Red Hat Enterprise Linux for Real Time 9

RHEL 9 for Real Time のインストール

Red Hat Enterprise Linux への RHEL for Real Time カーネルのインストール

Last Updated: 2024-08-30

Red Hat Enterprise Linux for Real Time 9 RHEL 9 for Real Time のインストール

Red Hat Enterprise Linux への RHEL for Real Time カーネルのインストール

法律上の通知

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

Red Hat Enterprise Linux に RHEL for Real Time カーネルをインストールして、低レイテンシーを取得し、応答時間を予測可能にします。リアルタイムカーネルのインストール、インストール後のタスクの実行、リアルタイム、ストック、起動するデバッグカーネルなどのカーネルバリアントの設定方法を説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、用語の置き換えは、今後の複数のリリースにわたって段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 RHEL FOR REAL TIME のインストール

多くの業界や組織には、非常に高いパフォーマンスのコンピューティング機能が必要で、特に金融業界や通信業界では、小さく予測可能なレイテンシーが必要になる場合があります。レイテンシー (応答時間) はイベントとシステム応答間の時間として定義され、通常マイクロ秒 (µs) で測定されます。

Linux 環境で実行されているほとんどのアプリケーションでは、基本的なパフォーマンスチューニングにより、レイテンシーを十分に改善できます。Red Hat は、レイテンシーを低く保つだけでなく、予測可能な機能も必要とする業界向けに、この内容を提供する 'ドロップイン' のカーネル置き換えを開発しました。RHEL for Real Time は RHEL 9 の一部として提供され、RHEL 9 とのシームレスな統合を提供します。RHEL for Real Time により、組織内のレイテンシーを測定、設定、記録することができます。

警告

RHEL for Real Time をインストールする前に、ベースプラットフォームが適切にチューニングされ、システム BIOS パラメーターが調整されていることを確認します。これらのタスクの実行に失敗すると、RHEL Real Time デプロイメントから一貫したパフォーマンスを得ることができなくなる可能性があります。

1.1. RHEL FOR REAL TIME でのレイテンシーの最適化

RHEL for Real Time は、非常に高い決定要件があるアプリケーション用に、適切にチューニングされたシステムでの使用を目的に設計されています。カーネルシステムのチューニングにより、決定論の改善が大きくなります。

たとえば、多くのワークロードで、システムを徹底してチューニングすることで、約 90% ほど一貫性が向上します。このため、RHEL for Real Time を使用する前に、まず標準的な RHEL のシステムチューニングを実行して、目的が満たされているかどうかを確認することが推奨されます。

システムのチューニングが重要なのは、リアルタイムカーネルを使用する場合でも標準カーネルの場合でも同じです。RHEL 9 リリースの一部として提供される標準カーネルを実行する未チューニングのシステムに、リアルタイムカーネルをインストールしても、目立った利点はありません。標準カーネルをチューニングすることで、達成可能なレイテンシー改善の 90% が得られます。リアルタイムカーネルは、最も要求の厳しいワークロードで必要とされる、レイテンシー改善の最後の 10% を提供します。

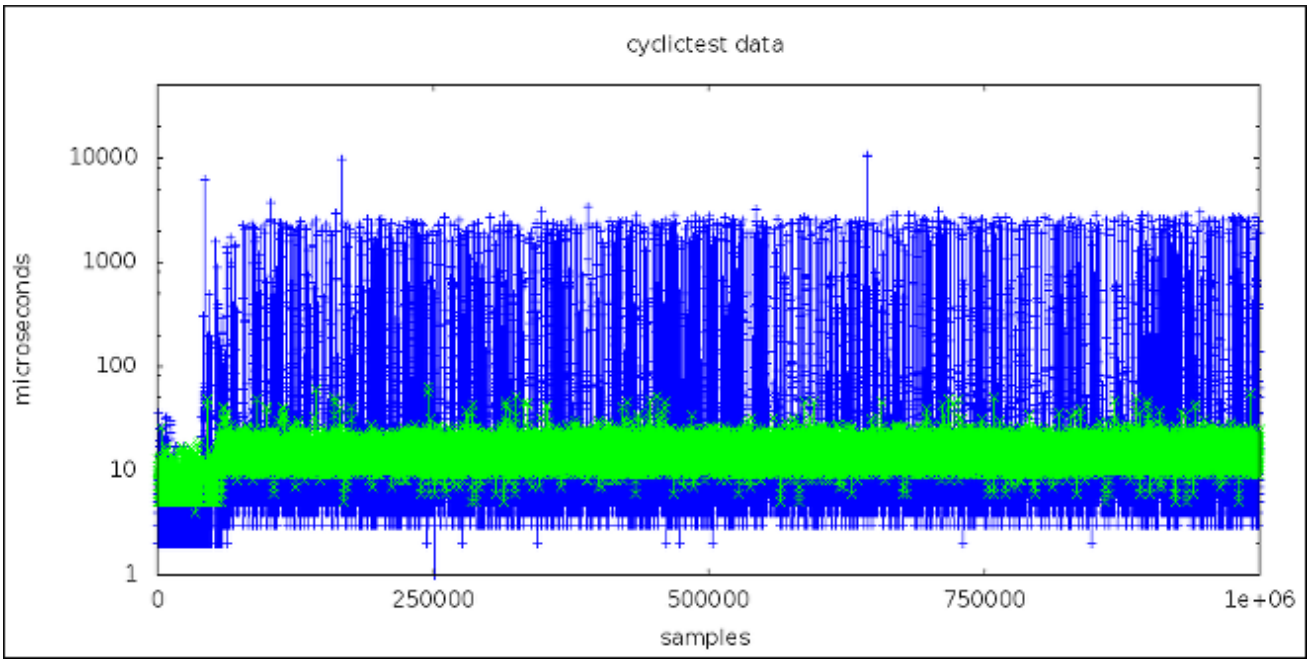
警告

リアルタイムカーネルシステムをチューニングする前に、ベースプラットフォームが適切にチューニングされ、システムの BIOS パラメーターの調整が行われていることを確認してください。これらのタスクの実行に失敗すると、RHEL Real Time デプロイメントから一貫したパフォーマンスを得ることができなくなる可能性があります。

リアルタイムカーネルの目的は、一貫性のある、低レイテンシーの、決定論に基づく予測可能な応答時間を提供することです。リアルタイムカーネルに関連する、追加のカーネルオーバーヘッドがありま

す。これは主に、個別にスケジュール設定されたスレッドでハードウェア割り込みを処理するためです。ワークロードのオーバーヘッドが増大すると、全体的なスループットが低下します。スループットの低下は 0% から 30% までの範囲ですが、正確な量はワークロードに大きく依存します。

カーネル遅延の要件がミリ秒 (ms) オーダーの般的なワークロードの場合は、標準の RHEL 9 カーネルで十分です。ただし、ワークロードに、割り込み処理やマイクロ秒 (µs) 範囲のプロセススケジューリングなどのコアカーネル機能に対する厳格な低遅延の決定論要件がある場合は、リアルタイムカーネルが最適です。



このグラフは、RHEL 9 と RHEL for Real Time カーネルを使用するマシン 100 万台のサンプルを比較したものです。

- このグラフの青い点は、チューニングされた RHEL 9 カーネルを実行しているマシンのシステム応答時間 (マイクロ秒単位) を表しています。
- グラフの緑色の点は、チューニングされたリアルタイムカーネルを実行しているマシンのシステムの応答時間を表しています。

このグラフからは、標準カーネルの応答時間が大きく変動しているのとは対照的に、リアルタイムカーネルの応答時間は一定であることが良く分かります。

1.2. DNF を使用した RHEL FOR REAL TIME のインストール

dnf を使用してリアルタイムカーネルをインストールするだけでなく、RHEL for Real Time を含む ISO イメージを Download Red Hat Enterprise Linux ポータルからダウンロードすることもできます。この ISO イメージを使用して、RHEL for Real Time に含まれるすべての RPM パッケージを取得できます。ただし、これは起動可能な ISO イメージではないため、これを使用して起動可能な USB または CD メディアを作成することはできません。

前提条件

- AMD64 または Intel64 システムに最新バージョンの RHEL 9 がインストールされている。リアルタイムカーネルは、Red Hat Enterprise Linux の実行が認定されている AMD64 および Intel 64 (x86_64 とも呼ばれます) サーバープラットフォームで動作します。
- マシンが登録され、RHEL が RHEL for Real Time サブスクリプションに割り当てられている。

- ベースプラットフォームが適切にチューニングされ、システム BIOS パラメーターが調整されている。

注記

リアルタイムカーネルをインストールする前に前提条件のタスクを実行しないと、RHEL for Real Time カーネルのデプロイメントで一貫したパフォーマンスが得られなくなる可能性があります。

手順

1. RHEL for Real Time リポジトリーを有効にします。

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

2. RHEL for Real Time パッケージグループをインストールします。

```
# dnf groupinstall RT
```

このグループにより、複数のパッケージがインストールされます。

- **kernel-rt** には、RHEL for Real Time カーネルパッケージが含まれています。
- **kernel-rt-core** には、コア RHEL for Real Time カーネルパッケージが含まれています。
- **kernel-rt-devel** には、RHEL for Real Time カーネル開発パッケージが含まれています。
- **kernel-rt-modules** には、RHEL for Real Time カーネルモジュールパッケージが含まれています。
- **kernel-rt-modules-core** には、コアカーネルパッケージのカーネルモジュールが含まれています。
- **kernel-rt-modules-extra** には、RHEL for Real Time カーネル追加モジュールパッケージが含まれています。
- **realtime-setup** は、RHEL for Real Time に必要な基本環境をセットアップします。
- **rteval** は、RHEL for Real Time のシステム適合性を評価します。
- **rteval-loads** には、**rteval** ロード用のソースコードが含まれています。
- **tuned-profiles-realtime** には、リアルタイムを対象とした追加の **TuneD** プロファイルを含むパッケージが含まれています。

3. (オプション) さらに、**tuna** パッケージがあります。これは、リアルタイムカーネルワークロードのチューニングに役立つツールを提供し、コマンドラインまたは GUI からの CPU 分離とスレッドアフィニティー操作を大幅に自動化します。このパッケージは、ベースの RHEL 9 リポジトリーで入手できます。

```
# dnf install tuna
```

注記

RHEL for Real Time カーネルがインストールされると、自動的にデフォルトのカーネルに設定され、次回の起動時に使用されます。**kernel**、**kernel-debug**、または **kernel-rt-debug** などの他の既存のカーネルバリアントをデフォルトのブートカーネルとして設定することもできます。詳細は、kernel-rt をデフォルトのブートカーネルとして設定するを参照してください。

検証手順

- インストールの場所を確認し、コンポーネントが正常にインストールされていることを確認します。

```
# rpm -ql realtime-setup
/etc/security/limits.d/realtime.conf
/etc/sysconfig/realtime-setup
/etc/udev/rules.d/99-rhel-rt.rules
/usr/bin/realtime-setup
/usr/bin/rt-setup-kdump
/usr/bin/slub_cpu_partial_off
/usr/lib/.build-id
/usr/lib/.build-id/a4
/usr/lib/.build-id/a4/da77908aa4c6f048939f3267f1c552c456d117
/usr/lib/systemd/system/rt-entsk.service
/usr/lib/systemd/system/rt-setup.service
/usr/sbin/kernel-is-rt
/usr/sbin/rt-entsk
```

関連情報

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

1.3. RHEL FOR REAL TIME リポジトリーで利用可能な RPM パッケージ

Red Hat Package Manager (RPM) for RHEL for Real Time リポジトリーには、以下のパッケージが含まれます。

- RHEL for Real Time カーネルパッケージである **kernel-rt** パッケージ。
- real-time カーネルのテストプログラムを含む RHEL for Real Time カーネルテストパッケージ。
- RHEL for Real Time デバッグパッケージ (デバッグおよびコードトレース用)。

表1.1 基本的な RHEL for Real Time カーネルパッケージ

RPM パッケージ名	説明	RT 固有	必須
kernel-rt	低レイテンシーおよびプリエンプション機能	はい	はい

表1.2 RHEL for Real Time カーネルテストパッケージ

RPM パッケージ名	説明	RT 固有	必須
kernel-rt-devel	カーネル開発用のヘッダーおよびライブラリー	はい	いいえ
kernel-rt-debug	デバッグ機能がコンパイルされた RHEL for Real Time カーネル (低速)	はい	いいえ
kernel-rt-debug-devel	デバッグカーネル開発に使用するヘッダーおよびライブラリー	はい	いいえ
realtime-tests	システムレイテンシーを測定し、優先度継承ミューテックスが適切に機能することを証明するためのユーティリティー	いいえ	いいえ

デバッグパッケージは、カーネルクラッシュダンプを分析するために **perf**、**trace-cmd**、および **crash** ユーティリティーで使用するために提供されています。デバッグパッケージにはシンボルテーブルが含まれ、これらは非常に大きいです。このため、デバッグパッケージは、他の RHEL for Real Time パッケージとは別に配信されます。デバッグパッケージは、**RHEL for Real Time - Debug RPMs** リポジトリーからダウンロードできます。

表1.3 基本的な RHEL for Real Time デバッグパッケージ

RPM パッケージ名	説明	RT 固有	必須
kernel-rt-debuginfo	**perf** や **trace-cmd** など、プロファイリングおよびデバッグ用のシンボル	はい	いいえ
kernel-rt-debug-debuginfo	プロファイリングおよび追跡用のシンボル	はい	いいえ

1.4. インストール後の手順

リアルタイムカーネルをインストールした後、次のことを確認します。

- 最適な低レイテンシーの決定論を実現するには、RHEL for Real Time 固有のシステムチューニングを実行します。
- リアルタイムカーネルと標準カーネルのモジュール互換性について理解します。
- **kdump** を有効にするには、**kexec/kdump** を有効にしてクラッシュダンプ情報を提供するように RHEL for Real Time を設定する必要があります。
- Real Time カーネルがデフォルトのカーネルであることを確認するには、以下のコマンドを実行します。

リアルタイムカーネルと標準カーネルのモジュール互換性

リアルタイムカーネルは、標準の Red Hat Enterprise Linux 9 カーネルとは大きく異なります。結果として、サードパーティーのカーネルモジュールは RHEL for Real Time と互換性がありません。

カーネルモジュールは、特別にビルドされたカーネルに固有のものです。リアルタイムカーネルは、標準のカーネルとは大きく異なるため、モジュールも大きく異なります。したがって、Red Hat Enterprise Linux 9 からサードパーティーのモジュールを取得して、リアルタイムカーネルでそのまま使用することはできません。サードパーティーのモジュールを使用する必要がある場合は、RHEL for Real Time の開発およびテストパッケージで利用可能な RHEL for Real Time ヘッダーファイルを使用して再コンパイルする必要があります。標準の Red Hat Enterprise Linux 9 に同梱されているものの、現時点ではリアルタイムカーネル用のカスタムビルドがないサードパーティードライバーは次のとおりです。

- EMC Powerpath
- NVidia graphics
- Qlogic の高度なストレージアダプター設定ユーティリティー

ユーザー空間の **syscall** インターフェイスは、RHEL for Real Time と互換性があります。

第2章 実行する RHEL カーネルの指定

インストール済みの任意のカーネル、標準、または Real Time を起動できます。起動中に GRUB メニューで必要なカーネルを手動で選択できます。どのカーネルをデフォルトで起動するかを設定することもできます。

real-time カーネルがインストールされると、自動的にデフォルトのカーネルに設定され、次回の起動時に使用されます。

2.1. デフォルトのカーネルの表示

デフォルトで起動するように設定されたカーネルを表示できます。

手順

- デフォルトのカーネルを表示するには、以下のコマンドを実行します。

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

コマンドの出力の **rt** は、デフォルトのカーネルがリアルタイムカーネルであることを示しています。

2.2. 実行中のカーネルの表示

現在実行中のカーネルを表示できます。

手順

- 現在システムが実行中のカーネルを表示します。

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

注記

システムが、8.3 から 8.4 などのマイナー更新を受け取ると、デフォルトのカーネルが Real Time カーネルから標準カーネルに自動的に戻る場合があります。

2.3. KERNEL-RT をデフォルトのブートカーネルとして設定する

新しくインストールされたシステムでは、標準の RHEL **kernel** がデフォルトのブートカーネルとして設定され、次回の起動時およびその後のシステム更新時にデフォルトのカーネルとして使用されます。この設定を変更して、ブートに使用するデフォルトのカーネルとして **kernel-rt** を設定し、すべてのシステム更新に対してこの設定を永続化することもできます。**kernel-rt** の設定は 1 回限りの手順であり、必要に応じて変更したり、別のカーネルに戻すことができます。**kernel**、**kernel-debug**、または **kernel-rt-debug** などの他の既存のカーネルバリアントをデフォルトのブートカーネルとして設定することもできます。

手順

1. **kernel-rt** をデフォルトのブートカーネルとして設定するには、次のコマンドを入力します。

```
# grubby --set-default=<RT_VMLINUZ>
```

RT_VMLINUZ は、**kernel-rt** カーネルに関連付けられた **vmlinux** ファイルの名前です。以下に例を示します。

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

2. システム更新時のデフォルトのブートカーネルとして **kernel-rt** を設定するには、次のコマンドを入力します。

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

UPDATEDEFAULT 変数を **yes** として指定すると、システムの更新で変更されるデフォルトのカーネルが設定されます。

出力例のデフォルトカーネルのパスは、インストールされている **kernel-rt-core** パッケージに固有のものです。**rpm -q kernel-rt-core** コマンドを使用すると、カーネルへのパスをパッケージから確認できます。

a. オプション: カーネルへのパスをパッケージから確認する必要がある場合は、まずインストールされているパッケージをリストします。

```
# 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. 最新のインストール済みパッケージをデフォルトとして使用するには、次のコマンドを入力して、ブートイメージへのパスをそのパッケージから見つけます。

```
# 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. **kernel-rt** をデフォルトのブートカーネルとして設定するには、次のコマンドを入力します。

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

検証

- **kernel-rt** がデフォルトのカーネルであることを確認するには、次のコマンドを入力します。

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

第3章 KDUMP のインストール

Red Hat Enterprise Linux の新規インストールでは、デフォルトで **kdump** サービスがインストールされ有効になっています。**kdump** について、およびデフォルトで有効になっていない場合に **kdump** をインストールする方法を説明します。

3.1. KDUMP とは

kdump は、クラッシュダンプのメカニズムを提供するサービスです。このサービスでは、分析用にシステムメモリーの内容を保存できます。**kdump** は **kexec** システムコールを使用し、再起動することなく 2 番目のカーネル (**キャプチャーカーネル**) で起動し、クラッシュしたカーネルメモリー (**クラッシュダンプ** または**vmcore**) の内容をキャプチャーして、ファイルへ保存します。この第 2 のカーネルは、システムメモリーの予約部分に収納されています。

重要

カーネルクラッシュダンプは、システム障害 (重大なバグ) 時に利用できる唯一の情報になります。したがって、ミッションクリティカルな環境では、**kdump** を稼働させることが重要です。Red Hat は、システム管理者が通常のカーネル更新サイクルで **kexec-tools** を定期的に更新してテストすることを推奨します。これは、新しいカーネル機能が実装されている場合に特に重要です。

kdump は、マシンにインストールされているすべてのカーネルに対して、または指定したカーネルに対してのみ有効にできます。これは、マシンで複数のカーネルが使用されており、その一部が安定しており、クラッシュの心配がない場合に役立ちます。

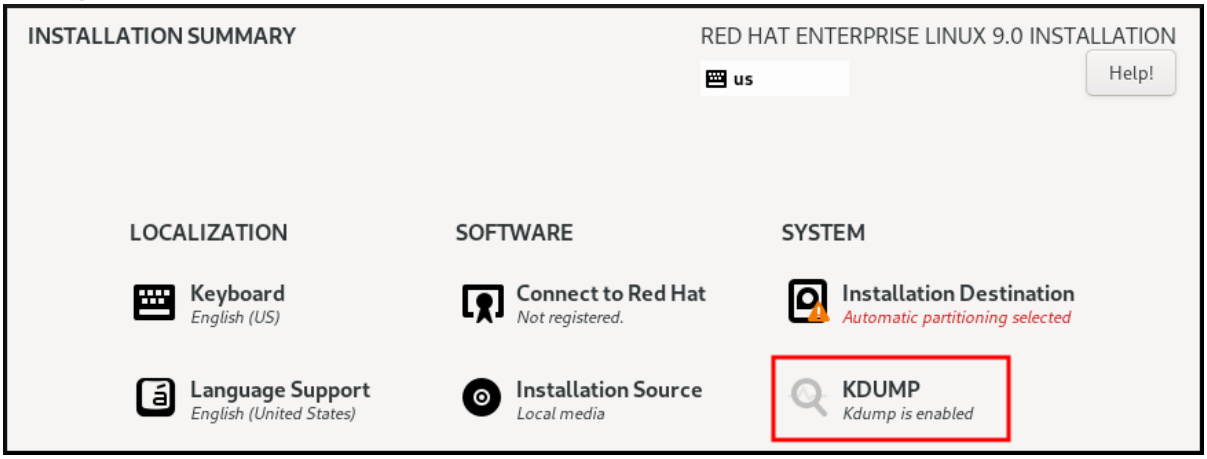
kdump をインストールすると、デフォルトの **/etc/kdump.conf** が作成されます。このファイルには、デフォルトの最小限の **kdump** 設定が含まれます。このファイルを編集して、**kdump** 設定をカスタマイズできますが、必須ではありません。

3.2. ANACONDA を使用した KDUMP のインストール

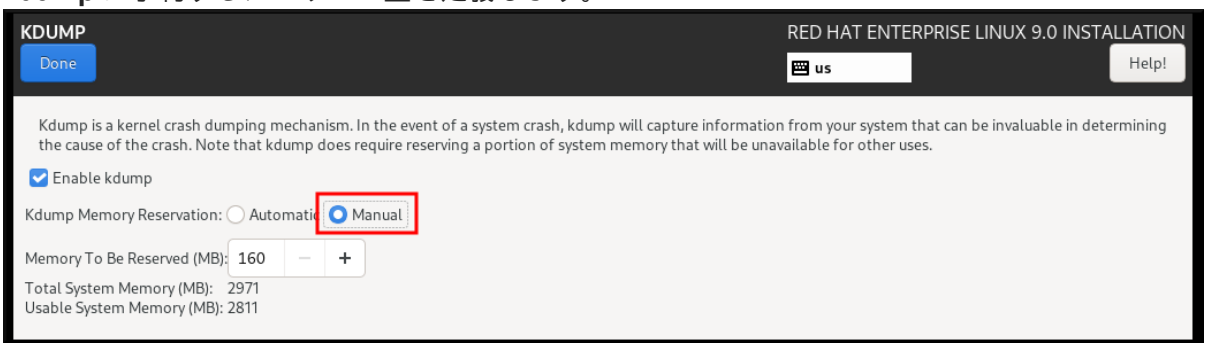
Anaconda インストーラーでは、対話式インストール時に **kdump** 設定用のグラフィカルインターフェイス画面が表示されます。インストーラー画面のタイトルは KDUMP で、メインの **インストールの概要** 画面から利用できます。**kdump** を有効にして、必要な量のメモリーを予約できます。

手順

1. **Kdump** 項目に移動します。
2. **kdump** が有効になっていない場合は有効にします。



3. **kdump** に予約するメモリーの量を定義します。



3.3. コマンドラインで KDUMP のインストール

カスタムの Kickstart インストールなどの一部のインストールオプションでは、デフォルトで **kdump** がインストールまたは有効化**されない**場合があります。この場合は、以下の手順を行ってください。

前提条件

- アクティブな RHEL サブスクリプション
- **kexec-tools** パッケージ
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

1. **kdump** がシステムにインストールされているかどうかを確認します。

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

このパッケージがインストールされている場合は以下を出力します。

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

このパッケージがインストールされていない場合は以下を出力します

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

2. **kdump** および必要なパッケージをインストールします。

```
# dnf install kexec-tools
```

第4章 コマンドラインで KDUMP の設定

kdump 環境を計画して構築します。

4.1. KDUMP サイズの見積もり

kdump 環境の計画および構築を行う際に、クラッシュダンプファイルに必要な領域を把握しておくことが重要です。

makedumpfile --mem-usage コマンドは、クラッシュダンプファイルに必要な領域を推定し、メモリー使用量に関するレポートを生成します。このレポートは、ダンプレベルと、除外して問題ないページを判断するのに役立ちます。

手順

- 次のコマンドを実行して、メモリー使用量に関するレポートを生成します。

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


TYPE        PAGES    EXCLUDABLE    DESCRIPTION
----------------------------------------------------------------
ZERO          501635      yes        Pages filled with zero
CACHE          51657      yes        Cache pages
CACHE_PRIVATE 5442        yes        Cache pages + private
USER          16301       yes        User process pages
FREE          77738211    yes        Free pages
KERN_DATA     1333192     no         Dumpable kernel data
```

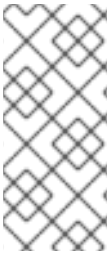
重要

makedumpfile --mem-usage は、必要なメモリーをページ単位で報告します。つまり、カーネルページサイズを元に、使用するメモリーのサイズを計算する必要があります。

4.2. メモリー使用量の設定

kdump のメモリー予約は、システムの起動中に行われます。メモリーサイズは、システムの GRUB (Grand Unified Bootloader) 設定で設定されます。メモリーサイズは、設定ファイルで指定された **crashkernel=** オプションの値と、システムの物理メモリーのサイズにより異なります。

crashkernel= オプションはさまざまな方法で定義できます。**crashkernel=** 値を指定するか、**auto** オプションを設定できます。**crashkernel=auto** パラメーターは、システムの物理メモリーの合計量に基づいて、メモリーを自動的に予約します。これを設定すると、カーネルは、キャプチャーカーネルに必要な適切な量のメモリーを自動的に予約します。これにより、OOM (Out-of-Memory) エラーの回避に役立ちます。



注記

kdump の自動メモリー割り当ては、システムのハードウェアアーキテクチャーと利用可能なメモリーサイズによって異なります。

システムに、自動割り当ての最小メモリーしきい値より少ないメモリーしかない場合は、手動で予約メモリーの量を設定できます。

前提条件

- システムの root 権限がある。
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

1. **crashkernel=** オプションを準備してください。
   - たとえば、128 MB のメモリーを予約するには、以下を使用します。

     ```
     crashkernel=128M
     ```

   - または、インストールされているメモリーの合計量に応じて、予約メモリーサイズを変数に設定できます。変数へのメモリー予約の構文は **crashkernel=<range1>:<size1>,<range2>:<size2>** です。以下に例を示します。

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

     システムメモリーの合計量が 512 MB - 2 GB の範囲にある場合、64 MB のメモリーを予約します。メモリーの合計量が 2 GB を超える場合、メモリー予約は 128 MB になります。

   - 予約メモリーのオフセット。
     一部のシステムでは、**crashkernel** の予約が早い段階で行われるため、特定の固定オフセットでメモリーを予約する必要があります。また、特別な用途のために、さらに多くのメモリーの予約が必要になることもあります。オフセットを定義すると、予約メモリーはそこから開始されます。予約メモリーをオフセットするには、以下の構文を使用します。

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

     この例では、**kdump** は 16 MB (物理アドレス **0x01000000**) から始まる 128 MB のメモリーを予約します。offset パラメーターを 0 に設定するか、完全に省略すると、**kdump** は予約メモリーを自動的にオフセットします。変数のメモリー予約を設定する場合は、この構文を使用することもできます。その場合、オフセットは常に最後に指定されます。以下に例を示します。

     ```
     crashkernel=512M-2G:64M,2G-:128M@16M
     ```

2. **crashkernel=** オプションをブートローダー設定に適用します。

   ```
   # grubby --update-kernel=ALL --args="crashkernel=<value>"
   ```

   <value> は、前のステップで準備した **crashkernel=** オプションの値に置き換えます。

関連情報

- kdump メモリー要件
- カーネルコマンドラインパラメーターの設定
- システムを起動する前に、grub で boot パラメーターを手動で変更する

- How to install and boot custom kernels in Red Hat Enterprise Linux 8
- **grubby(8)** の man ページ

4.3. KDUMP ターゲットの設定

クラッシュダンプは通常、ローカルファイルシステムにファイルとして保存され、デバイスに直接書き込まれます。または、**NFS** プロトコルまたは **SSH** プロトコルを使用して、ネットワーク経由でクラッシュダンプを送信するように設定できます。クラッシュダンプファイルを保存するオプションは、一度に 1 つだけ設定できます。デフォルトの動作では、ローカルファイルシステムの **/var/crash/** ディレクトリーに保存されます。

前提条件

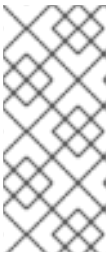
- **root** 権限
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

- ローカルファイルシステムの **/var/crash/** ディレクトリーにクラッシュダンプファイルを保存するには、**/etc/kdump.conf** ファイルを変更して、パスを指定します。

```
path /var/crash
```

path /var/crash オプションは、**kdump** がクラッシュダンプファイルを保存するファイルシステムへのパスを表します。



注記

- **/etc/kdump.conf** ファイルでダンプターゲットを指定すると、path は指定されたダンプ出力先に対する相対パスになります。
- **/etc/kdump.conf** ファイルでダンプターゲットを指定しない場合、パスはルートディレクトリーからの **絶対** パスを表します。

現在のシステムにマウントされている内容に応じて、ダンプターゲットと調整されたダンプパスが自動的に適用されます。

例4.1 kdump ターゲット設定

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

ここでは、ダンプターゲットが指定されているため (**ext4/dev/mapper/vg00-varcrashvol**)、**/var/crash** にマウントされます。**path** オプションも **/var/crash** に設定されているため、**kdump** は **vmcore** ファイルを **/var/crash/var/crash** ディレクトリーに保存します。

- クラッシュダンプを保存するローカルディレクトリーを変更するには、**root** として **/etc/kdump.conf** 設定ファイルを編集します。

  1. **#path /var/crash** の行頭にあるハッシュ記号 ("#") を削除します。
  2. 値を対象のディレクトリーパスに置き換えます。以下に例を示します。

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

     重要

     RHEL 9 では、**path** ディレクティブを使用して kdump ターゲットとして定義されたディレクトリーは、**kdump systemd** サービスの開始時に存在する必要があります。存在しない場合、サービスは失敗します。

- ファイルを別のパーティションに書き込むには、**/etc/kdump.conf** 設定ファイルを編集します。

  1. 必要に応じて **#ext4** の行頭にあるハッシュ記号 ("#") を削除します。
     - デバイス名 (**#ext4 /dev/vg/lv_kdump** 行)
     - ファイルシステムラベル (**#0ext4 LABEL=/boot** 行)
     - UUID (**#ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937** の行)
  2. ファイルシステムタイプと、デバイス名、ラベル、UUID を希望の値に変更します。以下に例を示します。

     ```
     ext4 UUID=03138356-5e61-4ab3-b58e-27507ac41937
     ```

     注記

     UUID 値を指定するための正しい構文は、**UUID="correct-uuid"** と **UUID=correct-uuid** の両方です。

     重要

     LABEL= または **UUID=** を使用してストレージデバイスを指定することが推奨されます。**/dev/sda3** などのディスクデバイス名は、再起動した場合に一貫性が保証されません。

- クラッシュダンプを直接書き込むには、**/etc/kdump.conf** 設定ファイルを修正します。

  1. **#raw /dev/vg/lv_kdump** の行頭にあるハッシュ記号 ("#") を削除します。
  2. 値を対象のデバイス名に置き換えます。以下に例を示します。

     ```
     raw /dev/sdb1
     ```

- **NFS** プロトコルを使用してクラッシュダンプをリモートマシンに保存するには、次の手順を実行します。

  1. **#nfs my.server.com:/export/tmp** の行頭にあるハッシュ記号 ("#") を削除します。

2. 値を、正しいホスト名およびディレクトリーパスに置き換えます。以下に例を示します。

```
nfs penguin.example.com:/export/cores
```

- **SSH** プロトコルを使用してクラッシュダンプをリモートマシンに保存するには、次の手順を実行します。

1. **#ssh user@my.server.com** の行頭にあるハッシュ記号 ("#") を削除します。
2. 値を正しいユーザー名およびホスト名に置き換えます。
3. **SSH** キーを設定に含めます。
   - **#sshkey /root/.ssh/kdump_id_rsa** の行頭にあるハッシュ記号 ("#") を削除します。
   - 値を、ダンプ先のサーバー上の正しいキーの場所に変更します。以下に例を示します。

```
ssh john@penguin.example.com
sshkey /root/.ssh/mykey
```

4.4. KDUMP コアコレクターの設定

kdump では、**core_collector** を使用してクラッシュダンプイメージをキャプチャーします。RHEL では、**makedumpfile** ユーティリティーがデフォルトのコアコレクターです。これは、以下に示すプロセスによりダンプファイルを縮小するのに役立ちます。

- クラッシュダンプファイルのサイズを圧縮し、さまざまなダンプレベルを使用して必要なページのみをコピーする
- 不要なクラッシュダンプページを除外する
- クラッシュダンプに含めるページタイプをフィルタリングする

構文

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

オプション

- **-c**、**-l**、または **-p**: **zlib** (**-c** オプションの場合)、**lzo** (**-l** オプションの場合)、または **snappy** (**-p** オプションの場合) のいずれかを使用して、ページごとに圧縮ダンプファイルの形式を指定します。
- **-d (dump_level)**: ページを除外して、ダンプファイルにコピーされないようにします。
- **--message-level**: メッセージタイプを指定します。このオプションで **message_level** を指定すると、出力の表示量を制限できます。たとえば、**message_level** で 7 を指定すると、一般的なメッセージとエラーメッセージを出力します。**message_level** の最大値は 31 です。

前提条件

- システムの root 権限がある。
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

1. **root** で、**/etc/kdump.conf** 設定ファイルを編集し、**#core_collector makedumpfile -l --message-level 1 -d 31** の行頭にあるハッシュ記号 ("#") を削除します。
2. クラッシュダンプファイルの圧縮を有効にするには、以下のコマンドを実行します。

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

-l オプションにより、**dump** の圧縮ファイル形式を指定します。**-d** オプションで、ダンプレベルを 31 に指定します。**--message-level** オプションで、メッセージレベルを 1 に指定します。

また、**-c** オプションおよび **-p** オプションを使用した以下の例を検討してください。

- **-c** を使用してクラッシュダンプファイルを圧縮するには、以下のコマンドを実行します。

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

- **-p** を使用してクラッシュダンプファイルを圧縮するには、以下のコマンドを実行します。

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

関連情報

- **makedumpfile(8)** の man ページ
- kdump の設定ファイル

4.5. KDUMP のデフォルト障害応答の設定

デフォルトでは、設定したターゲットの場所で **kdump** がクラッシュダンプファイルの作成に失敗すると、システムが再起動し、ダンプがプロセス内で失われます。この場合は、以下の手順を実施します。

前提条件

- root 権限
- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。

手順

1. **root** で、**/etc/kdump.conf** 設定ファイルの **#failure_action** の行頭にあるハッシュ記号 ("#") を削除します。
2. 値を任意のアクションに置き換えます。

```
failure_action poweroff
```

関連情報

- kdump ターゲットの設定

4.6. KDUMP 設定のテスト

kdump 設定をテストすると、設定が検証されます。また、指定したワークロードでクラッシュダンプが完了するまでにかかった時間が記録されます。

警告

kdump 設定をテストするコマンドにより、カーネルがクラッシュし、データが失われます。注意して指示に従い、**kdump** 設定のテストにアクティブな実稼働システムを使用しないでください。

手順

1. **kdump** を有効にしてシステムを再起動します。
2. **kdump** が有効かどうかを確認します。

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

3. カーネルクラッシュを強制的に実行します。

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

警告

このコマンドによりカーネルがクラッシュし、必要に応じてカーネルが再起動されます。

カーネルの再起動時に、**/etc/kdump.conf** ファイルで指定した場所 (デフォルトでは **/var/crash/**) に **address-YYYY-MM-DD-HH:MM:SS/vmcore** ファイルが作成されます。

関連情報

- kdump ターゲットの設定

第5章 KDUMP の有効化

この手順を使用すると、インストールされているすべてのカーネルまたは特定のカーネルに対して **kdump** サービスを有効または無効にすることができます。

5.1. インストールされているすべてのカーネルでの KDUMP の有効化

マシンにインストールされているすべてのカーネルに対して、**kdump** を有効にして起動できます。

前提条件

- 管理者権限

手順

1. インストールしたすべてのカーネルに **crashkernel=auto** コマンドラインパラメーターを追加します。

```
# grubby --update-kernel=ALL --args="crashkernel=auto"
```

2. **kdump** を有効にします。

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

検証

- **kdump** が実行されていることを確認します。

```
# 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. 特定のインストール済みカーネルでの KDUMP の有効化

マシン上の特定カーネルに対して、**kdump** を有効にできます。

前提条件

- 管理者権限

手順

1. マシンにインストールされているカーネルをリスト表示します。

```
# 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. 特定の **kdump** カーネルを、システムの GRUB (Grand Unified Bootloader) 設定ファイルに追加します。
以下に例を示します。

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

3. **kdump** を有効にします。

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

検証

- **kdump** が実行されていることを確認します。

```
# 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. KDUMP サービスの無効化

システムの起動時に **kdump** を無効にするには、以下の手順を行います。

前提条件

- **kdump** 設定とターゲットの要件をすべて満たしている。詳細は 対応している kdump 設定とターゲット を参照してください。
- **kdump** のインストール用のオプションがすべて、要件に応じて設定されている。詳細は、kdump のインストール を参照してください。

手順

1. 現在のセッションで **kdump** を停止するには、以下のコマンドを実行します。

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

2. **kdump** を無効にするには、以下を行います。

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

警告

kptr_restrict=1 を設定することが推奨されます。これにより、Kernel Address Space Layout (KASLR) が有効かどうかにかかわらず、**kdumpctl** はクラッシュカーネルを読み込みます。

トラブルシューティングの手順

kptr_restrict が (1) に設定されておらず、KASLR が有効になっている場合は、 **/proc/kcore** ファイルの内容がすべてゼロとして生成されます。したがって、**kdumpctl** サービスは **/proc/kcore** にアクセスしてクラッシュカーネルを読み込むことができません。

この問題を回避するために、**kptr_restrict=1** の設定を推奨する警告メッセージが **/usr/share/doc/kexec-tools/kexec-kdump-howto.txt** ファイルに表示されます。

kdumpctl サービスが必ずクラッシュカーネルを読み込むように、**kernel.kptr_restrict=1** が **sysctl.conf** ファイルに含まれていることを確認します。

関連情報

- systemd の管理

第6章 RHEL FOR REAL TIME バグの報告

推奨される RHEL for Real Time のバグ報告方法は、Red Hat Bugzilla でバグレポートを送信することです。バグを報告する前に、標準カーネルや RHEL for Real Time など、問題が発生したソースを特定すると役立ちます。

6.1. RHEL FOR REAL TIME バグの診断

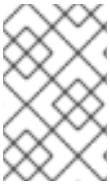
RHEL for Real Time または標準カーネルのどちらのカーネルが問題の原因であるかを特定すると、バグをより早く修正できる可能性が高くなります。手順に従うことで、バグレポートを送信する前に問題の原因を診断できます。

前提条件:

- RHEL for Real Time カーネルの最新バージョンがインストールされている。

手順:

1. RHEL for Real Time カーネルの最新バージョンを使用していることを確認します。
2. **GRUB** メニューを使用して、RHEL for Real Time カーネルを起動します。
3. 問題が発生した場合は、RHEL for Real Time に対するバグを報告してください。
4. 標準カーネルで問題を再現してみてください。
   このトラブルシューティングの手順は、問題の場所を特定するのに役立ちます。



注記

標準カーネルで問題が発生しない場合、バグはおそらく、Red Hat がベースライン (4.18.0) カーネルに適用した RHEL for Real Time 固有の拡張機能に導入された変更の結果です。

6.2. BUGZILLA でのバグレポートの送信

RHEL for Real Time に固有のバグを特定したら、次の手順に従って Bugzilla でバグレポートを送信します。

前提条件:

- Red Hat アカウントを持っている。

手順

1. Bugzilla アカウントにログインします。
2. **Enter A New Bug Report** をクリックします。
3. **Red Hat classification** を選択します。
4. **Red Hat Enterprise Linux** 製品を選択します。

5. **Component** を入力します。
   たとえば、バグがカーネルの問題である場合は、**kernel-rt** を使用します。あるいは、影響を受けるユーザースペースコンポーネントの名前を使用します (例: **rteval**)。

6. RHEL for Real Time カーネルのバグの問題の詳細な説明を提供します。
   問題の説明を入力するときは、標準の RHEL 8 カーネルで問題を再現できたかどうかを記述することもできます。

関連情報

- Red Hat Bugzilla – Red Hat Bugzilla アカウントの新規作成