Kdump. This memory is unavailable for other uses.

Kdump uses `kexec` to boot into a second kernel whenever the system crashes. `kexec` is a fast-boot mechanism which allows a Linux kernel to boot from inside the context of a kernel that is already running without passing through the bootloader stage.

14.2.1 Configuring and Using Kdump

During installation, you are given the option of enabling Kdump and specifying the amount of memory to reserve for it. If you prefer, you can enable kdump at a later time as described in this section.

If the `kexec-tools` and `system-config-kdump` packages are not already installed on your system, use `yum` to install them.

To enable Kdump by using the Kernel Dump Configuration GUI.

1. Enter the following command.
Configuring and Using Kdump

# system-config-kdump

The Kernel Dump Configuration GUI starts. If Kdump is currently disabled, the green Enable button is selectable and the Disable button is greyed out.

2. Click Enable to enable Kdump.

3. You can select the following settings tags to adjust the configuration of Kdump.

- **Basic Settings**
  - Allows you to specify the amount of memory to reserve for Kdump. The default setting is 128 MB.

- **Target Settings**
  - Allows you to specify the target location for the vmcore dump file on a locally accessible file system, to a raw disk device, or to a remote directory using NFS or SSH over IPv4. The default location is /var/crash.

  You cannot save a dump file on an eCryptfs file system, on remote directories that are NFS mounted on the rootfs file system, or on remote directories that access require the use of IPv6, SMB, CIFS, FCoE, wireless NICs, multipathed storage, or iSCSI over software initiators to access them.

- **Filtering Settings**
  - Allows to select which type of data to include in or exclude from the dump file. Selecting or deselecting the options alters the value of the argument that Kdump specifies to the -d option of the core collector program, makedumpfile.

- **Expert Settings**
  - Allows you to choose which kernel to use, edit the command line options that are passed to the kernel and the core collector program, choose the default action if the dump fails, and modify the options to the core collector program, makedumpfile.

  For example, if Kdump fails to start, and the following error appears in /var/log/messages, set the offset for the reserved memory to 48 MB or greater in the command line options, for example crashkernel=128M@48M:

  kdump: No crashkernel parameter specified for running kernel

  The Unbreakable Enterprise Kernel supports the use of the crashkernel=auto setting for UEK Release 3 Quarterly Update 1 and later. If you use the crashkernel=auto setting, the output of the dmesg command shows crashkernel=XM@0M, which is normal. The setting actually reserves 128 MB plus 64 MB for each terabyte of physical memory.

  **Note**
  
  You cannot configure crashkernel=auto for Xen or for the UEK prior to UEK Release 3 Quarterly Update 1. Only standard settings such as crashkernel=128M@48M are supported. For systems with more than 128 GB of memory, the recommended setting is crashkernel=512M@64M.
Click Help for more information on these settings.

4. Click Apply to save your changes. The GUI displays a popup message to remind you that you must reboot the system for the changes to take effect.

5. Click OK to dismiss the popup messages.


7. Reboot the system at a suitable time.

### 14.2.2 Files Used by Kdump

The Kernel Dump Configuration GUI modifies the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/boot/grub/grub.conf</td>
<td>Appends the crashkernel option to the kernel line to specify the amount of reserved memory and any offset value.</td>
</tr>
<tr>
<td>/etc/kdump.conf</td>
<td>Sets the location where the dump file can be written, the filtering level for the makedumpfile command, and the default behavior to take if the dump fails. See the comments in the file for information about the supported parameters.</td>
</tr>
</tbody>
</table>

If you edit these files, you must reboot the system to have the changes take effect.

### 14.3 About OSWatcher Black Box

Oracle OSWatcher Black Box (OSWbb) collects and archives operating system and network metrics that you can use to diagnose performance issues. OSWbb operates as a set of background processes on the server and gathers data on a regular basis, invoking such Unix utilities as vmstat, netstat, iostat, and top.

From release v4.0.0, you can use the OSWbb analyzer to provide information on system slowdowns, system hangs and other performance problems, and also to graph data collected from iostat, netstat, and vmstat. OSWbb requires that you have installed Java version 1.4.2 or higher on your system. You can use yum to install Java, or you can download a Java RPM for Linux from http://www.java.com.

OSWbb is particularly useful for Oracle RAC (Real Application Clusters) and Oracle Grid Infrastructure configurations. The RAC-DDT (Diagnostic Data Tool) script file includes OSWbb, but does not install it by default.

### 14.3.1 Installing OSWbb

To install OSWbb:


3. Copy the file to the directory where you want to install OSWbb, and run the following command:

   ```
   # tar xvf oswbb601.tar
   ```

   Extracting the tar file creates a directory named oswbb, which contains all the directories and files that are associated with OSWbb, including the startOSWbb.sh script.
4. If the `ksh` package is not already installed on your system, use `yum` to install it.

   ```bash
   # yum install ksh
   ```

5. Create a symbolic link from `/usr/bin/ksh` to `/bin/ksh`.

   ```bash
   # ln -s /bin/ksh /usr/bin/ksh
   ```

   This link is required because the OSWbb scripts expect to find `ksh` in `/usr/bin`.

6. To enable the collection of `iostat` information for NFS volumes, edit the `OSWatcher.sh` script in the `oswbb` directory, and set the value of `nfs_collect` to 1:

   ```bash
   nfs_collect=1
   ```

   **Note**
   
   This feature is available from release v5.1.

### 14.3.2 Running OSWbb

To start OSWbb, run the `startOSWbb.sh` script from the `oswbb` directory.

```bash
# ./startOSWbb.sh [frequency duration]
```

The optional frequency and duration arguments specifying how often in seconds OSWbb should collect data and the number of hours for which OSWbb should run. The default values are 30 seconds and 48 hours. The following example starts OSWbb recording data at intervals of 60 seconds, and has it record data for 12 hours:

```bash
# ./startOSWbb.sh 60 12
```

VMSTAT found on your system.
IOSTAT found on your system.
MPSTAT found on your system.
NETSTAT found on your system.
TOP found on your system.
Testing for discovery of OS CPU COUNT

Starting Data Collection...

```
```

OSWbb heartbeat: date/time
OSWbb heartbeat: date/time + 60 seconds

To stop OSWbb prematurely, run the `stopOSWbb.sh` script from the `oswbb` directory.

```bash
# ./stopOSWbb.sh
```

OSWbb collects data in the following directories under the `oswbb/archive` directory:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oswiostat</td>
<td>Contains output from the <code>iostat</code> utility.</td>
</tr>
<tr>
<td>oswmeminfo</td>
<td>Contains a listing of the contents of <code>/proc/meminfo</code>.</td>
</tr>
</tbody>
</table>
For More Information About the Diagnostic Tools

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oswmpstat</td>
<td>Contains output from the <code>mpstat</code> utility.</td>
</tr>
<tr>
<td>oswnetstat</td>
<td>Contains output from the <code>netstat</code> utility.</td>
</tr>
<tr>
<td>oswprvtnet</td>
<td>If you have enable private network tracing for RAC, contains information about the status of the private networks.</td>
</tr>
<tr>
<td>oswps</td>
<td>Contains output from the <code>ps</code> utility.</td>
</tr>
<tr>
<td>oswslabinfo</td>
<td>Contains a listing of the contents of <code>/proc/slabinfo</code>.</td>
</tr>
<tr>
<td>oswtop</td>
<td>Contains output from the <code>top</code> utility.</td>
</tr>
<tr>
<td>oswvmstat</td>
<td>Contains output from the <code>vmstat</code> utility.</td>
</tr>
</tbody>
</table>

OSWbb stores data in hourly archive files named `system_name_utility_name_timestamp.dat`, and each entry in a file is preceded by the characters *** and a timestamp.

14.4 For More Information About the Diagnostic Tools

For more information about `sosreport`, see the `sosreport(1)` manual page.

For more information about Kdump, refer to the help in the Kernel Dump Configuration GUI, and the `makedumpfile(8)` manual page.

For more information about OSWbb and OSWbba, refer to the OSWatcher Black Box User Guide (Article ID 301137.1) and the OSWatcher Black Box Analyzer User Guide (Article ID 461053.1), which are available from My Oracle Support (MOS) at http://support.oracle.com.