Red Hat Enterprise Linux for Real Time 9

Installing RHEL 9 for Real Time

Installing the RHEL for Real Time kernel on Red Hat Enterprise Linux

Last Updated: 2023-12-05
Installing the RHEL for Real Time kernel on Red Hat Enterprise Linux
Abstract

Install the RHEL for Real Time kernel on Red Hat Enterprise Linux to obtain low latency and make response times predictable. Learn how to install the real-time kernel, perform post-installation tasks, and configure a kernel variant, such as the real-time, stock, or the debug kernel to boot with.
# Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE ................................................................. 3

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION ........................................ 4

CHAPTER 1. INSTALLING RHEL FOR REAL TIME ................................................... 5
  1.1. RHEL FOR REAL TIME FOR OPTIMIZING LATENCY ................................. 5
  1.2. INSTALLING RHEL FOR REAL TIME USING DNF ................................. 6
  1.3. AVAILABLE RPM PACKAGES IN THE RHEL FOR REAL TIME REPOSITORY 8
  1.4. POST INSTALLATION INSTRUCTIONS ....................................................... 9

CHAPTER 2. SPECIFYING THE RHEL KERNEL TO RUN ........................................ 11
  2.1. DISPLAYING THE DEFAULT KERNEL ...................................................... 11
  2.2. DISPLAYING THE RUNNING KERNEL ...................................................... 11
  2.3. CONFIGURING KERNEL-RT AS THE DEFAULT BOOT KERNEL ................. 11

CHAPTER 3. INSTALLING KDUMP ..................................................................... 13
  3.1. WHAT IS KDUMP ................................................................................. 13
  3.2. INSTALLING KDUMP USING ANACONDA ............................................. 13
  3.3. INSTALLING KDUMP ON THE COMMAND LINE .................................... 14

CHAPTER 4. CONFIGURING KDUMP ON THE COMMAND LINE ............................... 15
  4.1. ESTIMATING THE KDUMP SIZE ......................................................... 15
  4.2. CONFIGURING KDUMP MEMORY USAGE ........................................... 15
  4.3. CONFIGURING THE KDUMP TARGET .................................................. 17
  4.4. CONFIGURING THE KDUMP CORE COLLECTOR ................................. 19
  4.5. CONFIGURING THE KDUMP DEFAULT FAILURE RESPONSES ............. 20
  4.6. TESTING THE KDUMP CONFIGURATION ............................................ 20

CHAPTER 5. ENABLING KDUMP ..................................................................... 22
  5.1. ENABLING KDUMP FOR ALL INSTALLED KERNELS ............................... 22
  5.2. ENABLING KDUMP FOR A SPECIFIC INSTALLED KERNEL ..................... 22
  5.3. DISABLING THE KDUMP SERVICE ...................................................... 23

CHAPTER 6. REPORTING RHEL FOR REAL TIME BUGS ........................................ 25
  6.1. DIAGNOSING RHEL FOR REAL TIME BUGS ........................................... 25
  6.2. SUBMITTING A BUG REPORT WITH BUGZILLA ..................................... 25
Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the Jira website.

2. Click Create in the top navigation bar

3. Enter a descriptive title in the Summary field.

4. Enter your suggestion for improvement in the Description field. Include links to the relevant parts of the documentation.

5. Click Create at the bottom of the dialogue.
CHAPTER 1. INSTALLING RHEL FOR REAL TIME

Many industries and organizations need extremely high performance computing and may require low and predictable latency, especially in the financial and telecommunications industries. Latency, or response time, is defined as the time between an event and system response and is generally measured in microseconds ($\mu s$).

For most applications running under a Linux environment, basic performance tuning can improve latency sufficiently. For those industries where latency not only needs to be low, but also accountable and predictable, Red Hat developed a ‘drop-in’ kernel replacement that provides this. RHEL for Real Time is distributed as part of RHEL 9 and provides seamless integration with RHEL 9. RHEL for Real Time offers clients the opportunity to measure, configure, and record latency times within their organization.

WARNING
Before installing RHEL for Real Time, ensure that the base platform is properly tuned and the system BIOS parameters are adjusted. Failure to perform these tasks may prevent getting consistent performance from a RHEL Real Time deployment.

1.1. RHEL FOR REAL TIME FOR OPTIMIZING LATENCY

RHEL for Real Time is designed to be used on well-tuned systems for applications with extremely high determinism requirements. Kernel system tuning offers the vast majority of the improvement in determinism.

For example, in many workloads, thorough system tuning improves consistency of results by around 90%. This is why, before using RHEL for Real Time, we recommend that customers first perform system tuning of standard RHEL to see if it meets their objectives.

System tuning is just as important when using the Real Time kernel as it is for the standard kernel. Installing the Real Time kernel on an untuned system running the standard kernel supplied as part of the RHEL 9 release is not likely to result in any noticeable benefit. Tuning the standard kernel will yield 90% of the possible latency gains. The Real Time kernel provides the last 10% of latency reduction required by the most demanding workloads.

WARNING
Before tuning Real Time kernel systems, ensure that the base platform is properly tuned and the system BIOS parameters are adjusted. Failure to perform these tasks may prevent getting consistent performance from a RHEL Real Time deployment.

The objective of the Real Time kernel consistent, low-latency determinism offering predictable response times. There is some additional kernel overhead associated with the real time kernel. This is due primarily to handling hardware interrupts in separately scheduled threads. The increased overhead
in some workloads results in some degradation in overall throughput. The exact amount is very workload
dependent, ranging from 0% to 30%.

For typical workloads with kernel latency requirements in the millisecond (ms) range, the standard RHEL
9 kernel is sufficient. However, if your workload has stringent low-latency determinism requirements for
core kernel features such as interrupt handling and process scheduling in the microsecond (μs) range,
then the Real Time kernel is for you.

This graph compares a million samples of machines that use the RHEL 9 and the RHEL for Real Time
kernel, respectively.

- The blue points in this graph represent the system response time (in microseconds) of machines
  running a tuned RHEL 9 kernel.
- The green points in the graph represent the system response time of machines running a tuned
  real-time kernel.

It is clear from this graph that the response time of the Real Time kernel is very consistent, in contrast to
the standard kernel, which has greater variability.

1.2. INSTALLING RHEL FOR REAL TIME USING DNF

In addition to installing the real-time kernel with `dnf`, an ISO image containing RHEL for Real Time is also
available for download from the Download Red Hat Enterprise Linux portal. You can use this ISO image
to obtain all the RPM packages included with RHEL for Real Time. However, because this is not a
bootable ISO image, you cannot use it to create a bootable USB or CD medium.

Prerequisites

- The latest version of RHEL 9 is installed on an AMD64 or Intel64 system. The real-time kernel
  runs on AMD64 and Intel 64 (also known as x86_64) server platforms that are certified to run
  Red Hat Enterprise Linux.
- Your machine is registered and RHEL is attached to a RHEL for Real Time subscription.
- Ensure that the base platform is properly tuned and the system BIOS parameters are adjusted.
NOTE

Failure to perform the prerequisite tasks before installing the real-time kernel might prevent a consistent performance from a RHEL for Real Time kernel deployment.

Procedure

1. Enable the RHEL for Real Time repository.

   ```bash
   # subscription-manager repos --enable rhel-9-for-x86_64-rt-rpms
   ```

2. Install the RHEL for Real Time package group.

   ```bash
   # dnf groupinstall RT
   ```

   This group installs several packages:

   - `kernel-rt` includes the RHEL for Real Time kernel package.
   - `kernel-rt-core` includes the core RHEL for Real Time kernel package.
   - `kernel-rt-devel` includes the RHEL for Real Time kernel development package.
   - `kernel-rt-modules` includes the RHEL for Real Time kernel modules package.
   - `kernel-rt-modules-core` includes the kernel modules for the core kernel package.
   - `kernel-rt-modules-extra` includes the RHEL for Real Time kernel extra modules package.
   - `realtime-setup` sets up the basic environment required by RHEL for Real Time.
   - `rteval` evaluates system suitability for RHEL for Real Time.
   - `rteval-loads` includes the source code for `rteval` loads.
   - `tuned-profiles-realtime` includes the additional TuneD profiles targeted to real-time.

3. (Optional) Additionally, the `tuna` package contains a tool that helps tune the real-time kernel workload, greatly automating CPU isolation and thread affinity operations from the command line or the GUI. This package is available in the base RHEL 9 repository.

   ```bash
   # dnf install tuna
   ```

NOTE

When the RHEL for Real Time kernel is installed, it is automatically set to be the default kernel and is used on the next boot. You can also configure other existing kernels variants, such as, `kernel`, `kernel-debug`, or `kernel-rt-debug` as the default boot kernel. For more information, see Configuring kernel-rt as the default boot kernel.

Verification steps

- Check the installation location and verify that the components have been installed successfully.
Additional resources

- Can KVM guests be run on real-time(RT) kernel?

1.3. AVAILABLE RPM PACKAGES IN THE RHEL FOR REAL TIME REPOSITORY

The Red Hat Package Manager (RPM) for RHEL for Real Time repository includes the following packages:

- **kernel-rt** package, which is the RHEL for Real Time kernel package.
- RHEL for Real Time kernel test packages, which contains test programs for the real-time kernel.
- RHEL for Real Time debugging packages, which are for debugging and code tracing.

Table 1.1. Basic RHEL for Real Time kernel packages

<table>
<thead>
<tr>
<th>RPM package name</th>
<th>Description</th>
<th>RT-specific</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel-rt</td>
<td>Low latency and preemption functionality</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1.2. RHEL for Real Time kernel test packages

<table>
<thead>
<tr>
<th>RPM package name</th>
<th>Description</th>
<th>RT-specific</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel-rt-devel</td>
<td>Headers and libraries for kernel development</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>kernel-rt-debug</td>
<td>RHEL for Real Time kernel with debugging functions compiled in (slow)</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
The debugging packages are provided to use with the perf, trace-cmd, and crash utilities for analyzing kernel crash dumps. The debugging packages include symbol tables and are quite large. For this reason, the debugging packages are separately delivered from the other RHEL for Real Time packages. You can download the debugging packages from RHEL for Real Time - Debug RPMs repository.

Table 1.3. Basic RHEL for Real Time debugging packages

<table>
<thead>
<tr>
<th>RPM package name</th>
<th>Description</th>
<th>RT-specific</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel-rt-debug-devel</td>
<td>Headers and libraries for development on debug kernel</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>realtime-tests</td>
<td>Utilities for measuring system latencies and for proving that priority-inheritance mutex functions properly</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

1.4. POST INSTALLATION INSTRUCTIONS

After you install the real-time kernel ensure that:

- To achieve optimal low-latency determinism, you perform RHEL for Real Time specific system tuning.
- You know about the module compatibility of the real-time kernel and the standard kernel.
- To enable kdump, you must configure RHEL for Real Time to provide crash dump information by enabling kexec/kdump.
- Verify that the Real Time kernel is the default kernel.

Module Compatibility of the Real Time Kernel and the Standard Kernel

The real-time kernel differs substantially from the standard Red Hat Enterprise Linux 9 kernel. As a consequence, third-party kernel modules are incompatible with RHEL for Real Time.

Kernel modules are inherently specific to the kernel they are built for. The real time kernel is substantially different from the standard kernel, and so are the modules. Therefore, you cannot take third-party modules from Red Hat Enterprise Linux 9 and use them as-is on the real-time kernel. If you must use a
third-party module, you must recompile it with the RHEL for Real Time header files, which are available in the RHEL for Real Time development and test packages. The third-party drivers that ship for the standard Red Hat Enterprise Linux 9 but do not currently have a custom build for the real-time kernel are:

- EMC Powerpath
- NVidia graphics
- Advanced storage adapter configuration utilities from Qlogic

The user space syscall interface is compatible with RHEL for Real Time.
CHAPTER 2. SPECIFYING THE RHEL KERNEL TO RUN

You can boot any installed kernel, standard or Real Time. You can select the required kernel manually in the GRUB menu during booting. You can also configure which kernel boot by default.

When the real-time kernel is installed, it is automatically set to be the default kernel and is used on the next boot.

2.1. DISPLAYING THE DEFAULT KERNEL

You can display the kernel configured to boot by default.

Procedure

- To view the default kernel:

  ```
  ~]# grubby --default-kernel
  /boot/vmlinuz-4.18.0-80.rt9.138.el8.x86_64
  ```

  The `rt` in the output of the command shows that the default kernel is a real time kernel.

2.2. DISPLAYING THE RUNNING KERNEL

You can display the currently running kernel

Procedure

- To show which kernel the system is currently running.

  ```
  ~]# uname -a
  Linux rt-server.example.com 4.18.0-80.rt9.138.el8.x86_64 ...
  ```

  **NOTE**

  When the system receives a minor update, for example, from 8.3 to 8.4, the default kernel might automatically change from the Real Time kernel back to the standard kernel.

2.3. CONFIGURING KERNEL-RT AS THE DEFAULT BOOT KERNEL

On a newly installed system, the stock RHEL kernel is set as the default boot kernel and is used as the default kernel on the next boot and subsequent system updates. You can change this configuration and set kernel-rt as the default kernel to boot with and also make this configuration persistent across the system updates. Configuring kernel-rt is a one-time procedure, which you can change or revert to another kernel if necessary. You can also configure other existing kernels variants, such as, kernel, kernel-debug, or kernel-rt-debug, as the default boot kernel.

Procedure

1. To configure kernel-rt as the default boot kernel, enter the following command:

  ```
  # grubby --set-default=<RT_VMLINUZ>
  ```
RT_VMLINUZ is the name of the vmlinux file that is associated with the kernel-rt kernel. For example:

```
# grubby --set-default=/boot/vmlinuz-5.14.0-284.11.1.rt14.296.el9_2.x86_64+rt
```

2. To configure kernel-rt as default boot kernel on system updates, enter the following command:

```
# sed -i 's/UPDATEDEFAULT=.*/UPDATEDEFAULT=yes/g'/etc/sysconfig/kernel
# sed -i 's/DEFAULTKERNEL=.*/DEFAULTKERNEL=kernel-rt-core/g'/etc/sysconfig/kernel
```

The UPDATEDDEFAULT variable when specified as yes, sets the default kernel to change with system updates.

In the example output, the path for the default kernel is specific to the kernel-rt-core package installed. You can determine the path to the kernel from a package by using the `rpm -q kernel-rt-core` command.

a. Optional: If you need to determine the path to the kernel from a package, first list the installed packages:

```
# rpm -q kernel-rt-core
```

```
kernel-rt-core-5.14.0-284.11.1.rt14.296.el9_2.x86_64
kernel-rt-core-5.14.0-284.10.1.rt14.295.el9_2.x86_64
kernel-rt-core-5.14.0-284.9.1.rt14.294.el9_2.x86_64
```

b. To use the latest installed package as the default, enter the following command to find the path to the boot image from that package:

```
# rpm -ql kernel-rt-core-5.14.0-284.11.1.rt14.296.el9_2.x86_64|grep '^/boot/vmlinu'
```

```
/boot/vmlinuz-5.14.0-284.11.1.rt14.296.el9_2.x86_64.x86_64+rt
```

c. To configure kernel-rt as the default boot kernel, enter the following command:

```
# grubby --set-default=/boot/vmlinuz-5.14.0-284.11.1.rt14.296.el9_2.x86_64.x86_64+rt
```

Verification

- To verify kernel-rt is the default kernel, enter the following command:

```
# grubby --default-kernel
```

```
/boot/vmlinuz-5.14.0-284.11.1.rt14.296.el9_2.x86_64.x86_64+rt
```
CHAPTER 3. INSTALLING KDUMP

The *kdump* service is installed and activated by default on the new Red Hat Enterprise Linux installations. Learn about *kdump* and how to install *kdump* when it is not enabled by default.

3.1. WHAT IS KDUMP

*kdump* is a service which provides a crash dumping mechanism. The service enables you to save the contents of the system memory for analysis. *kdump* uses the *kexec* system call to boot into the second kernel (a *capture kernel*) without rebooting; and then captures the contents of the crashed kernel’s memory (a *crash dump* or a *vmcore*) and saves it into a file. The second kernel resides in a reserved part of the system memory.

**IMPORTANT**

A kernel crash dump can be the only information available in the event of a system failure (a critical bug). Therefore, operational *kdump* is important in mission-critical environments. Red Hat advise that system administrators regularly update and test *kexec-tools* in your normal kernel update cycle. This is especially important when new kernel features are implemented.

You can enable *kdump* for all installed kernels on a machine or only for specified kernels. This is useful when there are multiple kernels used on a machine, some of which are stable enough that there is no concern that they could crash.

When *kdump* is installed, a default `/etc/kdump.conf` file is created. The file includes the default minimum *kdump* configuration. You can edit this file to customize the *kdump* configuration, but it is not required.

3.2. INSTALLING KDUMP USING ANACONDA

The Anaconda installer provides a graphical interface screen for *kdump* configuration during an interactive installation. The installer screen is titled as KDUMP and is available from the main Installation Summary screen. You can enable *kdump* and reserve the required amount of memory.

**Procedure**

1. Go to the Kdump field.

2. Enable *kdump* if not already enabled.
3. Define how much memory should be reserved for **kdump**.

![Kdump Configuration Interface](image)

### 3.3. INSTALLING KDUMP ON THE COMMAND LINE

Some installation options, such as custom Kickstart installations, in some cases do **not** install or enable **kdump** by default. If this is your case, follow the procedure below.

**Prerequisites**

- An active RHEL subscription
- The **kexec-tools** package
- Fulfilled requirements for **kdump** configurations and targets. For details, see [Supported kdump configurations and targets](#).

**Procedure**

1. Check whether **kdump** is installed on your system:

   ```bash
   # rpm -q kexec-tools
   ```

   **Output if the package is installed:**

   ```bash
   # kexec-tools-2.0.22-13.el9.x86_64
   ```

   **Output if the package is not installed:**

   ```bash
   package kexec-tools is not installed
   ```

2. Install **kdump** and other necessary packages by:

   ```bash
   # dnf install kexec-tools
   ```
CHAPTER 4. CONFIGURING KDUMP ON THE COMMAND LINE

Plan and build your kdump environment.

4.1. ESTIMATING THE KDUMP SIZE

When planning and building your kdump environment, it is important to know how much space the crash dump file requires.

The makedumpfile --mem-usage command estimates how much space the crash dump file requires. It generates a memory usage report. The report helps you determine the dump level and which pages are safe to be excluded.

Procedure

- Execute the following command to generate a memory usage report:

  ```bash
  # makedumpfile --mem-usage /proc/kcore
  ```

<table>
<thead>
<tr>
<th>TYPE</th>
<th>PAGES</th>
<th>EXCLUDABLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZERO</td>
<td>501635</td>
<td>yes</td>
<td>Pages filled with zero</td>
</tr>
<tr>
<td>CACHE</td>
<td>51657</td>
<td>yes</td>
<td>Cache pages</td>
</tr>
<tr>
<td>CACHE_PRIVATE</td>
<td>5442</td>
<td>yes</td>
<td>Cache pages + private</td>
</tr>
<tr>
<td>USER</td>
<td>16301</td>
<td>yes</td>
<td>User process pages</td>
</tr>
<tr>
<td>FREE</td>
<td>77738211</td>
<td>yes</td>
<td>Free pages</td>
</tr>
<tr>
<td>KERN_DATA</td>
<td>1333192</td>
<td>no</td>
<td>Dumpable kernel data</td>
</tr>
</tbody>
</table>

IMPORTANT

The makedumpfile --mem-usage command reports required memory in pages. This means that you must calculate the size of memory in use against the kernel page size.

4.2. CONFIGURING KDUMP MEMORY USAGE

The memory reservation for kdump occurs during the system boot. The memory size is set in the system's Grand Unified Bootloader (GRUB) configuration. The memory size depends on the value of the crashkernel= option specified in the configuration file and the size of the system physical memory.

You can define the crashkernel= option in many ways. You can specify the crashkernel= value or configure the auto option. The crashkernel=auto parameter reserves memory automatically, based on the total amount of physical memory in the system. When configured, the kernel automatically reserves an appropriate amount of required memory for the capture kernel. This helps to prevent Out-of-Memory (OOM) errors.

NOTE

The automatic memory allocation for kdump varies based on system hardware architecture and available memory size.

If the system has less than the minimum memory threshold for automatic allocation, you can configure the amount of reserved memory manually.
Prerequisites

- You have root permissions on the system.
- Fulfilled requirements for `kdump` configurations and targets. For details, see [Supported kdump configurations and targets](#).

Procedure

1. Prepare the `crashkernel=` option.
   - For example, to reserve 128 MB of memory, use the following:
     ```
     crashkernel=128M
     ```
   - Alternatively, you can set the amount of reserved memory to a variable depending on the total amount of installed memory. The syntax for memory reservation into a variable is `crashkernel=<range1>:<size1>,<range2>:<size2>`. For example:
     ```
     crashkernel=512M-2G:64M,2G-:128M
     ```
     The command reserves 64 MB of memory if the total amount of system memory is in the range of 512 MB and 2 GB. If the total amount of memory is more than 2 GB, the memory reserve is 128 MB.
   - Offset the reserved memory.
     Some systems require to reserve memory with a certain fixed offset because the `crashkernel` reservation happens early, and you may need to reserve more memory for special usage. When you define an offset, the reserved memory begins there. To offset the reserved memory, use the following syntax:
     ```
     crashkernel=128M@16M
     ```
     In this example, `kdump` reserves 128 MB of memory starting at 16 MB (physical address 0x01000000). If you set the offset parameter to 0 or omit entirely, `kdump` offsets the reserved memory automatically. You can also use this syntax when setting a variable memory reservation. In that case, the offset is always specified last. For example:
     ```
     crashkernel=512M-2G:64M,2G-:128M@16M
     ```

2. Apply the `crashkernel=` option to your boot loader configuration:
   ```
   # grubby --update-kernel=ALL --args="crashkernel=<value>"
   ```
   Replace `<value>` with the value of the `crashkernel=` option that you prepared in the previous step.

Additional resources

- Memory requirements for `kdump`
- Configuring kernel command-line parameters
- How to manually modify the boot parameter in grub before the system boots
4.3. CONFIGURING THE KDUMP TARGET

The crash dump is usually stored as a file in a local file system, written directly to a device. Alternatively, you can set up for the crash dump to be sent over a network using the **NFS** or **SSH** protocols. Only one of these options to preserve a crash dump file can be set at a time. The default behavior is to store it in the */var/crash/* directory of the local file system.

**Prerequisites**

- Root permissions.
- Fulfilled requirements for **kdump** configurations and targets. For details, see Supported kdump configurations and targets.

**Procedure**

To store the crash dump file in */var/crash/* directory of the local file system, edit the */etc/kdump.conf* file and specify the path:

```
path /var/crash
```

The option `path /var/crash` represents the path to the file system in which **kdump** saves the crash dump file.

**NOTE**

- When you specify a dump target in the */etc/kdump.conf* file, then the path is relative to the specified dump target.
- When you do not specify a dump target in the */etc/kdump.conf* file, then the path represents the absolute path from the root directory.

Depending on what is mounted in the current system, the dump target and the adjusted dump path are taken automatically.

**Example 4.1. The kdump target configuration**

```
# grep -v ^# /etc/kdump.conf | grep -v ^$
ext4 /dev/mapper/vg00-varcrashvol
path /var/crash
core_collector makedumpfile -c --message-level 1 -d 31
```

Here, the dump target is specified (`ext4 /dev/mapper/vg00-varcrashvol`), and thus mounted at */var/crash*. The `path` option is also set to */var/crash*, so the **kdump** saves the vmcore file in the */var/crash/var/crash* directory.

- To change the local directory in which the crash dump is to be saved, as root, edit the */etc/kdump.conf* configuration file:
1. Remove the hash sign (“#”) from the beginning of the `#path /var/crash` line.

2. Replace the value with the intended directory path. For example:
   ```
   path /usr/local/cores
   ```

**IMPORTANT**

In RHEL 9, the directory defined as the kdump target using the `path` directive must exist when the `kdump systemd` service is started - otherwise the service fails.

- To write the file to a different partition, edit the `/etc/kdump.conf` configuration file:

  1. Remove the hash sign (“#”) from the beginning of the `#ext4` line, depending on your choice.
     - device name (the `#ext4 /dev/vg/lv_kdump` line)
     - file system label (the `#ext4 LABEL=/boot` line)
     - UUID (the `#ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937` line)
  2. Change the file system type as well as the device name, label or UUID to the desired values. For example:
     ```
     ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937
     ```

**NOTE**

The correct syntax for specifying UUID values is both `UUID="correct-uuid"` and `UUID=correct-uuid`.

**IMPORTANT**

It is recommended to specify storage devices using a `LABEL=` or `UUID=`. Disk device names such as `/dev/sda3` are not guaranteed to be consistent across reboot.

- To write the crash dump directly to a device, edit the `/etc/kdump.conf` configuration file:

  1. Remove the hash sign (“#”) from the beginning of the `#raw /dev/vg/lv_kdump` line.
  2. Replace the value with the intended device name. For example:
     ```
     raw /dev/sdb1
     ```

- To store the crash dump to a remote machine using the NFS protocol:

  1. Remove the hash sign (“#”) from the beginning of the `#nfs my.server.com:/export/tmp` line.
  2. Replace the value with a valid hostname and directory path. For example:
     ```
     nfs penguin.example.com:/export/cores
     ```
To store the crash dump to a remote machine using the **SSH** protocol:

1. Remove the hash sign ("#") from the beginning of the `#ssh user@my.server.com` line.
2. Replace the value with a valid username and hostname.
3. Include your **SSH** key in the configuration.
   - Remove the hash sign from the beginning of the `#sshkey /root/.ssh/kdump_id_rsa` line.
   - Change the value to the location of a key valid on the server you are trying to dump to. For example:
     
     ```
     ssh john@penguin.example.com
     sshkey /root/.ssh/mykey
     ```

### 4.4. CONFIGURING THE KDUMP CORE COLLECTOR

The **kdump** service uses a **core_collector** program to capture the crash dump image. In RHEL, the **makedumpfile** utility is the default core collector. It helps shrink the dump file by:

- Compressing the size of a crash dump file and copying only necessary pages using various dump levels
- Excluding unnecessary crash dump pages
- Filtering the page types to be included in the crash dump.

**Syntax**

```
core_collector makedumpfile -l --message-level 1 -d 31
```

**Options**

- `-c`, `-l` or `-p`: specify compress dump file format by each page using either, **zlib** for `-c` option, **lzo** for `-l` option or **snappy** for `-p` option.
- `-d (dump_level)`: excludes pages so that they are not copied to the dump file.
- `--message-level`: specify the message types. You can restrict outputs printed by specifying `message_level` with this option. For example, specifying 7 as `message_level` prints common messages and error messages. The maximum value of `message_level` is 31

**Prerequisites**

- You have root permissions on the system.
- Fulfilled requirements for **kdump** configurations and targets. For details, see Supported kdump configurations and targets.

**Procedure**

1. As **root**, edit the `/etc/kdump.conf` configuration file and remove the hash sign ("#") from the beginning of the `#core_collector makedumpfile -l --message-level 1 -d 31`. 

2. To enable crash dump file compression, execute:

```
core_collector makedumpfile -l --message-level 1 -d 31
```

The `-l` option specifies the dump compressed file format. The `-d` option specifies dump level as 31. The `--message-level` option specifies message level as 1.

Also, consider following examples with the `-c` and `-p` options:

- To compress a crash dump file using `-c`:

```
core_collector makedumpfile -c -d 31 --message-level 1
```

- To compress a crash dump file using `-p`:

```
core_collector makedumpfile -p -d 31 --message-level 1
```

**Additional resources**

- `makedumpfile(8)` man page
- Configuration file for kdump

### 4.5. CONFIGURING THE KDUMP DEFAULT FAILURE RESPONSES

By default, when `kdump` fails to create a crash dump file at the configured target location, the system reboots and the dump is lost in the process. To change this behavior, follow the procedure below.

**Prerequisites**

- Root permissions.
- Fulfilled requirements for `kdump` configurations and targets. For details, see  Supported `kdump` configurations and targets.

**Procedure**

1. As `root`, remove the hash sign (“#”) from the beginning of the `#failure_action` line in the `/etc/kdump.conf` configuration file.

```
failure_action poweroff
```

2. Replace the value with a desired action.

**Additional resources**

- Configuring the `kdump` target

### 4.6. TESTING THE KDUMP CONFIGURATION

Testing the `kdump` configuration validates the configuration and also records the time taken for a crash dump to complete with the specified workload.
WARNING
The commands to test `kdump` configuration will cause the kernel to crash with loss of data. Follow the instructions with care and do not use an active production system to test the `kdump` configuration.

Procedure

1. Reboot the system with `kdump` enabled.

2. Check if `kdump` is active.

   ```
   # systemctl is-active kdump
   active
   ```

3. Force a kernel crash.

   ```
   echo c > /proc/sysrq-trigger
   ```

   WARNING
   The command causes the kernel to crash and reboots the kernel if required.

On a kernel reboot, the `address-YYYY-MM-DD-HH:MM:SS/vmcore` file is created at the location you have specified in the `/etc/kdump.conf` file (by default to `/var/crash/`).

Additional resources

- Configuring the `kdump` target
CHAPTER 5. ENABLING KDUMP

By using the procedure, you can enable or disable the *kdump* service for all installed kernels or for a specific kernel.

### 5.1. ENABLING KDUMP FOR ALL INSTALLED KERNELS

You can enable and start the *kdump* service for all kernels installed on the machine.

**Prerequisites**

- Administrator privileges

**Procedure**

1. Add the *crashkernel=auto* command-line parameter to all installed kernels:
   
   ```bash
   # grubby --update-kernel=ALL --args="crashkernel=auto"
   ```

2. Enable the *kdump* service.
   
   ```bash
   # systemctl enable --now kdump.service
   ```

**Verification**

- Check that the *kdump* service is running:
  
  ```bash
  # systemctl status kdump.service
  
  ○ kdump.service - Crash recovery kernel arming
    Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset: disabled)
    Active: active (live)
  ```

### 5.2. ENABLING KDUMP FOR A SPECIFIC INSTALLED KERNEL

You can enable the *kdump* service for a specific kernel on the machine.

**Prerequisites**

- Administrator privileges

**Procedure**

1. List the kernels installed on the machine.
   
   ```bash
   # ls -a /boot/vmlinuz-*
   /boot/vmlinuz-0-rescue-2930657cd0dc43c2b75db480e5e5b4a9 /boot/vmlinuz-4.18.0-330.el8.x86_64 /boot/vmlinuz-4.18.0-330.rt7.111.el8.x86_64
   ```

2. Add a specific *kdump* kernel to the system's Grand Unified Bootloader (GRUB) configuration file.
For example:

```
# grubby --update-kernel=vmlinuz-4.18.0-330.el8.x86_64 --args="crashkernel=auto"
```

3. Enable the **kdump** service.

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

**Verification**

- Check that the **kdump** service is running:

```
# systemctl status kdump.service
```

○ kdump.service - Crash recovery kernel arming
  Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset: disabled)
  Active: active (live)

### 5.3. DISABLING THE KDUMP SERVICE

To disable the **kdump** service at boot time, follow the procedure below.

**Prerequisites**

- Fulfilled requirements for **kdump** configurations and targets. For details, see [Supported kdump configurations and targets](#).

- All configurations for installing **kdump** are set up according to your needs. For details, see [Installing kdump](#).

**Procedure**

1. To stop the **kdump** service in the current session:

```
# systemctl stop kdump.service
```

2. To disable the **kdump** service:

```
# systemctl disable kdump.service
```

**WARNING**

It is recommended to set `kptr_restrict=1`. In that case, the **kdumpctl** service loads the crash kernel regardless of Kernel Address Space Layout (KASLR) being enabled or not.
Troubleshooting step

When `kptr_restrict` is not set to (1), and if KASLR is enabled, the contents of `/proc/kcore` file are generated as all zeros. Consequently, the `kdumpctl` service fails to access the `/proc/kcore` and load the crash kernel.

To work around this problem, the `/usr/share/doc/kexec-tools/kexec-kdump-howto.txt` file displays a warning message, which recommends the `kptr_restrict=1` setting.

To ensure that `kdumpctl` service loads the crash kernel, verify that `kernel.kptr_restrict = 1` is listed in the `sysctl.conf` file.

Additional resources

- Managing systemd
CHAPTER 6. REPORTING RHEL FOR REAL TIME BUGS

The preferred method for reporting a RHEL for Real Time bug is to submit a bug report on the Red Hat Bugzilla. Before filing a bug, it is useful to identify the source where the problem occurred, such as the standard kernel or the RHEL for Real Time kernel.

6.1. DIAGNOSING RHEL FOR REAL TIME BUGS

Identifying which kernel, the RHEL for Real Time or the standard kernel, is the source of the problem can increase the chances of having your bug fixed faster. By following the procedure steps, you can diagnose the source of the problem before submitting a bug report.

Prerequisite:

- The latest version of RHEL for Real Time kernel is installed.

Procedure:

1. Verify that you have the latest version of the RHEL for Real Time kernel.
2. Boot into RHEL for Real Time kernel using the GRUB menu.
3. If the problem occurs, report a bug against RHEL for Real Time.
4. Try to reproduce the problem with the standard kernel. This troubleshooting step assists in identifying the problem location.

NOTE

If the problem does not occur with the standard kernel, then the bug is probably the result of changes introduced in the RHEL for Real Time specific enhancements, which Red Hat has applied on top of the baseline (4.18.0) kernel.

6.2. SUBMITTING A BUG REPORT WITH BUGZILLA

After identifying the bug specific to RHEL for Real Time, use the following procedure steps to submit a bug report with Bugzilla.

Prerequisite:

- You have a Red Hat Bugzilla account.

Procedure

1. Log into your Bugzilla account.
2. Click Enter A New Bug Report.
4. Select the Red Hat Enterprise Linux product.
5. Enter **Component**.
   For example, use **kernel-rt** if it is a kernel issue or the name of the affected user space component, such as **rteval**.

   When entering the problem description you can also state if you were able to reproduce the problem on the standard RHEL 8 kernel.

Additional resources

- [Red Hat Bugzilla – Create a new Red Hat Bugzilla account](#)