Oracle Linux 8

Monitoring and Tuning the System

F24025-12
April 2022

Oracle Linux 8 Monitoring and Tuning the System,

F24025-12

Copyright © 2019, 2022, Oracle and/or its affiliates.

Contents

Preface

Oracle® Linux 8: Monitoring and Tuning the System describes the various utilities, features, and services that you can use to monitor system performance, detect performance issues, and improve the performance of various system components.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our

products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Monitoring the System and Optimizing Performance

Performance issues can be caused by a number of your system's components, including software or hardware, as well as any related interactions. Many performance diagnostics utilities are available in Oracle Linux and include tools that monitor and analyze the resource usage of different hardware components, as well as tracing tools for diagnosing performance issues in multiple processes and related threads.

Many performance issues are the result of configuration errors. You can avoid these errors by using a validated configuration that has been pre-tested for the supported software, hardware, storage, drivers, and networking components. A validated configuration incorporates best practices for an Oracle Linux deployment and has undergone real-world testing of the complete stack. Oracle publishes many validated configurations, which are freely available for download. Refer to the release notes for the release that you are running for additional recommendations on kernel parameter settings.

Working With System Performance and Monitoring Utilities

The following utilities enable you to collect information about system resource usage and errors, and help you to identify performance problems that are caused by overloaded disks, network, memory, or CPUs:

`dmesg`
Displays the contents of the kernel ring buffer, which can contain errors about system resource usage. Provided by the `util-linux` package.

`dstat`
Displays statistics about system resource usage. Provided by the `dstat` package.

`free`
Displays the amount of free and used memory in the system. Provided by the `procps` package, which is install by default in Oracle Linux 8.

`iostat`
Reports I/O statistics. Provided by the `sysstat` package.

`iotop`
Monitors disk and swap I/O on a per-process basis. Provided by the `iotop` package.

`ip`
Reports network interface statistics and errors. Provided by the `iproute` package, which is installed by default in Oracle Linux 8.

`mpstat`
Reports processor-related statistics. Provided by the `sysstat` package.

nfsiostat
Reports I/O statistics for NFS mounts. Provided by the `nfs-utils` package.

sar
Reports information about system activity. Provided by the `sysstat` package.

ss
Reports network interface statistics. Provided by the `iproute` package.

top
Provides a dynamic real-time view of the tasks that are running on a system. Provided by the `procps` package.

uptime
Displays the system load averages for the past 1, 5, and 15 minutes. Provided by the `procps` package.

vmstat
Reports virtual memory statistics. Provided by the `procps` package, which is installed by default in Oracle Linux 8.

Many of these utilities provide overlapping functionality. For more information, see the individual manual page for the utility.

See https://docs.oracle.com/en/learn/system_monitoring_linux8/ for a hands-on tutorial and associated video content on many of these utilities.

Monitoring the Usage of System Resources

You need to collect and monitor system resources so that you are provided with a continuous record of a system's performance. First, establish a baseline of acceptable measurements under typical operating conditions. You can then use that baseline as a reference point to make it easier to identify memory shortages, spikes in resource usage, and other problems when they occur. Monitoring system performance also enables you to plan for future growth and determine how configuration changes might affect future performance.

To run a monitoring command for a set number of seconds in real time and watch the output change, use the `watch` command. For example, run the `mpstat` command once per second with the following command:

```
sudo watch -n 1 mpstat
```

Alternatively, many of the commands enable you to specify the sampling interval in seconds, for example:

```
sudo mpstat seconds
```

If it is installed, the `sar` command records statistics every 10 minutes while the system is running and retains that information for every day of the current month. The following command displays all the statistics that `sar` recorded for the day (*DD*) of the current month:

```
sudo sar -A -f /var/log/sa/saDD
```

To run the `sar` command as a background process and collect data in a file that you can display later by using the `-f` option:

```
sudo sar -o datafile seconds count >/dev/null 2>&1 &
```

In the previous command, *count* is the number of samples to record.

See also Working With OSWatcher Black Box and Working With Performance Co-Pilot.

Monitoring CPU Usage

The `uptime`, `mpstat`, `sar`, `dstat`, and `top` utilities enable you to monitor CPU usage. When your system's CPU cores are occupied with executing code processes, other processes must wait until a CPU core becomes free or when the scheduler switches a CPU core to run its code. If too many processes are queued too often, then that can represent a bottleneck in the performance of the system.

The commands `mpstat -P ALL` and `sar -u -P ALL` display CPU usage statistics for each CPU core and is averaged across all of the CPU cores.

The `%idle` value shows the percentage of time that a CPU was not running system or process code. If the value of `%idle` is near 0% most of the time on all CPU cores, the system is CPU-bound for the workload that it is running. The percentage of time spent running system code (`%system` or `%sys`) should not usually exceed 30%, especially if `%idle` is close to 0%.

The system load average represents the number of processes that are running on CPU cores, waiting to run, or waiting for disk I/O activity to complete averaged over a period of time. On a busy system, the load average reported by `uptime` or `sar -q` should usually be not greater than two times the number of CPU cores over periods as long as 5 or 15 minutes. If the load average exceeds four times the number of CPU cores for long periods, the system is overloaded.

In addition to load averages (`ldavg-*`), the `sar -q` command reports the number of processes currently waiting to run (the *run-queue size*, `runq-sz`) and the total number of processes (`plist_sz`). The value of `runq-sz` also provides an indication of CPU saturation.

Determine the system's average load under normal loads, where users and applications do not experience problems with system responsiveness, and then look for deviations from this benchmark over time. A dramatic rise in the load average can indicate a serious performance problem.

A combination of sustained large load average or large run queue size and low `%idle` can indicate that the system has insufficient CPU capacity for the workload. When CPU usage is high, use a command such as `dstat` or `top` to determine which processes are most likely to be responsible. For example, the following `dstat` command shows which processes are using CPUs, memory, and block I/O most intensively:

```
dstat --top-cpu --top-mem --top-bio
```

The `top` command provides a real-time display of CPU activity. By default, `top` lists the most CPU-intensive processes on the system. In its upper section, `top` displays general information including the load averages over the past 1, 5 and 15 minutes, the number of running and sleeping processes (tasks), and total CPU and memory usage. In its lower section, `top` displays a list of processes, including the process ID number (PID), the process owner, CPU usage, memory usage, running time, and the command name. By default, the list is sorted by CPU usage, with the top consumer of CPU listed first. Type `f` to select which fields `top` displays, `o` to change the order of the fields, or `O` to change the sort field. For example, entering `On` sorts the list on the percentage memory usage field (`%MEM`).

Monitoring Memory Usage

The `sar -r` command reports memory utilization statistics, including `%memused`, which is the percentage of physical memory in use.

`sar -B` reports memory paging statistics, including `pgscank/s`, which is the number of memory pages scanned by the `kswapd` daemon per second, and `pgscand/s`, which is the number of memory pages scanned directly per second.

`sar -W` reports swapping statistics, including `pswpin/s` and `pswpout/s`, which are the numbers of pages per second swapped in and out per second.

If `%memused` is near 100% and the scan rate is continuously over 200 pages per second, the system has a memory shortage.

Once a system runs out of real or physical memory and starts using swap space, its performance deteriorates dramatically. If you run out of swap space, your programs or the entire operating system are likely to crash. If `free` or `top` indicate that little swap space remains available, this is also an indication you are running low on memory.

The output from the `dmesg` command might include notification of any problems with physical memory that were detected at boot time.

Using Adaptive Memory Management

The Adaptive Memory Management daemon is a user space service that monitors free memory on your Oracle Linux system and predicts memory fragmentation and usage. It can also automatically reclaim memory if the system if memory becomes too fragmented or is at risk of being filled to capacity.

If the system memory becomes highly fragmented, `adaptivemmd` triggers the kernel to compact memory so that fragmented space can be reclaimed before it is reallocated. If the system is likely to exhaust the available memory, watermarks are adjusted and this can trigger the kernel to free up new pages in memory. Adaptive Memory Management is available in Unbreakable Enterprise Kernel Release 6 and later.

Use `dnf` to install the `adaptivemm` package from the `ol8_UEKR6` yum repository or equivalent ULN channel:

```
sudo dnf install -y adaptivemm
```

To enable and run the `adaptivemmd` service:

```
sudo systemctl enable --now adaptivemmd
```

You can change the configuration options in `/etc/sysconfig/adaptivemmd`.

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

Monitoring Block I/O Usage

The `iostat` command monitors the loading of block I/O devices by observing the time that the devices are active relative to the average data transfer rates. You can use this information to adjust the system configuration to balance the I/O loading across disks and host adapters.

`iostat -x` reports extended statistics about block I/O activity at one second intervals, including `%util`, which is the percentage of CPU time spent handling I/O requests to a device, and `avgqu-sz`, which is the average queue length of I/O requests that were issued to that device. If `%util` approaches 100% or `avgqu-sz` is greater than 1, device saturation is occurring.

You can also use the `sar -d` command to report on block I/O activity, including values for `%util` and `avgqu-sz`.

The `iotop` utility can help you identify which processes are responsible for excessive disk I/O. `iotop` has a similar user interface to `top`. In its upper section, `iotop` displays the total disk input and output usage in bytes per second. In its lower section, `iotop` displays I/O information for each process, including disk input output usage in bytes per second, the percentage of time spent swapping in pages from disk or waiting on I/O, and the command name. Use the left and right arrow keys to change the sort field, and press `A` to toggle the I/O units between bytes per second and total number of bytes, or `O` to toggle between displaying all processes or only those processes that are performing I/O.

Monitoring File System Usage

The `sar -v` command reports the number of unused cache entries in the directory cache (`dentunusd`) and the numbers of in-use file handles (`file-nr`), inode handlers (`inode-nr`), and pseudo terminals (`pty-nr`).

`nfsiostat` reports I/O statistics for each NFS file system that is mounted. If this command is not available install the `nfs-utils` package.

Monitoring Network Usage

The `ip -s link` command displays network statistics and errors for all network devices, including the numbers of bytes transmitted (`TX`) and received (`RX`). The `dropped` and `overrun` fields provide an indicator of network interface saturation, for example:

```
ip -s link
```

The following output is displayed:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    RX: bytes  packets  errors  dropped overrun mcast
    240        4        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    240        4        0       0       0       0
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
    link/ether 08:00:27:60:95:d5 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    258187485  671730   0       0       0       17296
    TX: bytes  packets  errors  dropped carrier collsns
    13227598   130827   0       0       0       0
```

The `ss -s` command displays summary statistics for each protocol, for example:

```
ss -s
```

The following output is displayed:

```
Total: 193
TCP:   9 (estab 2, closed 0, orphaned 0, timewait 0)
```

Using the Graphical System Monitor

The GNOME desktop environment includes a graphical system monitor that enables you to display information about the system configuration, running processes, resource usage, and file systems.

To display the System Monitor, use the following command:

```
gnome-system-monitor
```

Selecting the **Resources** tab displays the following information:

- CPU usage history in graphical form and the current CPU usage as a percentage.
- Memory and swap usage history in graphical form and the current memory and swap usage.
- Network usage history in graphical form, the current network usage for reception and transmission, and the total amount of data received and transmitted.

To display the System Monitor Manual, press `F1` or select Help, then select Contents.

2
Working With the sos Command

The `sos` command collects information about a system such as hardware configuration, software configuration, and operational state. You can also use the `sos report` command to enable diagnostics and analytical functions on the current system. To assist in troubleshooting a problem, the `sos report` command records the information in a compressed file that you can send to a support representative.

Installing the sos Package

If the `sos` package is not already installed on your system, use `dnf` to install it:

```
sudo dnf install sos
```

Use the `sos report -l` command to list the available plugins and plugin options:

```
sudo sos report -l

The following plugins are currently enabled:
 alternatives         System alternatives
 anaconda             Anaconda installer
.
.
.
The following plugins are currently disabled:
 abrt                 inactive       Automatic Bug Reporting Tool
 acpid                inactive       ACPI daemon information
.
.
.
The following options are available for ALL plugins:
 timeout                   -1             Timeout in seconds for plugin
 cmd-timeout               -1             Timeout in seconds for a command execution.
.
.
.
The following plugin options are available:
 boot.all-images           off            collect lsinitrd for all images
 btrfs.details             off            gather details for every btrfs filesystem
.
.
.
```

See the `sos-report(1)` manual page for information about how to enable or disable plugins and how to set values for plugin options.

Running the sos Command

You can run the `sos report` command to record information about a problem area and specify options to tailor the report it generates as follows:

```
sudo sos report [options ...]
```

For example, to record only information about Apache and Tomcat and to gather all of the Apache logs, use the following command:

```
sudo sos report -o apache,tomcat -k apache.log=on
```

The output is similar to the following:

```
sosreport (version 4.1)
.
.
.
Press ENTER to continue, or CTRL-C to quit.
```

To enable all of the boolean options for all of the loaded plugins (excluding the `rpm.rpmva` plugin) and verify all packages:

```
sudo sos report -a -k rpm.rpmva=off
```

Note that this process can take a considerable amount of time, but once it has completed, press Enter and then provide any additional information that is required:

```
Please enter the case number that you are generating this report for []: case#
...
Finished running plugins
Creating compressed archive...

Your sosreport has been generated and saved in:
        /var/tmp/sosreport-hostname-case#-datestamp-ID.tar.xz

 Size   20.62MiB
 Owner  root
 sha256 428f7b4118acd2d349bb022946877d853aa0eefbb4d340af3839810dc634b8b7

Please send this file to your support representative.
```

The `sos report` command saves the report as an `xz`-compressed `tar` file in the `/tmp` directory.

Optionally, to obfuscate sensitive information, you can run the `sos clean` command on the compressed archive generated from the `sos report` command.

The cleaned report obfuscates the following details:

- IPv4 addresses and networks (network topologies are retained)
- MAC addresses
- Host names
- User names
- Any words or phrases that you specify with the `--keyword` option

> **Note:**
>
> Reports processed with the `sos clean` command obfuscate certain details that may be needed for advanced troubleshooting, such as networking information.

To generate a cleaned report, run the `sos clean` command on the compressed archive generated from the `sos report` command in the `/tmp` directory:

```
sudo sos clean /var/tmp/sosreport-hostname-case#-datestamp-ID.tar.xz
```

Press Enter to proceed. After the `sos clean` command completes, a new `xz`-compressed `tar` file with `-obfuscated` in the file name is created the `/tmp` directory.

You can also perform this step automatically by appending the `--clean` option to the `sos report` command, as follows:

```
sudo sos report --clean [options ...]
```

For more information, see the `sos-report(1)` and `sos-clean(1)` manual pages. See also https://github.com/sosreport/sos/wiki.

Reviewing Information Gathered by sosreport

The `sos` command is automatically configured to collect hardware information, system configuration files and log data; but, you can enable and disable modules to suit your own data protection needs.

> **Note:**
>
> The module information that is provided in this table relates to `sos` 3.9. To verify the modules you have installed on your system, read Installing the sos Package.
>
> Disabling modules prevents the `sos` command from collecting certain details that may be needed for advanced troubleshooting, such as networking information.

Module	Information Type	Included Files
`anaconda`	Installation log files	• `/root/install.log` • `/root/install.log.syslog` • `/var/log/anaconda` • `/var/log/anaconda.*`
`auditd`	Audit log files	• `/etc/audit/auditd.conf` • `/etc/audit/audit.rules` • `/var/log/audit/*`

Module	Information Type	Included Files
boot	System boot process details	• /etc/milo.conf • /etc/silo.conf • /boot/efi/efi/redhat/elilo.conf • /etc/yaboot.conf • /boot/yaboot.conf
cron	Root user cron commands	• /etc/cron* • /etc/crontab • /var/log/cron • /var/spool/cron
cups	Printer log files	• /etc/cups/*.conf • /etc/cups/*.types • /etc/cups/lpoptions • /etc/cups/ppd/*.ppd • /var/log/cups/*
date	Context data	• /etc/localtime
devicemapper	Hardware details	
filesys	List of all files in use	• /proc/fs/* • /proc/mounts • /proc/filesystems • /proc/self/mounts • /proc/self/mountinfo • /proc/self/mountstats • /proc/[0-9]*/mountinfo • /etc/mtab • /etc/fstab
grub2	Kernel and system start-up configuration	• /boot/efi/EFI/*/grub.cfg • /boot/grub2/grub.cfg • /boot/grub2/grubenv • /boot/grub/grub.cfg • /boot/loader/entries • /etc/default/grub • /etc/grub2.cfg • /etc/grub.d/*
hardware	Hardware details	• /proc/interrupts • /proc/irq • /proc/dma • /proc/devices • /proc/rtc • /var/log/mcelog • /sys/class/dmi/id/* • /sys/class/drm/*/edid

Module	Information Type	Included Files
`host`	Host identification	• `/etc/sos.conf` • `/etc/hostid`

Module	Information Type	Included Files
kernel	System log files	• /etc/conf.modules • /etc/modules.conf • /etc/modprobe.conf • /etc/modprobe.d • /etc/sysctl.conf • /etc/sysctl.d • /lib/modules/*/modules.dep • /lib/sysctl.d • /proc/cmdline • /proc/driver • /proc/kallsyms • /proc/lock* • /proc/buddyinfo • /proc/misc • /proc/modules • /proc/slabinfo • /proc/softirqs • /proc/sys/kernel/random/boot_id • /proc/sys/kernel/tainted • /proc/timer* • /proc/zoneinfo • /sys/firmware/acpi/* • /sys/kernel/debug/tracing/* • /sys/kernel/livepatch/* • /sys/module/*/parameters • /sys/module/*/initstate • /sys/module/*/refcnt • /sys/module/*/taint • /sys/module/*/version • /sys/devices/system/clocksource/*/available_clocksource • /sys/devices/system/clocksource/*/current_clocksource • /sys/fs/pstore • /var/log/dmesg

Module	Information Type	Included Files
libraries	List of shared libraries	• /etc/ld.so.conf • /etc/ld.so.conf.d/*
logs	System log files	• /etc/syslog.conf • /etc/rsyslog.conf • /etc/rsyslog.d • /run/log/journal/* • /var/log/auth.log • /var/log/auth.log.1 • /var/log/auth.log.2* • /var/log/boot.log • /var/log/dist-upgrade • /var/log/installer • /var/log/journal/* • /var/log/kern.log • /var/log/kern.log.1 • /var/log/kern.log.2* • /var/log/messages* • /var/log/secure* • /var/log/syslog • /var/log/syslog.1 • /var/log/syslog.2* • /var/log/udev • /var/log/unattended-upgrades
lvm2	Hardware details	
memory	Hardware details	• /proc/pci • /proc/meminfo • /proc/vmstat • /proc/swaps • /proc/slabinfo • /proc/pagetypeinfo • /proc/vmallocinfo • /sys/kernel/mm/ksm • /sys/kernel/mm/transparent_hugepage/enabled

Module	Information Type	Included Files
networking	Network identification	• /etc/dnsmasq* • /etc/host* • /etc/inetd.conf • /etc/iproute2 • /etc/network* • /etc/nftables • /etc/nftables.conf • /etc/nsswitch.conf • /etc/resolv.conf • /etc/sysconfig/nftables.conf • /etc/xinetd.conf • /etc/xinetd.d • /etc/yp.conf • /proc/net/* • /sys/class/net/*/device/numa_node • /sys/class/net/*/flags • /sys/class/net/*/statistics/*
pam	Login security settings	• /etc/pam.d/* • /etc/security
pci	Hardware details	• /proc/bus/pci • /proc/iomem • /proc/ioports
process	List of all running processes and process details	• /proc/sched_debug • /proc/stat • /proc/[0-9]*/smaps
processor	Hardware details	• /proc/cpuinfo • /sys/class/cpuid • /sys/devices/system/cpu
rpm	Installed software packages	• /var/lib/rpm/* • /var/log/rpmpkgs
sar	Resource and usage data	• /var/log/sa/*
selinux	Security settings	• /etc/sestatus.conf • /etc/selinux • /var/lib/selinux
services	All defined system services	• /etc/inittab • /etc/rc.d/* • /etc/rc.local
ssh	SSH configuration	• /etc/ssh/ssh_config • /etc/ssh/sshd_config

Module	Information Type	Included Files
x11	GUI logs for the X Window System	• /etc/X11/* • /var/log/Xorg.*.log • /var/log/Xorg.*.log.old • /var/log/XFree86.*.log • /var/log/XFree86.*.log.old
yum	Installed software packages	• /etc/pki/consumer/cert.pem • /etc/pki/entitlement/*.pem • /etc/pki/product/*.pem • /etc/yum/* • /etc/yum.repos.d/* • /etc/yum/pluginconf.d/* • /var/log/dnf.log

3
Working With 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`, `mpstat`, `netstat`, `iostat`, and `top`.

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.

Installing OSWbb

To install OSWbb:

1. Log in to My Oracle Support (MOS) at https://support.oracle.com.
2. Download OSWatcher from the link that is listed by Doc ID 301137.1 at https://support.oracle.com/epmos/faces/DocumentDisplay?id=301137.1.
3. Copy the file to the directory that you want to install OSWbb, then run the following command:

   ```
   tar xvf oswbbVERS.tar
   ```

   In the previous command, *VERS* represents the version number of OSWatcher, for example 832 for OSWatcher 8.32.

   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. 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` as follows:

   ```
   nfs_collect=1
   ```

Running OSWbb

To start OSWbb, run the `startOSWbb.sh` script from the `oswbb` directory.

```
sudo ./startOSWbb.sh [frequency duration]
```

The optional frequency and duration arguments specify 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:

```
sudo ./startOSWbb.sh 60 12

...
Testing for discovery of OS Utilities...
VMSTAT found on your system.
```

```
IOSTAT found on your system.
MPSTAT found on your system.
IFCONFIG found on your system.
NETSTAT found on your system.
TOP found on your system.

Testing for discovery of OS CPU COUNT
oswbb is looking for the CPU COUNT on your system
CPU COUNT will be used by oswbba to automatically look for cpu problems

CPU COUNT found on your system.
CPU COUNT = 4

Discovery completed.

Starting OSWatcher Black Box v7.3.0  on date and time
With SnapshotInterval = 60
With ArchiveInterval = 12
...
Data is stored in directory: OSWbba_archive

Starting Data Collection...

oswbb heartbeat: date and time
oswbb heartbeat: date and time + 60 seconds
...
```

In the previous output, *OSWbba_archive* is the path of the archive directory that contains the OSWbb log files.

To stop OSWbb prematurely, run the `stopOSWbb.sh` script from the `oswbb` directory:

```
sudo ./stopOSWbb.sh
```

OSWbb collects data in the directories that are under the `oswbb/archive` directory, which are described in the following table.

Directory	Description
`oswifconfig`	Contains output from `ifconfig`.
`oswiostat`	Contains output from `iostat`.
`oswmeminfo`	Contains a listing of the contents of `/proc/meminfo`.
`oswmpstat`	Contains output from `mpstat`.
`oswnetstat`	Contains output from `netstat`.
`oswprvtnet`	If you have enable private network tracing for RAC, contains information about the status of the private networks.
`oswps`	Contains output from `ps`.
`oswslabinfo`	Contains a listing of the contents of `/proc/slabinfo`.
`oswtop`	Contains output from `top`.
`oswvmstat`	Contains output from `vmstat`.

OSWbb stores data in hourly archive files, which are named *system_name_utility_name_timestamp*.dat. Each entry in a file is preceded by a timestamp.

Analyzing OSWbb Archived Files

You can use the OSWbb analyzer (OSWbba) to provide information about system slowdowns, system hangs, and other performance problems. You can also use OSWbba to graph data that is collected from the `iostat`, `netstat`, and `vmstat` utilities. OSWbba requires that you have Java version 1.4.2 or a later version installed on your system.

You can download a Java RPM for Linux by visiting http://www.java.com, or you can install Java by using the `dnf` command:

```
sudo dnf install java-1.8.0-jdk
```

Run OSWbba from the `oswbb` directory as follows:

```
sudo java -jar oswbba.jar -i OSWbba_archive
```

In the previous command, *OSWbba_archive* is the path of the archive directory that contains the OSWbb log files.

You can use OSWbba to display the following types of performance graph:

- Process run, wait and block queues.
- CPU time spent running in system, user, and idle mode.
- Context switches and interrupts.
- Free memory and available swap.
- Reads per second, writes per second, service time for I/O requests, and percentage utilization of bandwidth for a specified block device.

You can also use OSWbba to save the analysis to a report file, which reports instances of system slowdown, spikes in run queue length, or memory shortage, describes probable causes, and offers suggestions of how to improve performance.

```
sudo java -jar oswbba.jar -i OSWbba_archive -A
```

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) on My Oracle Support (MOS) at https://support.oracle.com.

4
Working With Performance Co-Pilot

Performance Co-Pilot (PCP) collects operating system and network metrics that you can use to diagnose performance issues. PCP provides a monitor host that you can use to send requests for metrics and logs to a pair of collector host services that are installed on each Oracle Linux system that you monitor.

Installing PCP

1. Enable the `ol8_appstream` and `ol8_addons` yum repositories on your system.

   For more information, see Oracle® Linux: Managing Software on Oracle Linux.

2. Install the `pcp-oracle-conf`, `pcp-system-tools` and `pcp-gui` packages by using the `dnf` command:

   ```
   sudo dnf install pcp-oracle-conf pcp-system-tools pcp-gui
   ```

3. Enable and start the Performance Metrics Collector Daemon (`pmcd`) and Performance Metrics Logger (`pmlogger`) collector host services:

   ```
   sudo systemctl enable --now pmcd pmlogger
   ```

Reviewing Information Gathered by PCP

The only metrics collected by the `pmlogger` service are those listed in the `/var/lib/pcp/config/pmlogger/config.default` file.

You can modify the frequency with which those metrics are collected in the same configuration file. For example, to increase the frequency from once every minute to once every 5 seconds:

```
...
# It is safe to make additions from here on ...
#

log mandatory on every 5 seconds {
  filesys.free
  filesys.used
  ...
}
```

All of the archives that the `pmlogger` service generates are stored in the `/var/log/pcp/pmlogger/`*`hostname`* directory. For more information, see the `pmlogconf(1)` manual page.

To verify the PCP configuration at the time that `pmlogger` collected specific performance metrics, use the `pcp` command:

```
sudo pcp -a 20220321.0.xz
```

Using PCP Monitor Host to Analyze Performance Metrics

All of the archives that the `pmlogger` service generates are stored in the `/var/log/pcp/pmlogger/`*`hostname`* directory. If you navigate to that directory in a terminal, you can run a number of commands to review the performance metrics that have been collected.

For more information about those commands and their parameters, see their respective manual pages.

Review Live Performance Metrics in Real Time

To monitor all the outgoing metrics from the `eth0` network interface in real time, use the `pmrep` command:

```
sudo pmrep -i eth0 -v network.interface.out
```

To monitor live hard drive operations for each partition with a two second interval, use the `pmval` command:

```
sudo pmval -t 2sec -f 3 disk.partitions.write
```

Review Recorded Performance Metrics

To review the data for specific performance metrics within a specified timespan, use the `pmdumptext` command. For example, to review resource usage metrics for CPU load, memory utilisation and disk write operations between 13:00 and 14:00 on a specific date:

```
sudo pmdumptext -Xlimu -t 10m -S @13:00 -T @14:00 \
 'kernel.all.load[1]' 'mem.util.used' 'disk.partitions.write' -a 20220321.0.xz
```

You can also use the `pmstat` to review system performance metrics in a format similar to that produced by the `sar` command. For example, to review performance metrics averaged over 10 minute interval between 09:00 and 10:00 on a specific date:

```
sudo pmstat -t 10m -S @09:00 -T @10:00 -a 20220321.0.xz
```

You can also compare the metrics between two time periods by using the `pmdiff` command. For example, to compare the metrics between 02:00 and 03:00 on one day to the metrics between 09:00 and 10:00 on a different day:

```
sudo pmdiff -S @02:00 -T @03:00 -B @09:00 -E @10:00 20220321.0.xz 20220320.0.xz
```

Review Details About Recorded Performance Metrics

To review detailed information about a specific metric, use the `pminfo` command. For example, to review details about free memory:

```
sudo pminfo -df mem.freemem -a 20220321.0.xz
```

Validate System Status When Performance Metrics Were Captured

To verify the host, timezone and time period that an archive containing performance metrics contains, use the `pmdumplog` command:

```
sudo pmdumplog -L 20220321.0.xz
```

To review a list of every enabled performance metric, use the `pminfo` command:

```
sudo pminfo -a 20220321.0.xz
```

5

Working With Tuned

You use the Tuned tool to monitor a system to optimize its performance under certain conditions. Tuned uses several predefined profiles to tune your system. The profiles that are provided are designed for particular use cases and fall into one of the following two categories: power-saving profiles and performance-boosting profiles. Performance-boosting profiles address low latency and high throughput for storage and the network and virtualization host performance.

You can modify the rules that are defined for each profile, as well as customize how a specific device is tuned by using a specific profile. In addition, you can configure Tuned so that any changes in device usage triggers an adjustment in the current settings so that the performance of active devices is improved and power consumption for inactive devices is reduced.

About Tuned Profiles

The following Tuned profiles are typically installed with Oracle Linux 8:

- `balanced` (default profile): Is a power-saving profile. This profile provides a balance between performance and power consumption. The profile uses auto-scaling and auto-tuning when possible. A possible drawback is increased latency.
- `powersave`: Is a profile that provides maximum power saving performance. The profile can minimize actual power consumption by throttling performance.

  > **Note:**
  >
  > In some instances, the `balanced` profile is a better choice than the `powersave` profile, as it is more efficient.

- `throughput-performance` (default profile): Is a server profile that is optimized for high throughput. The profile disables power-savings mechanisms and enables `sysctl` settings to improve the throughput performance of the disk and network IO.
- `latency-performance`: Is a server profile that is optimized for low latency. The profile disables power-savings mechanisms and enables `sysctl` settings to improve latency.
- `network-latency`: Is a profile that provides low latency network tuning and is based on the `latency-performance` profile. In addition, this profile disables transparent huge pages and NUMA balancing and tunes several network-related `sysctl` settings.
- `network-throughput`: A profile for throughput network tuning. It is based on the `throughput-performance` profile. In addition, this profile increases kernel network buffers.
- `virtual-guest` (default profile): Is a profile that is designed for virtual guests and is based on the `throughput-performance` profile. This profile decreases virtual memory swappiness and increases disk readahead values.

- `virtual-host`: Is a profile that is designed for virtual hosts and is based on the `throughput-performance` profile. This profile decreases virtual memory swappiness, increases disk readahead values, and enables a more aggressive value of dirty pages writeback.
- `desktop`: Is a profile that is optimized for desktop environments and is based on the `balanced` profile. In addition, this profile enables scheduler autogroups for better response of interactive applications.

> **Note:**
>
> You can install additional profiles to better match your system configuration and intended use case. For example, if you are using a real-time kernel with Oracle Linux, you can use a real-time profile. These optional packages can be installed from the `ol8_addons` channel.
>
> Note that real-time profiles have no effect on kernels that are not compiled with real-time support enabled.
>
> To list all of the profiles that are currently available for installation, use the following command:
>
> ```
> sudo dnf list tuned-profiles*
> ```

Tuned profiles that are installed on the system by default are stored in the `/usr/lib/tuned` and `/etc/tuned` directories. Distribution-specific profiles are stored in the `/usr/lib/tuned` directory. Note that each profile has its own directory. Each profile directory consists of a main configuration file, `tuned.conf`, as well as other optional files.

If you want to use a custom profile, copy the profile directory to the `/etc/tuned` directory, which is the location in which custom profiles are stored. In the event there are two profiles with the same name, the custom profile that is located in `/etc/tuned/` is used.

The `tuned.conf` file can contain one `[main]` section and additional sections for configuring plugin instances. Note that these sections are optional. For more information about profile configuration, see the `tuned.conf(5)` manual page.

About the Default Tuned Profiles

A default Tuned profile is automatically selected when you install Oracle Linux. The default profile that is selected is based on the given environment and the performance goals to be achieved in that particular use case. The following default profiles are provided:

- `throughput-performance`: Is a profile that is used in an environment that compute nodes are running Oracle Linux. This profile achieves the best throughput performance.
- `virtual-guest`: Is a profile that is used in an environment that virtual machines are running Oracle Linux. This profile achieves the best performance. If you are

not interested in the best performance, you can change the profile to either the `balanced` or `powersave` profile.

- `balanced`: Is a profile that is used for other use cases. This profile achieves balanced performance and power consumption.

About Static and Dynamic Tuning in Tuned

Static tuning applies settings that you have defined in the configuration files for `sysctl`, `sysfs`, and other system configuration tools throughout the operating system.

You can configure the `tuned` service to monitor the activity of system components and dynamically tuned system settings, based on information that the service collects about the system and its current running state.

Dynamic tuning can be particularly useful in situations where you need the load on devices like the CPU, hard drives, and network adapters to consume as little power as possible when idle, but require high throughput and low latency when under a high load.

To enable dynamic tuning, set the correct value in the `/etc/tuned/tuned-main.conf` settings file as follows:

```
dynamic_tuning = 1
```

You must then set the time interval in seconds for `tuned` to analyze the current system state in the same configuration file so that it can dynamically tune the system, based on the collected results, for example:

```
update_interval = 10
```

Installing and Enabling Tuned by Using the Command Line

The following procedure describes how to install and enable Tuned, install Tuned profiles, and preset a default Tuned profile for your Oracle Linux systems.

1. If the `tuned` package is not already installed, install it:

   ```
   sudo dnf install tuned
   ```

2. Enable and start the `tuned` service:

   ```
   sudo systemctl enable --now tuned
   ```

3. Check the active Tuned profile:

   ```
   sudo tuned-adm active
   ```

4. Verify that the Tuned profile is applied to the system:

   ```
   sudo tuned-adm verify
   ```

   The output confirms that the verification succeeded:

   ```
   Verification succeeded, current system settings match the preset profile.
   See tuned log file ('/var/log/tuned/tuned.log') for details.
   ```

   If a message about the current system settings not matching is displayed, try restarting the `tuned` service:

   ```
   sudo systemctl start tuned
   ```

Running Tuned in no-daemon Mode

Running `tuned` in `no-daemon` mode does not require any resident memory. However, note that when running the service in this mode, `tuned` does not perform any dynamic tuning. While in `no-daemon` mode, `tuned` only applies the settings and then exits.

To run `tuned` in `no-daemon` mode, you must set the following value in the `/etc/tuned/tuned-main.conf` settings file:

```
daemon = 0
```

> **NOT_SUPPORTED:**
>
> Take note that if you decide to run `tuned` in `no-daemon` mode, be aware that some functions do not work without running the daemon. In particular, `tuned` no longer supports D-Bus services or the hot-plug kernel subsystem. Consequently `tuned` can no longer automatically roll back any settings files that were changed.

Administering the Tuned Service and Tuned Profiles

You administer Tuned by using the `tuned-adm` command. The following tasks describe how to administer Tuned profiles and the `tuned` service on your Oracle Linux systems.

For more information, see the `tuned-adm(8)` and `tuned(8)` manual pages.

Listing Tuned Profiles

To list all of the available Tuned profiles on a system:

```
sudo tuned-adm list

Available profiles:
- balanced                    - General non-specialized tuned profile
- desktop                     - Optimize for the desktop use-case
- latency-performance         - Optimize for deterministic performance at the
cost of increased power consumption
- network-latency             - Optimize for deterministic performance at the
cost of increased power consumption, focused on low latency network performance
- network-throughput          - Optimize for streaming network throughput,
generally only necessary on older CPUs or 40G+ networks
- powersave                   - Optimize for low power consumption
- throughput-performance      - Broadly applicable tuning that provides
excellent performance across a variety of common server workloads
- virtual-guest               - Optimize for running inside a virtual guest
- virtual-host                - Optimize for running KVM guests
Current active profile: balanced
```

Note that the current active profile is also displayed in the previous output.

To display just the currently active profile, use the following command:

```
sudo tuned-adm active
```

Activating a Tuned Profile

The following procedure describes how to activate a Tuned profile by using the command line. To activate a Tuned profile by using the Cockpit web console, see Oracle Linux: Use Cockpit to Set Up Performance Profiles.

> **Note:**
>
> To activate a Tuned profile, the `tuned` service must be running on your system.

Use the following command activate a specific selected Tuned profile:

```
sudo tuned-adm profile profile-name
```

To have Tuned recommend the profile that is most suitable for your system, use the `tuned-adm recommend` command:

```
sudo tuned-adm recommend
```

The output of the command displays the recommended profile for the system on which the command is run:

```
virtual-guest
```

To activate a combination of multiple profiles, use the following command syntax:

```
sudo tuned-adm profile profile1 profile2
```

Disabling Tuned

To disable tuning temporarily, use the following command:

```
sudo tuned-adm off
```

The previous command disables any tuning settings until you restart the `tuned` service. When you restart the service, all of the previous tuning settings are re-applied.

You can disable tuning on a more permanent basis by stopping and disabling the `tuned` service as follows:

```
sudo systemctl disable --now tuned
```

6
Automating System Tasks

You can use automated tasks to perform periodic backups, monitor the system, run custom scripts, as well as other administrative tasks.

The `cron` and `anacron` utilities enable you to schedule the execution of recurring tasks, referred to as *jobs*, according to a combination of the following: time, day of the month, month, day of the week, and week. With the `cron` utility, you can schedule jobs to run as often as every minute. If the system is down when a job is scheduled, `cron` does not run the job when the system restarts.

The `anacron` utility you to schedule a system job to run only once per day. However, if a scheduled job has not been run, that job runs when the system restarts. The `anacron` utility is mainly intended for use on laptop computers.

You do not usually need to run `cron` and `anacron` directly. The `crond` daemon executes scheduled tasks on behalf of `cron` and it starts `anacron` once every hour. `crond` looks in `/etc/crontab` or in files in `/etc/cron.d` for system `cron` job definitions, and `/var/spool/cron` for `cron` job definitions belonging to users. `crond` checks each job definition to see whether it should run in the current minute. If a job is scheduled for execution, `crond` runs it as the owner of the job definition file or, for system `cron` jobs, the user specified in the job definition (if any).

`crond` runs the `0anacron` script in the `/etc/cron.hourly` directory as `root` once per hour according to the schedule in `/etc/cron.d/0hourly`. If `anacron` is not already running and the system is connected to mains and not battery power, `crond` starts `anacron`.

`anacron` runs the scripts in the `/etc/cron.daily`, `/etc/cron.weekly`, and `/etc/cron.monthly` directories as `root` once per day, week or month, according to the job definitions that are scheduled in `/etc/anacrontab`.

Configuring cron Jobs

System `cron` jobs are defined in `crontab`-format files in `/etc/crontab` or in files in `/etc/cron.d`. A `crontab` file usually consists of definitions for the `SHELL`, `PATH`, `MAILTO`, and `HOME` variables for the environment in which the jobs run, followed by the job definitions themselves. Comment lines start with a `#` character. Job definitions are specified in the following format:

```
minute  hour  day  month  day-of-week  user  command
```

Each of the fields that you can specify are defined as follows:

minute
Specify a value of 0-59.

hour
Specify a value of 0-23.

day
Specify a value of 1-31.

month
Specify a value of 1-12 or `jan`, `feb`,..., `dec`.

day-of-week
Specify a value of 0-7 (Sunday is 0 or 7) or `sun`, `mon`,...,`sat`.

user
Specify the user running the command; or, you can specify an asterisk (`*`), which indicates the owner of the `crontab` file.

command
Specify the shell script or command to be run.

For the *minute* through *day-of week* fields, you can use the following special characters:

*****
Specify an asterisk (`*`) for all of the valid values for the field.

-
Specify a dash (`-`) to indicate a range of integers, for example, `1-5`.

,
Specify a list of values, separated by commands (`,`), for example, `0,2,4`.

/
Specify a step value by using the forward slash (`/`), for example, `/3` in the *hour* field. This entry is interpreted as every three hours.

For example, the following entry would run a command every five minutes on weekdays:

```
0-59/5  *  *  *  1-5  *  command
```

Run a command at one minute past midnight on the first day of the months April, June, September, and November:

```
1  0  1  4,6,9,11  *  *  command
```

The `root` user can add job definition entries to the `/etc/crontab`, or add `crontab`-format files to the `/etc/cron.d` directory.

> **Note:**
>
> If you add an executable job script to the `/etc/cron.hourly` directory, `crond` runs the script once every hour. Your script should check that it is not already running.

For more information, see the `crontab(5)` manual page.

Controlling Access to Running cron Jobs

If permitted, users other than `root` can configure `cron` tasks by using the `crontab` utility. All user-defined `crontab`-format files are stored in the `/var/spool/cron` directory with the same name as the users that created them.

`root` can use the `/etc/cron.allow` and `/etc/cron.deny` files to restrict access to `cron`. `crontab` checks the access control files each time that a user tries to add or delete a `cron` job. If `/etc/cron.allow` exists, only users listed in it are allowed to use `cron`, and `/etc/cron.deny` is ignored. If `/etc/cron.allow` does not exist, users listed in `/etc/cron.deny` are not allowed to use `cron`. If neither file exists, only `root` can use `cron`. The format of both `/etc/cron.allow` and `/etc/cron.deny` is one user name on each line.

To create or edit a `crontab` file as a user, log in as that user and type the command `crontab -e`, which opens your `crontab` file in the `vi` editor (or the editor specified by the `EDITOR` or `VISUAL` environment variables). The file has the same format as `/etc/crontab` except that the user field is omitted. When you save changes to the file, these are written to the file `/var/spool/cron/`*`username`* . To list the contents of your `crontab` file, use the `crontab -l` command. To delete your `crontab` file, use the `crontab -r` command.

For more information, see the `crontab(5)` manual page.

Configuring anacron Jobs

System `anacron` jobs are defined in `/etc/anacrontab`, which contains definitions for the `SHELL`, `PATH`, `MAILTO`, `RANDOM_DELAY`, and `START_HOURS_RANGE` variables for the environment in which the jobs run, followed by the job definitions themselves. Comment lines start with a `#` character.

`RANDOM_DELAY` is the maximum number of random time in minutes that `anacron` adds to the *delay* parameter for a job. The default minimum delay is 6 minutes. The random offset is intended to prevent `anacron` overloading the system with too many jobs at the same time.

`START_HOURS_RANGE` is the time range of hours during the day when anacron can run scheduled jobs.

Job definitions are specified in the following format:

`period delay job-id command`

The following is a description of the fields that may be included:

period
Frequency of job execution specified in days or as `@daily`, `@weekly`, or `@monthly` for once per day, week, or month.

delay
Number of minutes to wait before running a job.

job-id
Unique name for the job in log files.

command
The shell script or command to be run.

The following entries are taken from the default `/etc/anacrontab` file:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days  delay in minutes  job-identifier   command
1               5                 cron.daily       nice run-parts /etc/
cron.daily
7               25                cron.weekly      nice run-parts /etc/
cron.weekly
@monthly        45                cron.monthly     nice run-parts /etc/
cron.monthly
```

By default, `anacron` runs jobs between 03:00 and 22:00 and randomly delays jobs by between 11 and 50 minutes. The job scripts in `/etc/cron.daily`, run anywhere between 03:11 and 03:50 every day if the system is running, or after the system is booted and the time is less than 22:00. The `run-parts` script sequentially executes every program within the directory specified as its argument.

Scripts in `/etc/cron.weekly` run once per week with a delay offset of between 31 and 70 minutes.

Scripts in `/etc/cron.monthly` run once per week with a delay offset of between 51 and 90 minutes.

For more information, see the `anacron(8)` and `anacrontab(5)` manual pages.

Running One-Time Tasks

You can use the `at` command to schedule a one-time task to run at a specified time, or the `batch` command to schedule a one-time task to run when the system load average drops below 0.8. The `atd` service must be running to use `at` or `batch`.

```
sudo systemctl is-active atd
```

`at` takes a time as its argument and reads the commands to be run from the standard input. For example, run the commands in the file `atjob` in 20 minutes time:

```
at now + 20 minutes < ./atjob

job 1 at 2013-03-19 11:25
```

The `atq` command shows the `at` jobs that are queued to run, for example:

```
sudo atq

1    2013-03-19 11:25 a root
```

The `batch` command also reads command from the standard input, but it does not run until the system load average drops below 0.8, for example:

```
sudo batch < batchjob

job 2 at 2013-03-19 11:31
```

To cancel one or more queued jobs, specify their job numbers to the `atrm` command, for example:

```
sudo atrm 1 2
```

For more information, see the `at(1)` manual page.

Changing the Behavior of Batch Jobs

The load average of a system, as displayed by the `uptime` and `w` commands, represents the average number of processes that are queued to run on the CPUs or CPU cores over a given time period. Typically, a system might not considered overloaded until the load average exceeds 0.8 times the number of CPUs or CPU cores. On such systems, you would usually want `atd` to be able to run batch jobs when the load average drops below the number of CPUs or CPU cores, rather than the default limit of 0.8. For example, on a system with 4 CPU cores, you could set the load-average limit above which `atd` will not run batch jobs to 3.2.

If you know that a batch job typically takes more than a minute to run, you can also change the minimum interval that `atd` waits between starting batch jobs. The default minimum interval is 60 seconds.

To change the load-average limit and minimum interval time for batch jobs:

1. Edit the `atd` configuration file, `/etc/sysconfig/atd`, uncomment the line that defines the `OPTS` variable, and edit the line to specify the new load-average limit and minimum interval time, for example:

   ```
   OPTS="-b 100 -l 3"
   ```

   This example sets the minimum interval to 100 seconds and the load-average limit to 3.

2. Restart the `atd` service:

   ```
   sudo systemctl restart atd
   ```

3. Verify that the `atd` daemon is running with the new minimum interval and load-average limit, for example:

   ```
   sudo systemctl status atd

   atd.service - Job spooling tools
      Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled)
      Active: active (running) since Mon 2014-04-28 15:37:04 BST; 2min 53s ago
    Main PID: 6731 (atd)
      CGroup: /system.slice/atd.service
              └─6731 /usr/sbin/atd -f -b 100 -l 3

   Apr 28 15:37:04 localhost.localdomain systemd[1]: Started Job spooling tools.
   ```

For more information, see the `systemctl(1)` and `atd(8)` manual pages.

7

Configuring and Using Auditing

Auditing collects data at the kernel level that you can then analyze to identify unauthorized activity. Auditing collects data in greater detail than system logging does; but, note that most audited events are uninteresting and insignificant. Because the process of examining audit trails to locate events of interest can be a significantly challenging, you may consider automating this process.

The audit configuration file, `/etc/audit/auditd.conf`, defines the following:

- Data retention policy
- Maximum size of the audit volume
- Action to take if the capacity of the audit volume is exceeded
- Locations of local and remote audit trail volumes

The default audit trail volume is `/var/log/audit/audit.log`. See the `auditd.conf(5)` manual page for more information.

By default, auditing captures specific events such as system logins, modifications to accounts, and `sudo` actions. You can alsoA configure auditing to capture detailed system call activity or modifications to certain files. The kernel audit daemon (`auditd`) records the events that you configure, including the event type, a time stamp, the associated user ID, and success or failure of the system call.

The entries in the audit rules file, `/etc/audit/audit.rules`, determine which events are audited. Each rule is a command-line option that is passed to the `auditctl` command. You should typically configure this file to match your site's security policy.

The following are examples of rules that you might set in the `/etc/audit/audit.rules` file:

Record all unsuccessful exits from `open` and `truncate` system calls for files in the `/etc` directory hierarchy.

```
-a exit,always -S open -S truncate -F /etc -F success=0
```

To record all files opened by a user with UID 10:

```
-a exit,always -S open -F uid=10
```

To record all files that have been written to or files with their attributes changed by any user who originally logged in with a UID of 500 or greater:

```
-a exit,always -S open -F auid>=500 -F perm=wa
```

To record requests for write or file attribute change access to the `/etc/sudoers` file and tag such a record with the string `sudoers-change`:

```
-w /etc/sudoers -p wa -k sudoers-change
```

To record requests for write and file attribute change access to the `/etc` directory hierarchy:

```
-w /etc/ -p wa
```

To require a reboot after changing the audit configuration:

```
-e 2
```

If specified, this rule should appear at the end of the `/etc/audit/audit.rules` file.

For more examples of audit rules, refer to the `/usr/share/doc/audit-version/stig.rules` file. See also the `auditctl(8)` and `audit.rules(7)` manual pages.

Stringent auditing requirements can impose a significant performance overhead and generate large amounts of audit data. Some site security policies stipulate that a system must shut down if events cannot be recorded because the audit volumes have exceeded their capacity. As a general rule, you should direct audit data to separate file systems in rotation to prevent overspill and to facilitate backups.

You can use the `-k` option to tag audit records so that you can locate them more easily in an audit volume with the `ausearch` command. For example, to examine records tagged with the string `sudoers-change`, you would enter:

```
sudo ausearch -k sudoers-change
```

The `aureport` command generates summaries of audit data. You can set up `cron` jobs that run `aureport` periodically to generate reports of interest. For example, the following command generates a reports that shows every login event from 1 second after midnight on the previous day until the current time:

```
sudo aureport -l -i -ts yesterday -te now
```

See the `ausearch(8)` and `aureport(8)` manual pages for more information.

See https://docs.oracle.com/en/learn/ol-auditd/ for a hands-on tutorial on using the auditing tools on Oracle Linux.

Working With System Log files

The log files contain messages about the system, kernel, services, and applications. The `journald` logging daemon, which is part of `systemd`, records system messages in non-persistent journal files in memory and in the `/run/log/journal` directory. `journald` forwards messages to the system logging daemon, `rsyslog`. As files in `/run` are volatile, the log data is lost after a reboot unless you create the directory `/var/log/journal`. You can use the `journalctl` command to query the journal logs.

For more information, see the `journalctl(1)` and `systemd-journald.service(8)` manual pages.

See https://docs.oracle.com/en/learn/system_logging_linux8/ for a hands-on tutorial introducing system logging tools.

About Logging Configuration (/etc/rsyslog.conf)

The configuration file for `rsyslogd` is `/etc/rsyslog.conf`, which contains global directives, module directives, and rules. By default, `rsyslog` processes and archives only `syslog` messages. If required, you can configure `rsyslog` to archive any other messages that `journald` forwards, including kernel, boot, `initrd`, `stdout`, and `stderr` messages.

Global directives specify configuration options that apply to the `rsyslogd` daemon. All configuration directives must start with a dollar sign (`$`) and only one directive can be specified on each line. The following example specifies the maximum size of the `rsyslog` message queue:

```
$MainMsgQueueSize 50000
```

The available configuration directives are described in the file `/usr/share/doc/rsyslog-version-number/rsyslog_conf_global.html`.

The design of `rsyslog` ensures that functionality is dynamically loaded from modules, which provide configuration directives. To load a module, specify the following directive:

```
$ModLoad MODULE_name
```

Modules have the following main categories:

- Input modules gather messages from various sources. Input module names always start with the `im` prefix (examples include `imfile` and `imrelp`).
- Filter modules enable `rsyslogd` to filter messages according to specified rules. The name of a filter module always starts with the `fm` prefix.
- Library modules provide functionality for other loadable modules. `rsyslogd` loads library modules automatically when required. You cannot configure the loading of library modules.
- Output modules provide the facility to store messages in a database or on other servers in a network, or to encrypt them. Output module names always starts with the `om` prefix (examples include `omsnmp` and `omrelp`).
- Message modification modules change the content of an `rsyslog` message.
- Parser modules enable `rsyslogd` to parse the message content of messages that it receives. The name of a parser module always starts with the `pm` prefix.
- String generator modules generate strings based on the content of messages in cooperation with `rsyslog`'s template feature. The name of a string generator module always starts with the `sm` prefix.

Input modules receive messages, which pass them to one or more parser modules. A parser module creates a representation of a message in memory, possibly modifying the message, and passes the internal representation to output modules, which can also modify the content before outputting the message.

A description of the available modules can be found at https://www.rsyslog.com/doc/rsyslog_conf_modules.html.

An `rsyslog` rule consists of a filter part, which selects a subset of messages, and an action part, which specifies what to do with the selected messages. To define a rule in the `/etc/rsyslog.conf` configuration file, specify a filter and an action on a single line, separated by one or more tabs or spaces.

You can configure `rsyslog` to filter messages according to various properties. The following are the most commonly used filters:

- Expression-based filters, written in the `rsyslog` scripting language, select messages according to arithmetic, boolean, or string values.

- Facility/priority-based filters filter messages based on facility and priority values that take the form *facility.priority*.
- Property-based filters filter messages by properties such as `timegenerated` or `syslogtag`.

The following table describes the available facility keywords for facility or priority-based filters.

Facility Keyword	Description
`auth`, `authpriv`	Security, authentication, or authorization messages.
`cron`	`crond` messages.
`daemon`	Messages from system daemons other than `crond` and `rsyslogd`.
`kern`	Kernel messages.
`lpr`	Line printer subsystem.
`mail`	Mail system.
`news`	Network news subsystem.
`syslog`	Messages generated internally by `rsyslogd`.
`user`	User-level messages.
`UUCP`	UUCP subsystem.
`local0 - local7`	Local use.

The following table describes the available priority keywords for facility or priority-based filters, in ascending order of importance.

Priority Keyword	Description
`debug`	Debug-level messages.
`info`	Informational messages.
`notice`	Normal but significant condition.
`warning`	Warning conditions.
`err`	Error conditions.
`crit`	Critical conditions.
`alert`	Immediate action required.
`emerg`	System is unstable.

All messages of the specified priority and higher are logged according to the specified action. An asterisk (`*`) wildcard specifies all facilities or priorities. Separate the names of multiple facilities and priorities on a line with commas (`,`). Separate multiple filters on one line with semicolons (;). Precede a priority with an exclamation mark (`!`) to select all messages except those with that priority.

The following are examples of facility/priority-based filters.

Select all kernel messages with any priority as follows:

```
kern.*
```

Select all mail messages with `crit` or higher priority as follows:

```
mail.crit
```

Select all `daemon` and `kern` messages with `warning` or `err` priority as follows:

```
daemon,kern.warning,err
```

Select all `cron` messages except those with `info` or `debug` priority as follows:

```
cron.!info,!debug
```

By default, `/etc/rsyslog.conf` includes the following rules:

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog

# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

You can send the logs to a central log server over TCP by adding the following entry to the `forwarding rules` section of `/etc/rsyslog.conf` on each log client:

```
*.*        @@logsvr:port
```

In the previous example, *logsvr* is the domain name or IP address of the log server and port is the port number (usually, 514).

On the log server, add the following entry to the `MODULES` section of `/etc/rsyslog.conf`:

```
$ModLoad imtcp
$InputTCPServerRun port
```

In the previous example, *port* corresponds to the port number that you set on the log clients.

To manage the rotation and archival of the correct logs, edit `/etc/logrotate.d/syslog` so that it references each of the log files that are defined in the `RULES` section of `/etc/rsyslog.conf`. You can configure how often the logs are rotated and how many past copies of the logs are archived by editing `/etc/logrotate.conf`.

It is recommended that you configure Logwatch on your log server to monitor the logs for suspicious messages, and disable Logwatch on log clients. However, if you do use Logwatch, disable high precision timestamps by adding the following entry to the GLOBAL DIRECTIVES section of `/etc/rsyslog.conf` on each system:

```
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
```

For more information, see the `logrotate(8)`, `logwatch(8)`, `rsyslogd(8)` and `rsyslog.conf(5)` manual pages. See also the HTML documentation in the `/usr/share/doc/rsyslog-5.8.10` directory and the documentation at https://www.rsyslog.com/doc/manual.html.

Configuring Logwatch

Logwatch is a monitoring system that you can configure to report on areas of interest in the system logs. After you install the `logwatch` package, the `/etc/cron.daily/0logwatch` script runs every night and sends an email report to `root`. You can set local configuration options in `/etc/logwatch/conf/logwatch.conf` that override the main configuration file `/usr/share/logwatch/default.conf/logwatch.conf`, including the following:

- Log files to monitor, including log files that are stored for other hosts.
- Names of services to monitor, or to be excluded from monitoring.
- Level of detail to report.
- User to be sent an emailed report.

You can also run `logwatch` directly from the command line.

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

Using Process Accounting

The `psacct` package implements the process accounting service in addition to the following utilities that you can use to monitor process activities:

`ac`
Displays connection times in hours for a user as recorded in the `wtmp` file (by default, `/var/log/wtmp`).

`accton`
Turns on process accounting to the specified file. If you do not specify a file name argument, process accounting is stopped. The default system accounting file is `/var/account/pacct`.

`lastcomm`
Displays information about previously executed commands as recorded in the system accounting file.

`sa`
Summarizes information about previously executed commands as recorded in the system accounting file.

> **Note:**
>
> As for any logging activity, ensure that the file system has enough space to store the system accounting and `wtmp` files. Monitor the size of the files and, if necessary, truncate them.

For more information, see the `ac(1)`, `accton(8)`, `lastcomm(1)`, and `sa(8)` manual pages.

8
Working With Kernel Dumps

The Kdump feature provides a kernel crash dumping mechanism in Oracle Linux. The `kdump` service enables you to save the contents of the system's memory for later analysis. The second kernel resides in a reserved part of the system memory.

Kdump uses the `kexec` system call to boot into the second kernel, called a *capture kernel*, without the need to reboot the system, and then captures the contents of the crashed kernel's memory as a crash dump (`vmcore`) and saves it. The `vmcore` crash dump can help with determining the cause of the crash.

Oracle recommends that you enable the Kdump feature because a crash dump might be the only information that is available if a system failure occurs. Ensuring that Kdump is enabled is critical in many mission-critical environments.

Prior to enabling Kdump, ensure that your system meets all of the memory requirements for using Kdump. To capture a kernel crash dump and save it for further analysis, you must permanently reserve part of the system's memory for the capture kernel. Note that this part of the system's memory will no longer be available to the main kernel. The following table lists the minimum amount of reserved memory that is required to use Kdump, based on the system's architecture and the amount of available memory.

Architecture	Available Memory	Minimum Reserved Memory
`x86_64`	1 GB to 64 GB	160 MB of RAM
	64 GB to 1 TB	256 MB of RAM
	1 TB and more	512 MB of RAM
Arm (aarch64)	2 TB and more	512 MB of RAM

Kdump Installation and Configuration

This section describes how to install and configure Kdump by using the command line.

For information about configuring Kdump by using the Cockpit web console, see Oracle Linux: Use Cockpit to Configure Kdump

Files That Are Used by Kdump

When you install and configure Kdump, the following files are modified:

`/boot/grub2/grub.cfg`
Appends the `crashkernel` option to the kernel line to specify the amount of reserved memory and any offset value.

/etc/kdump.conf
Sets the location in which 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.

When you edit these files, you must reboot the system for the changes to take effect.

For more information, see the `kdump.conf(5)` manual page.

Installing and Configuring Kdump

During an Oracle Linux interactive installation with the graphical installer, you have the option to enable Kdump and specify how much system memory is reserved for Kdump. The installer screen is titled **Kdump** and is available from the main Installation Summary screen of the installer.

If you do not enable Kdump at installation time, or it is not enabled by default during an installation, as in the case of a custom kickstart installation, you can install and enable the feature by using the command line.

Before you install and configure Kdump by using the command line, ensure that your system meets all of the necessary memory requirements. For details, see Working With Kernel Dumps.

1. If the `kdump` package is not already installed on your system, install it:

   ```
   sudo dnf install kexec-tools
   ```

2. As the `root` user, edit the `/etc/default/grub` file and set the `crashkernel=` option to the required value.

   For example, you would reserve 64 MB of memory as follows:

   ```
   crashkernel=64M
   ```

   You can also set the amount of reserved memory as a variable by using the following syntax: `crashkernel=`*`range1`*`:`*`size1`*`,`*`range2`*`:`*`size2`*

   For example, you might set the memory as a variable as follows:

   ```
   crashkernel=512M-2G:64M,2G-:128M
   ```

3. (Optional) If necessary, offset the reserved memory.

   Because the crashkernel reservation occurs very early, some systems require that you reserve memory with a certain fixed offset. When a fixed offset is specified, the reserved memory begins at that point. For example, you would reserve 128 MB of memory, starting at 16 MB as follows:

   ```
   crashkernel=128M@16M
   ```

   Note that if no offset parameter is set, Kdump offsets reserved memory automatically.

4. Refresh the grub configuration to apply your changes:

   ```
   sudo grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

5. Reboot the system and finish configuring Kdump.

   For instructions, see Configuring the Kdump Output Location.

6. When you have finished configuring Kdump, enable the `kdump` service:

```
sudo systemctl enable --now kdump.service
```

Configuring the Kdump Output Location

After installing Kdump, you can define the location in which the resulting output should be saved. For Oracle Linux, Kdump files are stored in the `/var/crash` directory by default.

To save the result to other locations, such as NFS mounts, externally mounted drives, and remote file servers, edit the `/etc/kdump.conf` file and remove the `#` comment character at the beginning of each line that you want to enable.

For example, to add a new directory location, prefix it with the `path` keyword:

```
path /usr/local/cores
```

Use `raw` to output directly to a specific device in the `/dev` directory. You can also manually specify the output file system for a particular device by using its label, name or UUID, for example:

```
ext4 UUID=5b065be6-9ce0-4154-8bf3-b7c4c7dc7365
```

Kdump files can also be transferred over a secure shell connection, as shown in the following example:

```
ssh user@example.com
sshkey /root/.ssh/mykey
```

It is also possible to export the result to a compatible network share:

```
nfs example.com:/output
```

When you have finished configuring the output location for Kdump, enable the `kdump` service as follows:

```
sudo systemctl enable --now kdump.service
```

Configuring the Default Kdump Failure State

By default, if `kdump` fails to output its result to the configured outlook locations, it reboots the server. This action deletes any data that has been collected for the dump, so you should uncomment and change the `default` value in the `/etc/kdump.conf` file as follows:

```
default dump_to_rootfs
```

The `dump_to_rootfs` option attempts to save the result to a local directory, which can be particularly useful if a network share is unreachable. Using `shell` instead enables you to copy the data manually from the command line.

> **Note:**
>
> The `poweroff`, `restart`, and `halt`options are also valid for the default `kdump` failure state; but, you will lose the collected data if those actions are performed.

Analyzing Kdump Output

The `crash` utility provides a shell prompt that enables you to analyze the contents of your `kdump` core dumps, which is particularly useful when troubleshooting problems.

1. If the `crash` package is not installed on your system, install it:

   ```
   sudo dnf install crash
   ```

2. Identify the currently running kernel, for example:

   ```
   uname -r
   ```

   The output of the previous command is similar to the following:

   ```
   4.18.0-80.el8.x86_64
   ```

3. Provide the location of the kernel debuginfo module and the location of the core dump as parameters to the `crash` utility, for example:

   ```
   sudo crash /usr/lib/debug/lib/modules/4.18.0-80.el8.x86_64/vmlinux \
    /var/crash/127.0.0.1-2019-10-28-12:38:25/vmcore
   ```

   In the previous command, `4.18.0-80.el8.x86_64` is the currently running kernel and `127.0.0.1-2019-10-28-12:38:25` represents the *ipaddress-timestamp*.

4. Inside the `crash` shell, you can use the `help log` command to better understand how to use the `log` command.

   You can also use the `bt`, `ps`, `vm`, and `files` commands to get more information about the core dump.

5. When you have finished analyzing the core dump, exit the shell.

For more detailed information about using the `crash` utility, see the `crash(8)` manual page.

Using Early Kdump

New in Oracle Linux 8, early Kdump enables the crash kernel and initramfs to load early enough to capture vmcore information for early crashes.

Because the `kdump` service starts too late, early crashes are not able to use normal `kdump` kernel booting. As a result, information about early crashes is lost. To address this issue, you can enable early Kdump by adding a dracut module so that the crash kernel and initramfs are loaded as early as possible. When early Kdump is enabled, the files are loaded just like a normal `kdump`, which is disabled by default.

Note that early Kdump does not support Fadump currently.

For more information about configuring early Kdump on your Oracle Linux 8 systems, see the step-by-step instructions in the `/usr/share/doc/kexec-tools/early-kdump-howto.txt` file.

Using Kdump with OCFS2

By default, a fenced node in an OCFS2 cluster restarts instead of panicking so that it can quickly rejoin the cluster. If the reason for the restart is not apparent, you can change the node's behavior so that it panics and generates a `vmcore` for analysis.

To configure a node to panic when it next fences, run the following command on the node after the cluster starts:

```
echo panic > /sys/kernel/config/cluster/cluster_name/fence_method
```

In the previous command, *cluster_name* is the name of the cluster. To set the value after each reboot of the system, add this line to `/etc/rc.local`. To restore the default behavior, set the value of `fence_method` to `reset` instead of! `panic` and remove the line from `/etc/rc.local`.

For more information, see Oracle® Linux 8: Setting Up High Availability Clustering.