Red Hat Software Collections 3

Red Hat Software Collections コンテナーイメージ

Red Hat Software Collections 3.8 コンテナーイメージの基本的な使用手順

Last Updated: 2024-02-27

Red Hat Software Collections 3 Red Hat Software Collections コンテナーイメージ

Red Hat Software Collections 3.8 コンテナーイメージの基本的な使用手順

Lenka Špačková
lspackova@redhat.com

Olga Tikhomirova

Robert Krátký

Vladimír Slávik

法律上の通知

概要

Red Hat Software Collections オファリングの一環として、Red Hat は、対応する Software Collections をベースとした多くのコンテナーイメージを提供しています。Red Hat Software Collections コンテナーイメージには、アプリケーション、Web サーバー、およびデータベースイメージが含まれます。本書では、Red Hat Software Collections で配信されるコンテナーイメージを取得、設定、および使用する方法を説明します。

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、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社 の CTO、Chris Wright のメッセージ を参照してください。

第1章 RED HAT SOFTWARE COLLECTIONS コンテナーイメージ

Red Hat Software Collections コンテナーイメージは、対応するコレクション、rhel7 または ubi7 のベースイメージに基づいています。Universal Base Images (UBI) の詳細は、Universal Base Images(UBI): イメージ、リポジトリー、パッケージ、およびソースコード を参照してください。

Red Hat Software Collections コンテナーイメージには、アプリケーション、デーモン、およびデータベースイメージが含まれます。Red Hat Software Collections コンテナーイメージの実行は、以下でサポートされます。

- Red Hat Enterprise Linux 7 Server
- Red Hat Enterprise Linux 7 Atomic Host
- Red Hat Enterprise Linux 8

Red Hat Enterprise Linux 7 の Software Collections として利用可能なコンポーネントの詳細は、Red Hat Software Collections および Red Hat Developer Toolset のドキュメントを参照してください。

Red Hat Software Collections コンテナーイメージは、以下の表で詳説されています。

- 3章*Red Hat Software Collections 3.8 に基づくコンテナーイメージ*
- 4章*Red Hat Software Collections 3.7 に基づくコンテナーイメージ*
- 5章*Red Hat Software Collections 3.6 に基づくコンテナーイメージ*
- 6章*Red Hat Software Collections 3.5 に基づいたコンテナーイメージ*
- 7章*Red Hat Software Collections 3.4 に基づくコンテナーイメージ*
- 8章*Red Hat Software Collections 3.3 に基づくコンテナーイメージ*
- 9章*Red Hat Software Collections 3.2 に基づいたコンテナーイメージ*
- 10章*Red Hat Software Collections 3.1 に基づいたコンテナーイメージ*
- 11章*Red Hat Software Collections 3.0 に基づくコンテナーイメージ*

Red Hat Ecosystem Catalog では、利用可能なコンテナーイメージを検索することもできます。

重要

Red Hat が提供する各コンテナーイメージの最新バージョンのみがサポートされます。

注記

コンテナー内のプロセスを制御する SELinux を使用する場合は、コンテナーにマウントされるボリュームのコンテンツが読み取り可能になり、ユースケースによっては書き込み可能であることを確認してください。詳細は podman の man ページ を参照してください。

関連資料

- コンテナーの使用

- コンテナーの管理
- OpenShift Enterprise アーキテクチャーの中核となる概念
- イメージ内の **/help.1** ファイルの Red Hat Software Collections コンテナーイメージの README ファイル、またはアップストリームの GitHub リポジトリー。

1.1. ビルダーイメージとしての RED HAT SOFTWARE COLLECTIONS コンテナーイメージ

Red Hat Software Collections コンテナーイメージをビルダーイメージとして使用してアプリケーションを構築、デプロイ、実行できます。一般的なユースケースに対応するために、以下の Source-to-Image (S2I) スクリプトがビルダーイメージに含まれています。

- イメージ内の **/usr/libexec/s2i/assemble** スクリプトを実行すると、アプリケーションアーティファクトで新しいイメージが生成されます。このスクリプトは、指定のアプリケーションのソースを取得して、イメージ内の適切なディレクトリーに配置します。アプリケーションソースに依存コンポーネントの定義が含まれる場合 (例: Python プロジェクトの場合に **PyPi** からのコンポーネントをリスト表示する **requirements.txt** など) には、コンポーネントはイメージにインストールされます。
- **/usr/libexec/s2i/run** スクリプトは、結果として生成されるコンテナーイメージのデフォルトコマンドとして設定されます (アプリケーションのアーティファクトを含む新規イメージ)。

podman を使用して、作成されたアプリケーションイメージを実行できます。手順は、コンテナーの使用 を参照してください。Red Hat Enterprise Linux 7 では、同じコマンドライン構文を使用する **podman** の代わりに **docker** コマンドを使用できます。

1.2. 既存のコンテナーイメージの拡張

Red Hat が提供するコンテナーイメージの機能を拡張するには、以下のオプションを使用してください。

- 環境変数を設定する。該当のコンテナーイメージについてのドキュメントを参照してください。
- OpenShift シークレット を使用する。
- カスタムアプリケーションイメージを構築する。手順は、*2章Red Hat Software Collections コンテナーイメージを使用したアプリケーションイメージの構築* を参照してください。
- OpenShift で Source-to-Image ビルドストラテジーを使用する。これにより、この機能をサポートするデーモンイメージ用に独自の設定ファイルを追加できます。該当のコンテナーイメージについてのドキュメントを参照してください。
- 他のデーモンまたはデータベースイメージの場合には、指定されたコンテナーイメージ上に新しいコンテナーを構築する。カスタム Dockerfile を作成し、FROM 句で元のコンテナーを使用します。対応するコンテナーイメージのドキュメントまたは、ナレッジベースの記事 How to Extend the rhscl/mariadb-101-rhel7 Container Image の**Build an application using a Dockerfile**のセクションを参照してください。

第2章 RED HAT SOFTWARE COLLECTIONS コンテナーイメージを使用したアプリケーションイメージの構築

Red Hat Software Collections コンテナーイメージを使用してアプリケーションイメージを構築する方法は複数あります。

- Red Hat が提供するコンテナーイメージをベースイメージとして使用する。
- S2I スクリプトを含む Dockerfile を使用する。
- OpenShift での **Source-to-Image** (S2I) の使用
- **source-to-image** ユーティリティーの使用

2.1. RED HAT SOFTWARE COLLECTIONS IMAGES を使用したアプリケーションイメージの構築

Red Hat が提供するコンテナーイメージをベースイメージとして使用するには、以下を実行します。

1. アプリケーションイメージ用に Dockerfile を作成し、このファイルに次の行が含まれるようにします。

```
FROM registry.redhat.io/rhscl_image_name
```

2. 以下の行を Dockerfile に配置して、**src**/ ディレクトリーにアプリケーションコードをイメージに追加します。

```
ADD src /opt/app-root/src
```

3. **podman** を使用してアプリケーションイメージをビルドします。

```
# podman build -t application_image_name .
```

4. **podman** を使用してアプリケーションイメージを実行します。たとえば、アプリケーションイメージ内でインタラクティブシェルを起動するには、以下を実行します。

```
# podman run -ti application_image_name /bin/bash -l
```

例2.1 rhscl/python-38-rhel7 ベースイメージを使用して Dockerfile から構築された Django アプリケーション

この例では、**rhscl/python-38-rhel7** コンテナーイメージからの単純な Django アプリケーションの作成に使用できる Dockerfile を示しています。

```
# Set base image
FROM registry.redhat.io/rhscl/python-38-rhel7

# Add application sources
ADD --chown=1001:0 app-src .

# Install the dependencies
RUN pip install -U "pip>=19.3.1" && \
```

```
    pip install -r requirements.txt && \
    python manage.py collectstatic --noinput && \
    python manage.py migrate

# Run the application
CMD python manage.py runserver 0.0.0.0:8080
```

関連資料

- Dockerfile からのイメージの構築
- Dockerfile リファレンスドキュメント

2.2. S2I スクリプトを使用した DOCKERFILE からのアプリケーションイメージの構築

Red Hat Software Collections コンテナーイメージをビルダーイメージとして使用し、ビルダーイメージに含まれる **assemble** および **run** S2I スクリプトを使用して、Dockerfile からアプリケーションイメージをビルドできます。**assemble** および **run** S2I スクリプトの詳細は、「ビルダーイメージとしての Red Hat Software Collections コンテナーイメージ」 を参照してください。

S2I スクリプトを使用して Dockerfile からアプリケーションイメージを作成するには、以下の手順に従います。

1. コンテナーレジストリーにログインします。

```
# podman login registry.redhat.io
```

2. ビルダーイメージをプルします。

```
# podman pull registry.redhat.io/rhscl_image_name
```

3. アプリケーションコードを準備します。
4. アプリケーションイメージのカスタム Dockerfile を作成し、以下を実行します。
   a. 以下の行を使用してビルダーイメージを定義します。

```
FROM registry.redhat.io/rhscl_image_name
```

   b. **src**/ ディレクトリーのアプリケーションソースをコンテナーに配置して、デフォルトのコンテナーユーザーがソースにアクセスするのに十分なパーミッションが指定されていることを確認します。

```
ADD --chown=1001:0 src /tmp/src
```

   c. **/usr/libexec/s2i/assemble** スクリプトを使用して依存関係をインストールします。

```
RUN /usr/libexec/s2i/assemble
```

   d. 作成されたイメージに、**/usr/libexec/s2i/run** スクリプトを使用して、デフォルトのコマンドを設定します。

```
CMD /usr/libexec/s2i/run
```

5. podman を使用してアプリケーションイメージをビルドします。

```
# podman build -t application_image_name .
```

6. podman を使用してアプリケーションイメージを実行します。たとえば、アプリケーションイメージ内でインタラクティブシェルを起動するには、以下を実行します。

```
# podman run -ti application_image_name /bin/bash -l
```

例2.2 S2I スクリプトを使用した Dockerfile からの Python 3.8 アプリケーションイメージの作成

この例は、ビルダーイメージによって提供される S2I スクリプトで Dockerfile から Python 3.8 アプリケーションをビルドし、実行する方法を示しています。

1. コンテナーレジストリーにログインします。

```
# podman login registry.redhat.io
```

2. ビルダーイメージをプルします。

```
# podman pull registry.redhat.io/rhscl/python-38-rhel7
```

3. https://github.com/sclorg/django-ex.git から入手できるアプリケーションコードをプルします。

```
$ git clone https://github.com/sclorg/django-ex.git app-src
```

または https://github.com/sclorg/s2i-python-container/tree/master/examples で入手できる例を使用します。

4. 以下の内容で Dockerfile を作成します。

```
FROM registry.redhat.io/rhscl/python-38-rhel7

# Add application sources to a directory that the assemble script expects them
# and set permissions so that the container runs without root access
USER 0
ADD app-src /tmp/src
RUN chown -R 1001:0 /tmp/src
USER 1001

# Install the dependencies
RUN /usr/libexec/s2i/assemble

# Set the default command for the resulting image
CMD /usr/libexec/s2i/run
```

5. 前の手順で準備した Dockerfile から新規イメージを構築します。

```
# podman build -t python-app .
```

6. 作成されたイメージを Python アプリケーションで実行します。

```
# podman run -d python-app
```

関連資料

- Dockerfile からのイメージの構築
- Dockerfile リファレンスドキュメント
- イメージの **/help.1** ファイルまたは、アップストリームの GitHub リポジトリー にある、該当のビルダーイメージの README ファイルの **Environment variables for Source-to-Image セクション**
- 環境変数は、Red Hat Ecosystem Catalog のイメージの詳細な説明にも記載されています。

2.3. OPENSHIFT で SOURCE-TO-IMAGE を使用したアプリケーションイメージのビルド

OpenShift の Source-to-Image (S2I) は、アプリケーションのソースコードを入力として取り、ビルダーの Red Hat Software Collections コンテナーイメージを使用し、アセンブルされたアプリケーションを出力として実行する新規イメージを生成することができるフレームワークです。

OpenShift で S2I を使用してアプリケーションを作成するには、以下を実行します。

1. OpenShift で利用可能なイメージを使用してアプリケーションを構築します。

```
$ oc new-app openshift_image_name~path_to_application_source_code
```

たとえば、OpenShift の **python:3.8** イメージストリームタグで利用可能なサポート対象のイメージを使用して Python 3.8 アプリケーションをビルドするには、以下を実行します。

```
$ oc new-app python:3.8~https://github.com/sclorg/django-ex.git
```

2. 利用可能な Pod (インスタンス) をリスト表示します。

```
$ oc get pods
```

3. localhost で選択した Pod を実行します。

```
$ oc exec pod -- curl 127.0.0.1:8080
```

関連資料

- OpenShift Container Platform ドキュメント
- S2I 要件
- GitHub の Source-to-image README ファイル
- 該当のビルダーイメージの README ファイルの Source-to-Image セクションの環境変数。

2.4. SOURCE-TO-IMAGE ユーティリティーを使用したアプリケーションイメージのビルド

Red Hat Software Collections オファリングでは、**source-to-image** ユーティリティーを提供し、Red Hat Enterprise Linux 7 サーバーでは OpenShift なしで使用できます。

注記

source-to-image は、Red Hat Enterprise Linux 7 でのみ利用でき、 **docker** がプルしたイメージとのみ機能します。**source-to-image** ユーティリティーで **podman** を使用することはできません。

ビルドプロセスは、以下の 3 つの要素で設定されており、これら 3 つの要素が最終的なコンテナーイメージに統合されます。

- アプリケーションのソースコード。プログラミング言語またはフレームワークで記述されています。
- **source-to-image** ユーティリティーを使用してイメージビルドをサポートする Red Hat Software Collections コンテナーイメージ。
- ビルダーイメージの一部である S2I スクリプト。これらのスクリプトの詳細は、「ビルダーイメージとしての Red Hat Software Collections コンテナーイメージ」 を参照してください。

ビルドプロセス時に、**source-to-image** ユーティリティーはソースコードおよびスクリプトを含む **.tar** ファイルを作成し、そのファイルをビルダーイメージにストリーミングします。

システムで **source-to-image** ユーティリティーを使用するには、以下を実行します。

1. Red Hat Software Collections にサブスクライブします。手順は、Red Hat Software Collections へのアクセス を参照してください。
2. **source-to-image** パッケージを提供する Red Hat Software Collections Server リポジトリーと、**source-to-image** で必要な **docker** パッケージが含まれる Red Hat Enterprise Linux 7 Server リポジトリーを有効にします。

```
# subscription-manager repos --enable rhel-server-rhscl-7-rpms --enable rhel-7-server-extras-rpms
```

3. **source-to-image** パッケージをインストールします。

```
# yum install source-to-image
```

4. コンテナーレジストリーにログインします。

```
# docker login registry.redhat.io
```

ビルダーイメージをプルします。

```
# docker pull registry.redhat.io/rhscl_image_name
```

アプリケーションのソースコードからアプリケーションイメージをビルドします。

```
# s2i build path_to_application_source_code_repository --context-
dir=source_code_context_directory application_image_name
```

5. **docker** を使用して、作成されるイメージを実行します。

例2.3 source-to-image ユーティリティーを使用した Git リポジトリーからの Python 3.8 アプリケーションのビルド

この例では、**rhscl/python-38-rhel7** ビルダーイメージおよび **source-to-image** ユーティリティーを使用して、公開 Git リポジトリーから利用可能なテストアプリケーションをビルドする方法を示しています。

1. コンテナーレジストリーにログインします。

```
# docker login registry.redhat.io
```

2. **rhscl/python-38-rhel7** ビルダーイメージをプルします。

```
# docker pull registry.redhat.io/rhscl/python-38-rhel7
```

3. **3.8/test/setup-test-app/** ディレクトリーの GitHub s2i-python リポジトリーからテストアプリケーションをビルドします。

```
# s2i build https://github.com/sclorg/s2i-python-container.git --context-dir=3.8/test/setup-
test-app/ registry.redhat.io/rhscl/python-38-rhel7 python-38-rhel7-app
```

これにより、新しいアプリケーションイメージ **python-38-rhel7-app** が作成されます。

4. 作成された **python-38-rhel7-app** イメージを実行します。

```
# docker run -d -p 8080:8080 --name example-app python-38-rhel7-app
```

5. http://localhost:8080/ から、作成されたサンプルドキュメントを取得します。

```
$ wget http://localhost:8080/
```

6. コンテナーを停止します。

```
# docker stop example-app
```

関連資料

- S2I 要件
- GitHub の Source-to-image README ファイル
- イメージの **/help.1** ファイルまたは、アップストリームの GitHub リポジトリー にある、該当のビルダーイメージの README ファイルの **Environment variables for Source-to-Image セクション**

第3章 RED HAT SOFTWARE COLLECTIONS 3.8 に基づくコンテナーイメージ

コンポーネント	詳細	サポート対象のアーキテクチャー
デーモンイメージ		
rhscl/nginx-120-rhel7	nginx 1.20 サーバーおよびリバースプロキシーサーバー	x86_64、s390x、ppc64le
データベースイメージ		
rhscl/redis-6-rhel7	Redis 6 キー値ストア	x86_64、s390x、ppc64le
Red Hat Developer Toolset イメージ		
rhscl/devtoolset-12-toolchain-rhel7 (2022 年 11 月以降に利用可能)	Red Hat Developer Toolset ツールチェーン	x86_64、s390x、ppc64le
rhscl/devtoolset-12-perftools-rhel7 (2022 年 11 月以降に利用可能)	Red Hat Developer Toolset perftools	x86_64、s390x、ppc64le
rhscl/devtoolset-11-toolchain-rhel7	Red Hat Developer Toolset ツールチェーン (EOL)	x86_64、s390x、ppc64le
rhscl/devtoolset-11-perftools-rhel7	Red Hat Developer Toolset perftools (EOL)	x86_64、s390x、ppc64le

説明:

- x86_64 - AMD64 および Intel 64 アーキテクチャー
- s390x - 64-bit IBM Z
- ppc64le - IBM POWER (リトルエンディアン)

すべてのイメージは、Red Hat Software Collections のコンポーネントに基づいています。Red Hat Enterprise Linux 7 用のイメージは、Red Hat コンテナーレジストリーから利用できます。

Red Hat Software Collections 3.8 が提供するコンポーネントの詳細は、Red Hat Software Collections 3.8 リリースノート を参照してください。

Red Hat Developer Toolset 11 のコンポーネントに関する詳細は、Red Hat Developer Toolset 11 ユーザーガイド を参照してください。

Red Hat Developer Toolset 12 のコンポーネントに関する詳細は、Red Hat Developer Toolset 12 ユーザーガイド を参照してください。

EOL イメージに対応しなくなりました。

第4章 RED HAT SOFTWARE COLLECTIONS 3.7 に基づくコンテナーイメージ

コンポーネント	詳細	サポートされているアーキテクチャー
アプリケーションイメージ		
rhscl/ruby-30-rhel7	アプリケーションを構築して実行する Ruby 3.0 プラットフォーム	x86_64、s390x、ppc64le
rhscl/ruby-27-rhel7	アプリケーションを構築して実行する Ruby 2.7 プラットフォーム (EOL)	x86_64、s390x、ppc64le
rhscl/ruby-26-rhel7	アプリケーションを構築して実行する Ruby 2.6 プラットフォーム (EOL)	x86_64、s390x、ppc64le
データベースイメージ		
rhscl/mariadb-105-rhel7	MariaDB 10.5 SQL データベースサーバー	x86_64、s390x、ppc64le
rhscl/postgresql-13-rhel7	PostgreSQL 13 SQL データベースサーバー	x86_64、s390x、ppc64le
Red Hat Developer Toolset イメージ		
rhscl/devtoolset-10-toolchain-rhel7	Red Hat Developer Toolset ツールチェーン (EOL)	x86_64、s390x、ppc64le
rhscl/devtoolset-10-perftools-rhel7	Red Hat Developer Toolset perftools (EOL)	x86_64、s390x、ppc64le

説明:

- x86_64 - AMD64 および Intel 64 アーキテクチャー
- s390x - 64-bit IBM Z
- ppc64le - IBM POWER (リトルエンディアン)

すべてのイメージは、Red Hat Software Collections のコンポーネントに基づいています。Red Hat Enterprise Linux 7 用のイメージは、Red Hat コンテナーレジストリーから利用できます。

Red Hat Software Collections 3.7 が提供するコンポーネントの詳細は、Red Hat Software Collections 3.7 リリースノート を参照してください。

Red Hat Developer Toolset 10 のコンポーネントに関する詳細は、Red Hat Developer Toolset 10 ユーザーガイド を参照してください。

Red Hat Software Collections 2 をベースとするコンテナーイメージの詳細は、Red Hat Software Collections 2 Container Images の使用 を参照してください。

EOL イメージに対応しなくなりました。

第5章 RED HAT SOFTWARE COLLECTIONS 3.6 に基づくコンテナーイメージ

コンポーネント	詳細	サポートされているアーキテクチャー
アプリケーションイメージ		
rhscl/nodejs-14-rhel7	アプリケーションを構築して実行する Node.js 14 プラットフォーム	x86_64、s390x、ppc64le
rhscl/perl-530-rhel7	アプリケーションを構築して実行する Perl 5.30 プラットフォーム	x86_64、s390x、ppc64le
rhscl/php-73-rhel7	アプリケーションを構築して実行する PHP 7.3 プラットフォーム	x86_64、s390x、ppc64le
rhscl/ruby-25-rhel7	アプリケーションを構築して実行する Ruby 2.5 プラットフォーム (EOL)	x86_64
デーモンイメージ		
rhscl/httpd-24-rhel7	Apache HTTP 2.4 サーバー	x86_64、s390x、ppc64le
rhscl/nginx-118-rhel7	nginx 1.18 サーバーおよびリバースプロキシーサーバー (EOL)	x86_64、s390x、ppc64le

説明:

- x86_64 - AMD64 および Intel 64 アーキテクチャー
- s390x - 64-bit IBM Z
- ppc64le - IBM POWER (リトルエンディアン)

すべてのイメージは、Red Hat Software Collections のコンポーネントに基づいています。Red Hat Enterprise Linux 7 用のイメージは、Red Hat コンテナーレジストリーから利用できます。

Red Hat Software Collections 3.6 が提供するコンポーネントの詳細は、Red Hat Software Collections 3.6 リリースノート を参照してください。

Red Hat Developer Toolset 10 のコンポーネントに関する詳細は、Red Hat Developer Toolset 10 ユーザーガイド を参照してください。

Red Hat Software Collections 2 をベースとするコンテナーイメージの詳細は、Red Hat Software Collections 2 Container Images の使用 を参照してください。

EOL イメージに対応しなくなりました。

第6章 RED HAT SOFTWARE COLLECTIONS 3.5 に基づいたコンテナーイメージ

コンポーネント	詳細	サポートされているアーキテクチャー
アプリケーションイメージ		
rhscl/python-38-rhel7	アプリケーションを構築して実行する Python 3.8 プラットフォーム	x86_64、s390x、ppc64le
デーモンイメージ		
rhscl/varnish-6-rhel7	Varnish Cache 6.0 HTTP リバースプロキシー	x86_64、s390x、ppc64le
Red Hat Developer Toolset Red Hat Developer Toolset イメージ		
rhscl/devtoolset-9-toolchain-rhel7	Red Hat Developer Toolset ツールチェーン (EOL)	x86_64、s390x、ppc64le
rhscl/devtoolset-9-perftools-rhel7	Red Hat Developer Toolset perftools (EOL)	x86_64、s390x、ppc64le

説明:

- x86_64 - AMD64 および Intel 64 アーキテクチャー
- s390x - 64-bit IBM Z
- ppc64le - IBM POWER (リトルエンディアン)

すべてのイメージは、Red Hat Software Collections のコンポーネントに基づいています。Red Hat Enterprise Linux 7 用のイメージは、Red Hat コンテナーレジストリーから利用できます。

Red Hat Software Collections 3.5 が提供するコンポーネントの詳細は、Red Hat Software Collections 3.5 リリースノート を参照してください。

Red Hat Developer Toolset 9.1 のコンポーネントに関する詳細は、Red Hat Developer Toolset 9 ユーザーガイド を参照してください。

Red Hat Software Collections 2 をベースとするコンテナーイメージの詳細は、Red Hat Software Collections 2 Container Images の使用 を参照してください。

EOL イメージに対応しなくなりました。

第7章 RED HAT SOFTWARE COLLECTIONS 3.4 に基づくコンテナーイメージ

コンポーネント	詳細	サポートされているアーキテクチャー
アプリケーションイメージ		
rhscl/nodejs-12-rhel7	アプリケーションを構築して実行する Node.js 12 プラットフォーム (EOL)	x86_64、s390x、ppc64le
デーモンイメージ		
rhscl/nginx-116-rhel7	nginx 1.16 サーバーおよびリバースプロキシーサーバー (EOL)	x86_64、s390x、ppc64le
データベースイメージ		
rhscl/postgresql-12-rhel7	PostgreSQL 12 SQL データベースサーバー	x86_64、s390x、ppc64le

説明:

- x86_64 - AMD64 および Intel 64 アーキテクチャー
- s390x - 64-bit IBM Z
- ppc64le - IBM POWER (リトルエンディアン)

すべてのイメージは、Red Hat Software Collections のコンポーネントに基づいています。Red Hat Enterprise Linux 7 用のイメージは、Red Hat コンテナーレジストリーから利用できます。

Red Hat Software Collections 3.4 が提供するコンポーネントの詳細は、Red Hat Software Collections 3.4 リリースノート を参照してください。

Red Hat Developer Toolset 9.0 のコンポーネントに関する詳細は、Red Hat Developer Toolset 9 ユーザーガイド を参照してください。

Red Hat Software Collections 2 をベースとするコンテナーイメージの詳細は、Red Hat Software Collections 2 Container Images の使用 を参照してください。

EOL イメージに対応しなくなりました。

第8章 RED HAT SOFTWARE COLLECTIONS 3.3 に基づくコンテナーイメージ

コンポーネント	詳細	サポート対象のアーキテクチャー
データベースイメージ		
rhscl/mariadb-103-rhel7	MariaDB 10.3 SQL データベースサーバー (EOL)	x86_64、s390x、ppc64le
rhscl/redis-5-rhel7	Redis 5 キー値ストア (EOL)	x86_64、s390x、ppc64le
Red Hat Developer Toolset イメージ		
rhscl/devtoolset-8-toolchain-rhel7	Red Hat Developer Toolset ツールチェーン (EOL)	x86_64、s390x、ppc64le
rhscl/devtoolset-8-perftools-rhel7	Red Hat Developer Toolset perftools (EOL)	x86_64、s390x、ppc64le

説明:

- x86_64 - AMD64 および Intel 64 アーキテクチャー
- s390x - 64-bit IBM Z
- ppc64le - IBM POWER (リトルエンディアン)

すべてのイメージは、Red Hat Software Collections のコンポーネントに基づいています。Red Hat Enterprise Linux 7 用のイメージは、Red Hat コンテナーレジストリーから利用できます。

Red Hat Software Collections 3.3 が提供するコンポーネントの詳細は、Red Hat Software Collections 3.3 リリースノート を参照してください。

Red Hat Developer Toolset 8.1 のコンポーネントに関する詳細は、Red Hat Developer Toolset 8 ユーザーガイド を参照してください。

Red Hat Software Collections 2 をベースとするコンテナーイメージの詳細は、Red Hat Software Collections 2 Container Images の使用 を参照してください。

EOL イメージに対応しなくなりました。

第9章 RED HAT SOFTWARE COLLECTIONS 3.2 に基づいたコンテナーイメージ

コンポーネント	詳細	サポートされているアーキテクチャー
アプリケーションイメージ		
rhscl/nodejs-10-rhel7	アプリケーション (EOL) を構築して実行する Node.js 10 プラットフォーム	x86_64、s390x、ppc64le
rhscl/php-72-rhel7	アプリケーション (EOL) を構築して実行する PHP 7.2 プラットフォーム	x86_64、s390x、ppc64le
デーモンイメージ		
rhscl/nginx-114-rhel7	nginx 1.14 サーバーおよびリバースプロキシーサーバー (EOL)	x86_64、s390x、ppc64le
データベースイメージ		
rhscl/mysql-80-rhel7	MySQL 8.0 SQL データベースサーバー	x86_64、s390x、ppc64le

説明:

- x86_64 - AMD64 および Intel 64 アーキテクチャー
- s390x - 64-bit IBM Z
- ppc64le - IBM POWER (リトルエンディアン)

すべてのイメージは、Red Hat Software Collections のコンポーネントに基づいています。Red Hat Enterprise Linux 7 用のイメージは、Red Hat コンテナーレジストリーから利用できます。

Red Hat Software Collections 3.2 が提供するコンポーネントの詳細は、Red Hat Software Collections 3.2 リリースノート を参照してください。

Red Hat Developer Toolset 8.0 のコンポーネントに関する詳細は、Red Hat Developer Toolset 8 ユーザーガイド を参照してください。

Red Hat Software Collections 2 をベースとするコンテナーイメージの詳細は、Red Hat Software Collections 2 Container Images の使用 を参照してください。

EOL イメージに対応しなくなりました。

第10章 RED HAT SOFTWARE COLLECTIONS 3.1 に基づいたコンテナーイメージ

コンポーネント	詳細	サポートされているアーキテクチャー
アプリケーションイメージ		
rhscl/php-70-rhel7	アプリケーション (EOL) を構築して実行する PHP 7.0 プラットフォーム	x86_64
rhscl/perl-526-rhel7	アプリケーションを構築して実行する Perl 5.26 プラットフォーム (EOL)	x86_64
デーモンイメージ		
rhscl/varnish-5-rhel7	Varnish Cache 5.0 HTTP リバースプロキシー (EOL)	x86_64、s390x、ppc64le
データベースイメージ		
rhscl/mongodb-36-rhel7	MongoDB 3.6 NoSQL データベースサーバー (EOL)	x86_64
rhscl/postgresql-10-rhel7	PostgreSQL 10 SQL データベースサーバー	x86_64、s390x、ppc64le
Red Hat Developer Toolset イメージ		
rhscl/devtoolset-7-toolchain-rhel7	Red Hat Developer Toolset ツールチェーン (EOL)	x86_64、s390x、ppc64le
rhscl/devtoolset-7-perftools-rhel7	Red Hat Developer Toolset perftools (EOL)	x86_64、s390x、ppc64le

説明:

- x86_64 - AMD64 および Intel 64 アーキテクチャー
- s390x - 64-bit IBM Z
- ppc64le - IBM POWER (リトルエンディアン)

すべてのイメージは、Red Hat Software Collections のコンポーネントに基づいています。Red Hat Enterprise Linux 7 用のイメージは、Red Hat コンテナーレジストリーから利用できます。

Red Hat Software Collections 3.1 が提供するコンポーネントの詳細は、Red Hat Software Collections 3.1 リリースノート を参照してください。

Red Hat Developer Toolset 7.1 のコンポーネントに関する詳細は、Red Hat Developer Toolset 7 ユーザーガイド を参照してください。

Red Hat Software Collections 2 をベースとするコンテナーイメージの詳細は、Red Hat Software Collections 2 Container Images の使用 を参照してください。

EOL イメージに対応しなくなりました。

第11章 RED HAT SOFTWARE COLLECTIONS 3.0 に基づくコンテナーイメージ

コンポーネント	詳細	サポートされているアーキテクチャー
アプリケーションイメージ		
rhscl/nodejs-8-rhel7	アプリケーション (EOL) を構築して実行する Node.js 8 プラットフォーム	x86_64、s390x、ppc64le
rhscl/php-71-rhel7	アプリケーション (EOL) を構築して実行する PHP 7.1 プラットフォーム	x86_64
rhscl/python-36-rhel7	アプリケーションを構築して実行する Python 3.6 プラットフォーム	x86_64、s390x、ppc64le
デーモンイメージ		
rhscl/nginx-112-rhel7	Nginx 1.12 サーバーおよびリバースプロキシーサーバー (EOL)	x86_64、s390x、ppc64le
データベースイメージ		
rhscl/mariadb-102-rhel7	MariaDB 10.2 SQL データベースサーバー (EOL)	x86_64
rhscl/mongodb-34-rhel7	MongoDB 3.4 NoSQL データサーバー (EOL)	x86_64
rhscl/postgresql-96-rhel7	PostgreSQL 9.6 SQL データベースサーバー (EOL)	x86_64

説明:

- x86_64 - AMD64 および Intel 64 アーキテクチャー
- s390x - 64-bit IBM Z
- ppc64le - IBM POWER (リトルエンディアン)

すべてのイメージは、Red Hat Software Collections のコンポーネントに基づいています。Red Hat Enterprise Linux 7 用のイメージは、Red Hat コンテナーレジストリーから利用できます。

Red Hat Software Collections 3.0 が提供するコンポーネントの詳細は、Red Hat Software Collections 3.0 リリースノート を参照してください。

Red Hat Developer Toolset 7.0 のコンポーネントに関する詳細は、Red Hat Developer Toolset 7 ユーザーガイド を参照してください。

Red Hat Software Collections 2 をベースとするコンテナーイメージの詳細は、Using Red Hat Software Collections 2 Container Images を参照してください。

EOL イメージに対応しなくなりました。

第12章 アプリケーションイメージ

12.1. NODE.JS

12.1.1. 説明

rhscl/nodejs-14-rhel7 イメージは、アプリケーションを構築して実行する Node.js 14 プラットフォームを提供します。

12.1.2. アクセス

rhscl/nodejs-14-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/nodejs-14-rhel7
```

12.1.3. 設定

環境変数を設定するには、ソースコードリポジトリー内の **.s2i/environment** ファイルにキーと値のペアとして配置できます。

変数名	詳細
NODE_ENV	NodeJS ランタイムモード (デフォルト:production)
DEV_MODE	true に設定すると、**nodemon** は作業中にサーバーを自動的に再読み込みするために使用されます (デフォルト:false)。**DEV_MODE** を true に設定すると、**NODE_ENV** のデフォルトが development に変更されます (明示的に設定されていない場合)。
NPM_RUN	**package.json** ファイルの **スクリプト** セクション (デフォルト: npm run "start") で定義されている代替/カスタムランタイムモードを選択します。**DEV_MODE** が使用されている間は、これらのユーザー定義の run-scripts は利用できません。
HTTP_PROXY	アセンブリー時に npm プロキシーを使用
HTTPS_PROXY	アセンブリー時に npm プロキシーを使用
NPM_MIRROR	ビルドプロセス中にカスタム NPM レジストリーミラーを使用してパッケージをダウンロードします。

12.2. PHP

12.2.1. 詳細

rhscl/php-73-rhel7 イメージは、アプリケーションをビルドして実行する PHP 7.3 プラットフォームを提供します。**npm** を使用する **Node.js** は、PHP イメージに事前インストールされます。

12.2.2. アクセス

rhscl/php-73-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/php-73-rhel7
```

12.2.3. 設定

環境変数を設定するには、ソースコードリポジトリー内の **.s2i/environment** ファイルにキーと値のペアとして配置します。

以下の環境変数は、**php.ini** ファイルに同等のプロパティー値を設定します。

変数名	説明	デフォルト
ERROR_REPORTING	PHP が、アクションを実行するエラー、警告、および通知を PHP に通知します。	**E_ALL & ~E_NOTICE**
DISPLAY_ERRORS	PHP がエラー、通知、および arning を出力するかどうか、および場所を制御します。	**ON**
DISPLAY_STARTUP_ERRORS	表示エラーとは別に処理するために PHP の起動シーケンス中に発生する表示エラーの原因	**OFF**
TRACK_ERRORS	最後のエラー/警告メッセージを **$php_errormsg** (ブール値) に保存します。	**OFF**
HTML_ERRORS	エラーに関連するドキュメントへのリンクエラー	**ON**
INCLUDE_PATH	PHP ソースファイルのパス	**.:/opt/app-root/src:/opt/rh/rh-php73/root/usr/share/pear**
PHP_MEMORY_LIMIT	メモリー制限	128M
SESSION_NAME	セッション名	**PHPSESSID**
SESSION_HANDLER	セッションの保存方法	**files**
SESSION_PATH	セッションデータファイルの場所	**/tmp/sessions**
SESSION_COOKIE_DOMAIN	Cookie が有効であるドメイン	

変数名	説明	デフォルト
SESSION_COOKIE_HTTPONLY	httpOnly フラグをクッキーに追加するかどうか	**0**
SESSION_COOKIE_SECURE	Cookie をセキュアな接続でのみ送信すべきかどうかを指定します。	**OFF**
SHORT_OPEN_TAG	<? タグと ?> タグ間のコードが PHP によって認識されるかどうかを決定します。	**OFF**
DOCUMENTROOT	アプリケーションの DocumentRoot を定義するパス (例: /public)	/

必要に応じて、**rh-php7*** Software Collection のバージョンを置き換えます。

以下の環境変数は、**opcache.ini** ファイルに同等のプロパティー値を設定します。

変数名	説明	デフォルト
OPCACHE_MEMORY_CONSUMPTION	OPcache 共有メモリーストレージサイズ (メガバイト単位)	128
OPCACHE_REVALIDATE_FREQ	更新のスクリプトタイムスタンプをチェックする頻度 (秒単位)。0 を指定すると、すべてのリクエストの更新が OPcache チェックされます。	2
OPCACHE_MAX_FILES	OPcache ハッシュテーブルの鍵の最大数 (スクリプト)。200 から 1000000 までの数字のみが許可されます。	4000

以下を設定して、PHP 設定の読み込みに使用されるディレクトリー全体を上書きすることもできます。

変数名	説明
PHPRC	**php.ini** ファイルへのパスを設定します。
PHP_INI_SCAN_DIR	追加の ini 設定ファイルをスキャンするパス

Apache MPM のプレフォーク 設定を上書きして、PHP アプリケーションのパフォーマンスを向上させることができます。Cgroup の制限を設定すると、イメージは最適な値を自動的に設定しようとします。値を独自に指定すると、いつでも上書きできます。

変数名	説明	デフォルト
HTTPD_START_SERVERS	StartServers ディレクティブは、起動時に作成された子サーバープロセスの数を設定します。	8
HTTPD_MAX_REQUEST_WORKERS	MaxRequestWorkers ディレクティブは、サービスされる同時リクエストの数に制限を設定します。	256 (これは、**TOTAL_MEMORY / 15MB** 式を使用してコンテナーに Cgroup 制限を設定するすることで自動的に調整されます)15MB は、1 つの **httpd** プロセスの平均サイズです。

カスタムの composer リポジトリーミラー URL を使用して、デフォルトの **packagist.org** の代わりにパッケージをダウンロードできます。

変数名	詳細
COMPOSER_MIRROR	カスタムの composer リポジトリーミラー URL を composer 設定に追加します。注記: これは、**composer.json** に記載されているパッケージにのみ影響します。
COMPOSER_INSTALLER	https://getcomposer.org/installer の Composer をダウンロードするためのデフォルトの URL を上書きします。非接続環境で役に立ちます。
COMPOSER_ARGS	**composer install** コマンドラインに引数を追加します (例: **--no-dev**)。

アプリケーションの DocumentRoot がソースディレクトリー **/opt/app-root/src** 内の入れ子になっている場合、ユーザーは独自の **.htaccess** ファイルを提供できます。これにより、Apache の動作のオーバーライドが可能になり、アプリケーションリクエストの処理方法を指定できます。**.htaccess** ファイルは、アプリケーションソースの root に置く必要があります。**.htaccess** の詳細は Apache HTTP Server Tutorial を参照してください。

12.2.4. イメージの拡張

PHP イメージは、source-to-image を使用して拡張できます。

たとえば、**~/image-configuration/** ディレクトリーの設定を使用してカスタマイズされた PHP イメージ **my-php-rhel7** を構築するには、以下のコマンドを実行します。

```
$ s2i build ~/image-configuration/ rhscl/php-73-rhel7 my-php-rhel7
```

ソースイメージのバージョンを適宜変更してください。

アプリケーションの構造は以下の例のようになります。

ディレクトリー名	詳細
./httpd-cfg	追加の Apache 設定ファイル (***.conf**) を含めることができます。
./httpd-ssl	独自の SSL 証明書 (**certs/** サブディレクトリー内) とキー (**private/** サブディレクトリー) を含めることができます。
./php-pre-start	**httpd** の起動前にソースとなるシェルスクリプト (***.sh**) を含めることができます。
./php-post-assemble	**assemble** スクリプトの末尾で読み込まれるシェルスクリプト (***.sh**) を含めることができます。
./	アプリケーションのソースコード

12.3. PERL

12.3.1. 詳細

rhscl/perl-530-rhel7 イメージは、アプリケーションをビルドして実行する Perl 5.30 プラットフォームを提供します。Perl Web アプリケーションをデプロイするための **mod_perl** を使用する **Apache httpd 2.4** と、**npm** を使用した **Node.js** が事前にインストールされます。

これらのイメージは、Perl Web Server Gateway Interface (PSGI) アプリケーションのデプロイメントもサポートします。

12.3.2. アクセス

rhscl/perl-530-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/perl-530-rhel7
```

12.3.3. 設定

環境変数を設定するには、ソースコードリポジトリー内の **.s2i/environment** ファイルにキーと値のペアとして配置できます。

変数名	説明	デフォルト
ENABLE_CPAN_TEST	指定したすべての cpan パッケージのインストールと、そのテストの実行を許可します。	**false**

変数名	説明	デフォルト
CPAN_MIRROR	依存関係をインストールするために cpanminus によって使用されるミラー URL を指定します。	デフォルトでは URL が指定されていません。
PERL_APACHE2_RELOAD	変更した Perl モジュールの自動読み込みを有効にします。	**false**
HTTPD_START_SERVERS	StartServers ディレクティブは、起動時に作成された子サーバープロセスの数を設定します。	**8**
HTTPD_MAX_REQUEST_WORKERS	Apache によって処理される同時リクエストの数	**256** ですが、メモリーが制限されている場合は自動的に降格されます。
PSGI_FILE	PSGI アプリケーションファイルへの相対パスを指定します。空の値を使用して PSGI auto-configuration を無効にします。	存在する場合は、最上位ディレクトリーに単一の *.psgi ファイル
PSGI_URI_PATH	PSGI アプリケーションによって処理される URI パスを指定します。	/

Comprehensive Perl Archive Network (CPAN) から追加の Perl モジュールをインストールするには、アプリケーションソースのルートディレクトリーに **cpanfile** を作成します。このファイルは、Module-CPANFile CPAN ディストリビューションで定義されている **cpanfile** 形式に準拠する必要があります。cpanfile 形式の詳細は、cpanfile のドキュメント を参照してください。

Apache httpd の動作を変更するには、必要に応じて **.htaccess** ファイルをアプリケーションソースツリーから削除します。**.htaccess** の詳細は Apache HTTP Server Tutorial を参照してください。

12.4. PYTHON

12.4.1. 詳細

rhscl/python-38-rhel7 イメージは、アプリケーションをビルドして実行する Python 3.8 プラットフォームを提供します。**npm** を使用する **Node.js** は事前にインストールされます。

12.4.2. アクセス

rhscl/python-38-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/python-38-rhel7
```

12.4.3. 設定

環境変数を設定するには、ソースコードリポジトリー内の **.s2i/environment** ファイルにキーと値のペアとして配置できます。

変数名	詳細
APP_SCRIPT	スクリプトファイルからアプリケーションを実行するのに使用されます。これは、スクリプトファイルへのパス (null に設定されない限りデフォルトは **app.sh**) へのパスである必要があります。
APP_FILE	Python スクリプトからアプリケーションを実行するために使用されます。これは、アプリケーションを起動するために Python インタープリターに渡される Python ファイル (デフォルトが **app.py**) へのパスでなければなりません。
APP_MODULE	以下に記載されているように、Gunicorn を使用してアプリケーションを実行するために使用されます。この変数は、**MODULE_NAME:VARIABLE_NAME** パターンで WSGI 呼び出し可能なアプリケーションを指定します。ここで、**MODULE_NAME** はモジュールの完全ドットパスで、**VARIABLE_NAME** は指定されたモジュール内で WSGI 呼び出し可能なアプリケーションを参照します。Gunicorn は、指定がない場合は WSGI 呼び出し可能なアプリケーションを探します。**APP_MODULE** が指定されていない場合、run スクリプトはプロジェクトで **wsgi.py** ファイルを検索し、それが存在する場合は使用します。アプリケーションのインストールに **setup.py** を使用している場合は、**MODULE_NAME** の部分をそこから読み込むことができます。たとえば、**setup-test-app** を参照してください。
APP_HOME	この変数を使用して、アプリケーションを実行するサブディレクトリーを指定することもできます。この変数が参照するディレクトリーには、**wsgi.py** (Gunicorn 用) または **manage.py** (Django 用) を含める必要があります。**APP_HOME** が指定されていない場合は、**assemble** スクリプトおよび **run** スクリプトはアプリケーションのルートディレクトリーを使用します。
APP_CONFIG	Gunicorn 設定ファイルを使用した有効な Python ファイルへのパス。
DISABLE_MIGRATE	この変数を空でない値に設定し、生成されたイメージの実行時に **manage.py migrate** の実行を無効にします。これは Django プロジェクトにのみ影響します。

変数名	詳細
DISABLE_COLLECTSTATIC	この変数を空でない値に設定し、ビルド中に **manage.py collectstatic** の実行を無効にします。これは Django プロジェクトにのみ影響します。
DISABLE_SETUP_PY_PROCESSING	この変数を空でない値に設定し、**setup.py** スクリプトの処理を省略するには、**requirements.txt** の **-e .** を使用して処理をトリガーするか、アプリケーションを site-packages ディレクトリーにインストールしないようにします。
ENABLE_PIPENV	この変数を設定して、高レベルの Python パッケージツールの Pipenv を使用して、アプリケーションの依存関係を管理します。これは、プロジェクトに適切にフォーマットされた **Pipfile** および **Pipfile.lock** が含まれる場合にのみ使用してください。
ENABLE_INIT_WRAPPER	この変数を空でない値に設定し、init ラッパーを使用します。これは、Django 開発サーバーや Tornado など、ゾンビプロセスを取得できないサーバーの場合に役立ちます。このオプションは、**APP_SCRIPT** 変数または **APP_FILE** 変数と共に使用することができます。ゾンビプロセスの処理を正しく取得するため、これは **APP_MODULE** で使用される Gunicorn に適用されません。
PIP_INDEX_URL	この変数は、カスタムインデックスの URL またはミラーを使用して、ビルドプロセス時に必要なパッケージをダウンロードするように設定します。これは、requirements.txt に記載されているパッケージにのみ影響します。
UPGRADE_PIP_TO_LATEST	この変数を空でない値に設定し、Python パッケージをインストールする前に **pip** プログラムを最新バージョンにアップグレードします。設定されていない場合は、Python バージョン用にプラットフォームに含まれるデフォルトバージョンが使用されます。
WEB_CONCURRENCY	ワーカー 数のデフォルト設定を変更する場合は、このパラメーターを設定します。デフォルトでは、これは利用可能なコア数 2 に設定されます。

12.5. RUBY

12.5.1. 説明

rhscl/ruby-30-rhel7 イメージは、アプリケーションをビルドして実行する Ruby 3.0 プラットフォームを提供します。**rhscl/ruby-27-rhel7** イメージは Ruby 2.7 プラットフォームを提供します。

npm を使用する **Node.js** は事前にインストールされます。

12.5.2. アクセス

rhscl/ruby-30-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/ruby-30-rhel7
```

rhscl/ruby-27-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/ruby-27-rhel7
```

12.5.3. 設定

環境変数を設定するには、ソースコードリポジトリー内の **.s2i/environment** ファイルにキーと値のペアとして配置できます。

変数名	説明
RACK_ENV	この変数は、Ruby アプリケーションが (上書きされない限り) デプロイされる環境 (**production**、**development**、**test**) を指定します。各レベルは、ロギングの詳細度、エラーページ、Ruby gem インストールなどに関して動作が異なります。アプリケーションアセットは、**RACK_ENV** が **production** に設定されている場合にのみコンパイルされることに注意してください。
DISABLE_ASSET_COMPILATION	**true** に設定されたこの変数は、アセットコンパイルプロセスがスキップされることを示します。これは、アプリケーションが **production** 環境で実行される場合のみ実行されるため、アセットがすでにコンパイルされている場合のみ使用してください。
PUMA_MIN_THREADS、**PUMA_MAX_THREADS**	これらの変数は、Puma のスレッドプールで利用可能な最小および最大のスレッドを示します。
PUMA_WORKERS	この変数は、起動する worker プロセスの数を示します。Puma の クラスターモード に関するドキュメントを参照してください。
RUBYGEM_MIRROR	この変数を設定して、カスタム RubyGems のミラー URL を使用して、ビルドプロセス中に必要な gem パッケージをダウンロードします。

S2I スクリプトを機能させるには、アプリケーションの Gemfile に **puma** または **rack** gem を含める必要があります。

第13章 デーモンイメージ

13.1. APACHE HTTP サーバー

13.1.1. 説明

rhscl/httpd-24-rhel7 イメージは、Apache HTTP 2.4 サーバーを提供します。イメージは、Apache HTTP Web サーバーに基づく他のアプリケーションのベースイメージとして使用できます。

13.1.2. アクセス

rhscl/httpd-24-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/httpd-24-rhel7
```

rhscl/httpd-24-rhel7 イメージは、S2I ツールの使用をサポートしています。

13.1.3. 設定と使用方法

Apache HTTP Server コンテナーイメージは、**podman run** コマンドで **-e** オプションを使用して設定できる以下の設定変数をサポートします。

変数名	説明
HTTPD_LOG_TO_VOLUME	デフォルトでは、**httpd** は標準出力にログインするため、**podman logs** コマンドを使用してログにアクセスできます。**HTTPD_LOG_TO_VOLUME** が設定されていると、**httpd** は **/var/log/httpd24** にログインし、コンテナーボリュームを使用してホストシステムにマウントできます。このオプションは、コンテナーが UID **0** として実行される場合に許可されます。
HTTPD_MPM	この変数を設定して、デフォルトの Multi-Processing Module (MPM) を、パッケージのデフォルト MPM から変更できます。

イメージを実行し、ログファイルをコンテナーボリュームとしてホストの **/wwwlogs** にマウントするには、以下のコマンドを実行します。

```
$ podman run -d -u 0 -e HTTPD_LOG_TO_VOLUME=1 --name httpd -v /wwwlogs:/var/log/httpd24:Z
rhscl/httpd-24-rhel7
```

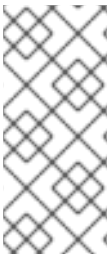
(デフォルトの **prefork** ではなく) **イベント** MPM を使用してイメージを実行するには、以下のコマンドを実行します。

```
$ podman run -d -e HTTPD_MPM=event --name httpd rhscl/httpd-24-rhel7
```

-v /host:/container オプションを **podman run** コマンドに渡すと、以下のマウントポイントを設定することもできます。

ボリュームマウントポイント	説明
/var/www	Apache HTTP Server データディレクトリー
/var/log/httpd24	Apache HTTP Server ログディレクトリー (root で実行している場合のみ利用可能)

ホストからコンテナーにディレクトリーをマウントする場合は、マウントされたディレクトリーに適切なパーミッションがあり、ディレクトリーの所有者とグループが、コンテナー内で実行中のユーザー UID または名前と一致していることを確認します。



注記

rhscl/httpd-24-rhel7 コンテナーイメージは、OpenShift の source-to-image ストラテジー内で正常に機能するためにデフォルトの UID として **1001** を使用するようになりました。また、コンテナーイメージはデフォルトで **8080** ポートをリッスンします。以前は、**rhscl/httpd-24-rhel7** コンテナーイメージはデフォルトでポート **80** でリッスンし、UID **0** として実行されていました。

rhscl/httpd-24-rhel7 コンテナーイメージを UID **0** として実行するには、**podman run** コマンドに **-u 0** オプションを指定します。

```
podman run -u 0 rhscl/httpd-24-rhel7
```

13.2. NGINX

13.2.1. 説明

rhscl/nginx-120-rhel7 イメージは、nginx 1.20 サーバーとリバースプロキシーサーバーを提供します。このイメージは、nginx 1.20 Web サーバーをベースとした他のアプリケーションのベースイメージとして使用できます。**rhscl/nginx-118-rhel7** イメージは nginx 1.18 を提供します。

13.2.2. アクセス

rhscl/nginx-120-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/nginx-120-rhel7
```

rhscl/nginx-118-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/nginx-118-rhel7
```

13.2.3. 設定

nginx コンテナーイメージは、**podman run** コマンドで **-e** オプションを使用して設定できる以下の設定変数に対応します。

変数名	説明
NGINX_LOG_TO_VOLUME	デフォルトでは、nginx は標準出力にログインするため、**podman logs** コマンドを使用してログにアクセスできます。**NGINX_LOG_TO_VOLUME** を設定すると、nginx は **/var/opt/rh/rh-nginx120/log/nginx/** または **/var/opt/rh/rh-nginx120/log/nginx/** にログを記録します。これは、コンテナーボリュームを使用してホストシステムにマウントできます。

rhscl/nginx-120-rhel7 イメージおよび **rhscl/nginx-118-rhel7** イメージは、S2I ツールの使用をサポートしています。

13.3. VARNISH CACHE

13.3.1. 詳細

rhscl/varnish-6-rhel7 イメージは、HTTP リバースプロキシーである Varnish Cache 6.0 を提供します。

13.3.2. アクセス

rhscl/varnish-6-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/varnish-6-rhel7
```

13.3.3. 設定

その他の設定は必要ありません。

S2I ツールを使用した Red Hat Software Collections Varnish Cache イメージのサポートS2I がアクセスするディレクトリーの **default.vcl** 設定ファイルは、VCL 形式である必要があります。

第14章 データベースイメージ

14.1. MARIADB

14.1.1. 説明

rhscl/mariadb-105-rhel7 イメージは、MariaDB 10.5 SQL データベースサーバーを提供します。

14.1.2. アクセス

rhscl/mariadb-105-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/mariadb-105-rhel7
```

14.1.3. 設定と使用方法

使用方法と設定は MySQL イメージと同じです。デーモンの名前は **mysqld** で、すべての環境変数の名前が MySQL と同じであることに注意してください。

イメージは、**-e VAR=VALUE** オプションを **podman run** コマンドに渡して初期化中に設定できる以下の環境変数を認識します。

変数名	説明
MYSQL_USER	作成される MySQL アカウントのユーザー名
MYSQL_PASSWORD	ユーザーアカウントのパスワード
MYSQL_DATABASE	データベース名
MYSQL_ROOT_PASSWORD	root ユーザーのパスワード (任意)
MYSQL_CHARSET	デフォルト文字セット (オプション)
MYSQL_COLLATION	デフォルトの表記 (オプション)

注記

root ユーザーはデフォルトでパスワードを設定しておらず、ローカル接続のみを許可します。このオプションを設定するには、コンテナーの初期化時に **MYSQL_ROOT_PASSWORD** 環境変数を設定します。これにより、**root** アカウントにリモートでログインできるようになります。ローカル接続にはパスワードは必要ありません。リモートの **root** アクセスを無効にするには、**MYSQL_ROOT_PASSWORD** の設定を解除して、コンテナーを再起動するだけです。

重要

パスワードはイメージ設定の一部であるため、権限のないユーザー (**MYSQL_USER**) のパスワードを変更するためにサポートされる唯一の方法は、**root** ユーザーは環境変数 **MYSQL_PASSWORD** および **MYSQL_ROOT_PASSWORD** をそれぞれ変更する方法のみです。SQL ステートメントまたは他の方法でデータベースパスワードを変更すると、変数に格納された値と実際のパスワードが一致しなくなります。データベースコンテナーが起動するたびに、パスワードは環境変数に保存されている値にリセットされます。

以下の環境変数は MySQL 設定ファイルに影響しますが、すべて任意になります。

変数名	説明	デフォルト
MYSQL_LOWER_CASE_TABLE_NAMES	テーブル名の保存方法と比較方法を設定します。	**0**
MYSQL_MAX_CONNECTIONS	同時クライアント接続の最大許容数	151
MYSQL_MAX_ALLOWED_PACKET	1 つのパケットまたは生成/中間文字列の最大サイズ。	200M
MYSQL_FT_MIN_WORD_LEN	FULLTEXT インデックスに含まれる単語の最小長	4
MYSQL_FT_MAX_WORD_LEN	FULLTEXT インデックスに含まれる単語の最大長	20
MYSQL_AIO	ネイティブ AIO が破損した場合に **innodb_use_native_aio** 設定値を制御します。http://help.directadmin.com/item.php?id=529 を参照してください。	1
MYSQL_TABLE_OPEN_CACHE	すべてのスレッドのオープンテーブルの数	400
MYSQL_KEY_BUFFER_SIZE	インデックスブロックに使用されるバッファーのサイズ	32m (または利用可能なメモリーの 10%)
MYSQL_SORT_BUFFER_SIZE	ソートに使用されるバッファーサイズ	256K
MYSQL_READ_BUFFER_SIZE	連続スキャンに使用されるバッファーのサイズ	8M (または利用可能なメモリーの 5%)
MYSQL_INNODB_BUFFER_POOL_SIZE	InnoDB がテーブルおよびインデックスデータをキャッシュするバッファープールのサイズ	32m (または利用可能なメモリーの 50%)

変数名	説明	デフォルト
MYSQL_INNODB_LOG_FILE_SIZE	ロググループ内の各ログファイルのサイズ	8M (または使用可能なメモリーの 15%)
MYSQL_INNODB_LOG_BUFFER_SIZE	InnoDB がディスクのログファイルへの書き込みに使用するバッファーのサイズ	8M (または使用可能なメモリーの 15%)
MYSQL_DEFAULTS_FILE	代替の設定ファイルを参照します。	**/etc/my.cnf**
MYSQL_BINLOG_FORMAT	set は binlog 形式を設定します。サポートされる値は **row** および **statement** です。	**statement**

--memory パラメーターセットを使用して MariaDB イメージを実行すると、パラメーターが明示的に指定されていない限り、以下のパラメーターの値は利用可能なメモリーに基づいて自動的に算出されます。

変数名	デフォルトのメモリーパーセンテージ
MYSQL_KEY_BUFFER_SIZE	10%
MYSQL_READ_BUFFER_SIZE	5%
MYSQL_INNODB_BUFFER_POOL_SIZE	50%
MYSQL_INNODB_LOG_FILE_SIZE	15%
MYSQL_INNODB_LOG_BUFFER_SIZE	15%

-v /host:/container オプションを **podman run** コマンドに渡すと、以下のマウントポイントを設定することもできます。

ボリュームマウントポイント	説明
/var/lib/mysql/data	MySQL データディレクトリー

注記

ホストからコンテナーにディレクトリーをマウントする場合は、マウントされたディレクトリーに適切なパーミッションがあり、ディレクトリーの所有者とグループが、コンテナー内で実行中のユーザー UID または名前と一致していることを確認します。

14.1.4. イメージの拡張

How to Extend the rhscl/mariadb-101-rhel7 Container Image を参照してください。これは、**rhscl/mariadb-105-rhel7** にも該当します。

14.2. MYSQL

14.2.1. 詳細

rhscl/mysql-80-rhel7 イメージは、MySQL 8.0 SQL データベースサーバーを提供します。

14.2.2. アクセスと使用方法

rhscl/mysql-80-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/mysql-80-rhel7
```

必須の環境変数のみを設定し、データベースをホストディレクトリーに保存しないようにするには、以下のコマンドを実行します。

```
# podman run -d --name mysql_database -e MYSQL_USER=<user> -e
MYSQL_PASSWORD=<pass> \
  -e MYSQL_DATABASE=<db> -p 3306:3306 rhscl/mysql-80-rhel7
```

これにより、データベース db を使用する MySQL を実行する **mysql_database** という名前のコンテナーと、認証情報 **user:pass** を持つユーザーが作成されます。ポート **3306** が公開され、ホストにマッピングされます。コンテナーの実行後もデータベースを永続化する必要がある場合には、**-v /host/db/path:/var/lib/mysql/data** 引数も追加します。**/host/db/path** ディレクトリーは MySQL データディレクトリーになります。

データベースディレクトリーが初期化されていない場合、エントリーポイントスクリプトは最初に **mysql_install_db** を実行し、必要なデータベースユーザーおよびパスワードを設定します。データベースを初期化するか、すでに存在する場合は **mysqld** が実行され、PID **1** として実行されます。**podman stop mysql_database** コマンドを実行して、接続解除したコンテナーを停止できます。

14.2.3. 設定

イメージは、**-e VAR=VALUE** を **podman run** コマンドに渡して初期化中に設定できる以下の環境変数を認識します。

変数名	説明
MYSQL_USER	作成される MySQL アカウントのユーザー名
MYSQL_PASSWORD	ユーザーアカウントのパスワード
MYSQL_DATABASE	データベース名
MYSQL_ROOT_PASSWORD	root ユーザーのパスワード (任意)

注記

root ユーザーはデフォルトでパスワードを設定しておらず、ローカル接続のみを許可します。このオプションを設定するには、コンテナーの初期化時に **MYSQL_ROOT_PASSWORD** 環境変数を設定します。これにより、root アカウントにリモートでログインできるようになります。ローカル接続にはパスワードは必要ありません。リモートの root アクセスを無効にするには、**MYSQL_ROOT_PASSWORD** の設定を解除して、コンテナーを再起動するだけです。

重要

パスワードはイメージ設定の一部であるため、権限のないユーザー (**MYSQL_USER**) のパスワードを変更するためにサポートされる唯一の方法は、root ユーザーは環境変数 **MYSQL_PASSWORD** および **MYSQL_ROOT_PASSWORD** をそれぞれ変更する方法のみです。SQL ステートメントまたは他の方法でデータベースパスワードを変更すると、変数に格納された値と実際のパスワードが一致しなくなります。データベースコンテナーが起動するたびに、パスワードは環境変数に保存されている値にリセットされます。

以下の環境変数は MySQL 設定ファイルに影響しますが、すべて任意になります。

変数名	説明	デフォルト
MYSQL_LOWER_CASE_TABLE_NAMES	テーブル名の保存方法と比較方法を設定します。	**0**
MYSQL_MAX_CONNECTIONS	同時クライアント接続の最大許容数	151
MYSQL_MAX_ALLOWED_PACKET	1 つのパケットまたは生成/中間文字列の最大サイズ。	200M
MYSQL_FT_MIN_WORD_LEN	FULLTEXT インデックスに含まれる単語の最小長	4
MYSQL_FT_MAX_WORD_LEN	FULLTEXT インデックスに含まれる単語の最大長	20
MYSQL_AIO	ネイティブ AIO が破損した場合に **innodb_use_native_aio** 設定値を制御します。http://help.directadmin.com/item.php?id=529 を参照してください。	1
MYSQL_TABLE_OPEN_CACHE	すべてのスレッドのオープンテーブルの数	400
MYSQL_KEY_BUFFER_SIZE	インデックスブロックに使用されるバッファーのサイズ	32m (または利用可能なメモリーの 10%)

変数名	説明	デフォルト
MYSQL_SORT_BUFFER_SIZE	ソートに使用されるバッファーサイズ	256K
MYSQL_READ_BUFFER_SIZE	連続スキャンに使用されるバッファーのサイズ	8M (または利用可能なメモリーの 5%)
MYSQL_INNODB_BUFFER_POOL_SIZE	InnoDB がテーブルおよびインデックスデータをキャッシュするバッファープールのサイズ	32m (または利用可能なメモリーの 50%)
MYSQL_INNODB_LOG_FILE_SIZE	ロググループ内の各ログファイルのサイズ	8M (または使用可能なメモリーの 15%)
MYSQL_INNODB_LOG_BUFFER_SIZE	InnoDB がディスクのログファイルへの書き込みに使用するバッファーのサイズ	8M (または使用可能なメモリーの 15%)
MYSQL_DEFAULTS_FILE	代替の設定ファイルを参照します。	**/etc/my.cnf**
MYSQL_BINLOG_FORMAT	set は、binlog 形式を設定します。サポートされる値は **row** および **statement** です。	**statement**
MYSQL_LOG_QUERIES_ENABLED	クエリーロギングを有効にするには、この変数を **1** に設定します。	**0**

MySQL イメージが **--memory** パラメーターセットを使用して実行されると、以下のパラメーターが明示的に指定されていない限り、そのパラメーターの値は利用可能なメモリーに基づいて自動的に算出されます。

変数名	デフォルトのメモリーパーセンテージ
MYSQL_KEY_BUFFER_SIZE	10%
MYSQL_READ_BUFFER_SIZE	5%
MYSQL_INNODB_BUFFER_POOL_SIZE	50%
MYSQL_INNODB_LOG_FILE_SIZE	15%
MYSQL_INNODB_LOG_BUFFER_SIZE	15%

-v /host:/container オプションを **podman run** コマンドに渡すと、以下のマウントポイントを設定することもできます。

ボリュームマウントポイント	説明
/var/lib/mysql/data	MySQL データディレクトリー

注記

ホストからコンテナーにディレクトリーをマウントする場合は、マウントされたディレクトリーに適切なパーミッションがあり、ディレクトリーの所有者とグループが、コンテナー内で実行中のユーザー UID または名前と一致していることを確認します。

14.3. POSTGRESQL

14.3.1. 詳細

rhscl/postgresql-13-rhel7 イメージは、PostgreSQL 13 SQL データベースサーバーを提供します。**rhscl/postgresql-12-rhel7** イメージは、PostgreSQL 12 サーバーを提供します。**rhscl/postgresql-10-rhel7** イメージは、PostgreSQL 10 サーバーを提供します。

14.3.2. アクセスと使用方法

rhscl/postgresql-13-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/postgresql-13-rhel7
```

rhscl/postgresql-12-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/postgresql-12-rhel7
```

rhscl/postgresql-10-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/postgresql-10-rhel7
```

必須の環境変数のみを設定し、データベースをホストディレクトリーに保存しないようにするには、以下のコマンドを実行します。

```
# podman run -d --name postgresql_database -e POSTGRESQL_USER=<user> \
  -e POSTGRESQL_PASSWORD=<pass> -e POSTGRESQL_DATABASE=<db> \
  -p 5432:5432 <image_name>
```

これにより、データベース db を使用する MySQL を実行する **postgresql_database** という名前のコンテナーと、認証情報 **user:pass** を持つユーザーが作成されます。ポート **5432** が公開され、ホストにマッピングされます。コンテナーの実行後もデータベースを永続化する必要がある場合には、**-v /host/db/path:/var/lib/pgsql/data** 引数も追加します。これは、PostgreSQL データベースクラスターのディレクトリーになります。

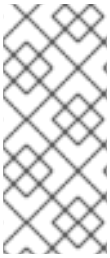
データベースクラスターディレクトリーが初期化されていない場合、エントリーポイントスクリプトは最初に **initdb** を実行し、必要なデータベースユーザーおよびパスワードを設定します。データベースを初期化するか、すでに存在する場合は、**postgres** が実行され、**PID 1** として実行されます。**podman stop postgresql_database** コマンドを実行して、デタッチされたコンテナーを停止できます。

postgres デーモンは、まずログを標準出力に書き込みます。コンテナーイメージのログを確認するには、**podman logs <image_name>** コマンドを使用します。ログの出力はロギングコレクタープロセスにリダイレクトされ、**pg_log/** ディレクトリーに表示されます。

14.3.3. 設定

イメージは、**-e VAR=VALUE** を **podman run** コマンドに渡して初期化中に設定できる以下の環境変数を認識します。

変数名	説明
POSTGRESQL_USER	作成される PostgreSQL アカウントのユーザー名
POSTGRESQL_PASSWORD	ユーザーアカウントのパスワード
POSTGRESQL_DATABASE	データベース名
POSTGRESQL_ADMIN_PASSWORD	postgres 管理アカウントのパスワード (任意)



注記

postgres 管理者アカウントには、デフォルトでパスワードが設定されず、ローカル接続のみが許可されます。コンテナーを初期化するときに **POSTGRESQL_ADMIN_PASSWORD** 環境変数を設定することで設定できます。これにより、**postgres** アカウントにリモートでログインすることができます。ローカル接続にはパスワードは必要ありません。

重要

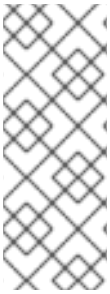
パスワードはイメージ設定の一部であるため、データベースユーザーと postgres 管理ユーザーのパスワードを変更するためにサポートされている唯一の方法は、環境変数 **POSTGRESQL_PASSWORD** および **POSTGRESQL_ADMIN_PASSWORD** をそれぞれ変更することです。SQL ステートメントまたは前述の環境変数以外の方法でデータベースのパスワードを変更すると、変数に格納されている値と実際のパスワードが一致しなくなります。データベースのコンテナーイメージを起動するたびに、パスワードは環境変数に保存されている値にリセットされます。

以下のオプションは移行に関するものです。

変数名	説明	デフォルト
POSTGRESQL_MIGRATION_REMOTE_HOST	移行元となるホスト名/IP	
POSTGRESQL_MIGRATION_ADMIN_PASSWORD	リモートの **postgres** 管理ユーザーのパスワード	
POSTGRESQL_MIGRATION_IGNORE_ERRORS	オプション: sql のインポートエラーを無視する	いいえ

以下の環境変数は PostgreSQL 設定ファイルに影響があり、すべて任意です。

変数名	説明	デフォルト
POSTGRESQL_MAX_CONNECTIONS	許可されるクライアント接続の最大数。これにより、準備済みトランザクションの最大数も設定します。	100
POSTGRESQL_MAX_PREPARED_TRANSACTIONS	準備状態にあるトランザクションの最大数を設定します。準備済みトランザクションを使用している場合は、max_connections よりも大きい値にする必要があります。	0
POSTGRESQL_SHARED_BUFFERS	データのキャッシュに使用する PostgreSQL 専用のメモリー容量を設定します。	32M
POSTGRESQL_EFFECTIVE_CACHE_SIZE	オペレーティングシステムおよびデータベース自体によるディスクキャッシュで利用できるメモリーの推定値に設定します。	128M

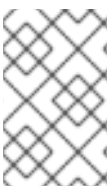


注記

PostgreSQL イメージが **--memory** パラメーターセットで実行され、**POSTGRESQL_SHARED_BUFFERS** および **POSTGRESQL_EFFECTIVE_CACHE_SIZE** に値が指定されていない場合は、**--memory** パラメーターで提供される値をもとにこれらの値が自動的に算出されます。値はアップストリームの式に基づいて計算され、指定されたメモリーの 1/4 および 1/2 に設定されます。

-v /host:/container オプションを **podman run** コマンドに渡すと、以下のマウントポイントを設定することもできます。

ボリュームマウントポイント	説明
/var/lib/pgsql/data	PostgreSQL データベースクラスターのディレクトリー



注記

ホストからコンテナーにディレクトリーをマウントする場合は、マウントされたディレクトリーに適切なパーミッションがあり、ディレクトリーの所有者とグループが、コンテナー内で実行中のユーザー UID または名前と一致していることを確認します。

podman run コマンドで **-u** オプションを使用しない限り、コンテナーのプロセスは通常 UID **26** で実行されます。データディレクトリーのパーミッションを変更するには、以下のコマンドを使用します。

```
$ setfacl -m u:26:-wx /your/data/dir
$ podman run <...> -v /your/data/dir:/var/lib/pgsql/data:Z <...>
```

14.3.4. データの移行

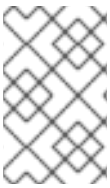
PostgreSQL コンテナーイメージは、リモート PostgreSQL サーバーからのデータの移行をサポートします。以下のコマンドを使用して、イメージ名を変更し、必要に応じて任意の設定変数を追加します。

```
$ podman run -d --name postgresql_database \
    -e POSTGRESQL_MIGRATION_REMOTE_HOST=172.17.0.2 \
    -e POSTGRESQL_MIGRATION_ADMIN_PASSWORD=remoteAdminP@ssword \
    [ OPTIONAL_CONFIGURATION_VARIABLES ]
    rhscl/postgresql-12-rhel7
```

移行は、ダンプと復元の方法を行います (リモートクラスターに対して **pg_dumpall** を実行し、**psql** でローカルにダンプをインポート)。プロセスがストリーミング (unix パイプライン) されるため、このプロセス中に作成された中間ダンプファイルは、追加のストレージ領域を消費しないようにします。

適用中に SQL コマンドの一部が失敗すると、移行スクリプトのデフォルト動作も失敗し、スクリプト化された無人移行の結果がすべてまたはゼロであることを確認してください。同じ原則を使用して作成された以前のバージョンの PostgreSQL サーバーコンテナーから移行する場合 (たとえば、**rhscl/postgresql-10-rhel7** から **rhscl/postgresql-12-rhel7** への移行)、最も一般的なケースでは、正常な移行が期待されます (保証はされません)。異なる種類の PostgreSQL コンテナーイメージからの移行に失敗する可能性があります。

このすべてまたはゼロの原則が適切ではない場合は、任意の **POSTGRESQL_MIGRATION_IGNORE_ERRORS** オプションがあり、このオプションでベストエフォートの移行が行われます。ただし、データの一部が失われ、ユーザーが標準エラー出力を確認し、移行後の時間に手動で問題を修正する必要があります。



注記

コンテナーイメージはユーザーの利便性のための移行に役立ちますが、完全に自動移行は保証されません。そのため、データベースの移行に進む前に、データをすべて移行するために手動の手順を実施する必要があります。

移行シナリオで **POSTGRESQL_USER** などの変数を使用しない場合があります。データベース、ロール、パスワードに関する情報を含むすべてのデータは、古いクラスターからコピーされます。古い PostgreSQL コンテナーイメージの初期化に使用するオプションの設定変数を使用するようにしてください。リモートクラスターでデフォルト以外の設定が行われる場合は、設定ファイルも手動でコピーする必要がある場合があります。

警告

以前の PostgreSQL クラスターと新しい PostgreSQL クラスター間の IP 通信はデフォルトでは暗号化されていません。リモートクラスターで SSL を設定するか、異なる手段を使用してセキュリティーを確保するかはユーザー次第となります。

14.3.5. データベースのアップグレード

警告

データディレクトリーのアップグレードを実施する前に、すべてのデータのバックアップを作成していることを確認してください。アップグレードに失敗した場合は、手動でロールバックが必要になる場合があります。

PostreSQL イメージは、以前の rhscl イメージによって提供される PostgreSQL サーバーバージョンによって作成されたデータディレクトリーの自動アップグレードをサポートします。たとえば、**rhscl/postgresql-13-rhel7** イメージは **rhscl/postgresql-12-rhel7** からのアップグレードをサポートします。アップグレードプロセスは、イメージ A からイメージ B へ切り替え、**$POSTGRESQL_UPGRADE** 変数を適切に設定して、データベースデータ変換を明示的に要求できるように設計されています。

アップグレードプロセスは、**pg_upgrade** バイナリーを使用して内部的に実装され、コンテナーには 2 つのバージョンの PostgreSQL サーバーを含める必要があります (詳細は、**pg_upgrade** の man ページを参照してください)。

pg_upgrade プロセスおよび新しいサーバーバージョンについては、新しいデータディレクトリーを初期化する必要があります。このデータディレクトリーは、通常、外部のバインドマウントポイントである **/var/lib/pgsql/data/** ディレクトリーのコンテナーツールによって自動的に作成されます。**pg_upgrade** の実行は、ダンプと復元のアプローチと似ています。これは、(コンテナー内の) 古い PostgreSQL サーバーと新しい PostgreSQL サーバーの両方を起動し、古いデータディレクトリーをダンプし、同時に新しいデータディレクトリーに復元します。この操作には、データファイルのコピーが多数必要です。選択したアップグレードのタイプに応じて **$POSTGRESQL_UPGRADE** 変数を設定します。

copy	データファイルは古いデータディレクトリーから新しいディレクトリーにコピーされます。このオプションは、アップグレードに失敗した場合にデータ損失のリスクが低くなります。
hardlink	データファイルは、以前のデータディレクトリーから新しいデータディレクトリーにハードリンクされるため、パフォーマンスが最適化されます。ただし、障害が発生した場合でも、古いディレクトリーは使用できなくなります。

注記

コピーしたデータ用に十分な容量があることを確認してください。領域が不十分なためにアップグレードが失敗すると、データが失われる可能性があります。

14.3.6. イメージの拡張

PostgreSQL イメージは、source-to-image を使用して拡張できます。

たとえば、**~/image-configuration/** ディレクトリーの設定が含まれるカスタマイズされた **new-postgresql** イメージをビルドするには、以下のコマンドを使用します。

```
$ s2i build ~/image-configuration/ postgresql new-postgresql
```

S2I ビルドに渡されるディレクトリーには、以下のディレクトリーを 1 つ以上含める必要があります。

postgresql-pre-start/	コンテナーを起動する初期段階で、このディレクトリーから ***.sh** ファイルが読み込まれます。バックグラウンドで実行している PostgreSQL デーモンはありません。
postgresql-cfg/	含まれる設定ファイル (***.conf**) は、イメージの **postgresql.conf** ファイルの最後に含まれます。
postgresql-init/	含まれるシェルスクリプト (***.sh**) は、(**initdb** の実行に成功した後に) データベースが新規に初期化されると読み込まれます。これにより、データディレクトリーは空ではなくなります。これらのスクリプトを利用する際に、ローカルの PostgreSQL サーバーが稼働している状態です。永続データディレクトリーを含む再デプロイメントのシナリオでは、スクリプトは読み込まれません (no-op)。
postgresql-start/	**postgresql-init/** 同様ですが、これらのスクリプトは常に読み込まれます (**postgresql-init/** スクリプトが存在する場合はその後に)。

S2I ビルドでは、提供されたすべてのファイルが新しいイメージの **/opt/app-root/src/** ディレクトリーにコピーされます。カスタマイズには同じ名前のファイルのみがカスタマイズでき、ユーザーが提供するファイルは **/usr/share/container-scripts/** ディレクトリーのデフォルトファイルよりも優先されるため、上書きできます。

14.4. REDIS

14.4.1. 説明

rhscl/redis-6-rhel7 イメージは、高度なキー/値ストア Redis 6 を提供します。

14.4.2. アクセス

rhscl/redis-6-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/redis-6-rhel7
```

14.4.3. 設定と使用方法

必須の環境変数のみを設定し、データベースをホストディレクトリーに保存しないようにするには、次のコマンドを実行します。

```
# podman run -d --name redis_database -p 6379:6379 rhscl/redis-6-rhel7
```

このコマンドにより、**redis_database** という名前のコンテナーが作成されます。ポート **6379** が公開され、ホストにマッピングされます。

以下の環境変数は Redis 設定ファイルに影響しますが、任意です。

変数名	説明
REDIS_PASSWORD	サーバーアクセスのパスワード

パスワードを設定するには、次のコマンドを実行します。

```
# podman run -d --name redis_database -e REDIS_PASSWORD=strongpassword rhscl/redis-6-rhel7
```

重要

Redis は高速であり、ブルートフォース攻撃のターゲットとなる可能性があるため、非常に強力なパスワードを使用します。

コンテナーの実行後もデータベースを永続化するには、**podman run** コマンドに **-v /host/db/path:/var/lib/redis/data:Z** オプションを追加します。

ボリュームマウントポイント	説明
/var/lib/redis/data	Redis データディレクトリー

注記

ホストからコンテナーにディレクトリーをマウントする場合は、マウントされたディレクトリーに適切なパーミッションがあり、ディレクトリーの所有者とグループが、コンテナー内で実行中のユーザー UID または名前と一致していることを確認します。

コンテナーイメージのログを確認するには、**podman logs <image_name>** コマンドを使用します。

第15章 RED HAT DEVELOPER TOOLSET イメージ

Red Hat Developer Toolset は、Red Hat Enterprise Linux プラットフォームの開発者向けの Red Hat 製品です。Red Hat Enterprise Linux の複数のバージョンにインストールおよび使用できる、完全な開発ツールおよびパフォーマンス分析ツールを提供します。また、Red Hat Developer Toolset ツールチェーンで構築された実行ファイルは、複数のバージョンの Red Hat Enterprise Linux にデプロイおよび実行できます。互換性の詳細は、Red Hat Developer Toolset 12 ユーザーガイド を参照してください。

重要

最新バージョンの Red Hat Developer Toolset を提供するコンテナーイメージのみがサポートされます。

15.1. ビルド済みのコンテナーイメージから RED HAT DEVELOPER TOOLSET ツールの実行

ローカルマシンにすでにプルしたビルド済みの Red Hat Developer Toolset コンテナーイメージの一般的な使用情報を表示するには、**root** で以下のコマンドを実行します。

```
# podman run image_name usage
```

ビルド済みのコンテナーイメージ内でインタラクティブなシェルを起動するには、**root** で以下のコマンドを実行します。

```
# podman run -ti image_name /bin/bash -l
```

上記のコマンドの両方で、**image_name** パラメーターを、ローカルシステムにプルして使用するコンテナーイメージの名前に置き換えます。

たとえば、選択したツールチェーンコンポーネントでコンテナーイメージ内でインタラクティブなシェルを起動するには、**root** で以下のコマンドを実行します。

```
# podman run -ti rhscl/devtoolset-12-toolchain-rhel7 /bin/bash -l
```

例15.1 ビルド済みの Red Hat Developer Toolset Toolchain イメージでの GCC の使用

この例では、Red Hat Developer Toolset の選択したツールチェーンコンポーネントでビルド済みコンテナーイメージを取得および起動する方法と、そのイメージ内で **gcc** コンパイラーの実行方法を説明します。

1. **Managing Containers** ドキュメントの Using podman to work with containers の手順に従って、コンテナー環境が正しく設定されていることを確認してください。
2. 公式の Red Hat コンテナーレジストリーから、ビルド済みの Red Hat Developer Toolset コンテナーイメージをプルします。

   ```
   # podman pull rhscl/devtoolset-12-toolchain-rhel7
   ```

3. インタラクティブシェルでコンテナーイメージを起動するには、次のコマンドを発行します。

   ```
   # podman run -ti rhscl/devtoolset-12-toolchain-rhel7 /bin/bash -l
   ```

4. コンテナーを通常の (root 以外の) ユーザーとして起動するには、**sudo** コマンドを使用します。ホストシステムからコンテナーのファイルシステムにディレクトリーをマップするには、**podman** コマンドに **-v** (または **--volume**) オプションを追加します。

```
$ sudo podman run -v ~/Source:/src -ti rhscl/devtoolset-12-toolchain-rhel7 /bin/bash -l
```

上記のコマンドでは、ホストの ~/**Source**/ ディレクトリーがコンテナー内の /**src**/ ディレクトリーとしてマウントされます。

5. コンテナーのインタラクティブシェルに移動したら、予想通りに Red Hat Developer Toolset ツールを実行できます。たとえば、**gcc** コンパイラーのバージョンを確認するには、次のコマンドを実行します。

```
bash-4.2$ gcc -v
[...]
gcc version 12.2.1 20221121 (Red Hat 12.2.1-4) (GCC)
```

関連情報

Red Hat Developer Toolset で利用可能なコンポーネントの詳細は、以下のオンラインリソースを参照してください。

- Red Hat Developer Toolset 12 ユーザーガイド
- Red Hat Developer Toolset 12.1 リリースノート
- Red Hat Developer Toolset 12.0 リリースノート

15.2. RED HAT DEVELOPER TOOLSET ツールチェーンコンテナーイメージ

15.2.1. 説明

Red Hat Developer Toolset Toolchain イメージは、GNU コンパイラーコレクション (GCC) および GNU デバッガー (GDB) を提供します。

rhscl/devtoolset-12-toolchain-rhel7 イメージには、以下のパッケージに対応するコンテンツが含まれます。

コンポーネント	バージョン	パッケージ
gcc	12.2.1	devtoolset-12-gcc
g++		devtoolset-12-gcc-c++
gfortran		devtoolset-12-gcc-gfortran
gdb	11.2	devtoolset-12-gdb

さらに、**devtoolset-12-binutils** パッケージが依存関係として含まれています。

15.2.2. アクセス

rhscl/devtoolset-12-toolchain-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/devtoolset-12-toolchain-rhel7
```

15.3. RED HAT DEVELOPER TOOLSET PERFORMANCE TOOLS コンテナーイメージ

15.3.1. 説明

Red Hat Developer Toolset Performance Tools イメージは、プロファイリングおよびパフォーマンス測定ツールを多数提供します。

rhscl/devtoolset-12-perftools-rhel7 イメージには以下のコンポーネントが含まれます。

コンポーネント	バージョン	パッケージ
dwz	0.14	devtoolset-12-dwz
Dyninst	12.1.0	devtoolset-12-dyninst
elfutils	0.187	devtoolset-12-elfutils
ltrace	0.7.91	devtoolset-12-ltrace
make	4.3	devtoolset-12-make
memstomp	0.1.5	devtoolset-12-memstomp
OProfile	1.4.0	devtoolset-12-oprofile
strace	5.18	devtoolset-12-strace
SystemTap	4.7	devtoolset-12-systemtap
Valgrind	3.19.0	devtoolset-12-valgrind

さらに、**devtoolset-12-gcc** と **devtoolset-12-binutils** パッケージは依存関係として含まれています。

15.3.2. アクセス

rhscl/devtoolset-12-perftools-rhel7 イメージをプルするには、**root** で以下のコマンドを実行します。

```
# podman pull registry.redhat.io/rhscl/devtoolset-12-perftools-rhel7
```

15.3.3. 使用方法

コンテナーイメージからの SystemTap ツールの使用

SystemTap ツールをコンテナーイメージから使用している場合、追加の設定が必要で、特別なコマンドラインオプションを指定してコンテナーを実行する必要があります。

以下の 3 つの条件を満たす必要があります。

1. このイメージは、スーパーユーザー権限で実行する必要があります。これを実行するには、以下のコマンドを実行してイメージを実行します。

   ```
   ~]$ podman run --ti --privileged --ipc=host --net=host --pid=host devtoolset-12-my-perftools
   /bin/bash -l
   ```

   ビルド済みの **perftools** イメージを使用するには、上記のコマンドで **devtoolset-12-perftools-rhel7** をイメージ名に置き換えます。

2. 以下のカーネルパッケージをコンテナーにインストールする必要があります。

   - **kernel**
   - **kernel-devel**
   - **kernel-debuginfo**
     上記のパッケージのバージョン番号およびリリース番号は、ホストシステムで実行しているカーネルのバージョン番号およびリリース番号と一致する必要があります。以下のコマンドを実行して、ホストシステムのカーネルのバージョンおよびリリース番号を確認します。

     ```
     ~]$ uname -r
     3.10.0-1160.90.1.el7.x86_64
     ```

     kernel-debuginfo パッケージは **Debug** リポジトリーでのみ利用できることに注意してください。**rhel-7-server-debug-rpms** リポジトリーを有効にします。debuginfo パッケージにアクセスする方法は、RHEL システムで debuginfo パッケージをダウンロードまたはインストールする を参照してください。

     必要なパッケージの正しいバージョンをインストールするには、**yum** パッケージマネージャーと **uname** コマンドの出力を使用します。たとえば、正しいバージョンの **kernel** をインストールするには、**root** で以下のコマンドを実行します。

     ```
     ~]# yum install -y kernel-$(uname -r)
     ```

3. **podman commit** コマンドを実行して、コンテナーを再利用可能なイメージに保存します。カスタムビルドの **SystemTap**コンテナーを保存するには、以下を実行します。

   ```
   ~]$ podman commit devtoolset-12-systemtap-$(uname -r)
   ```

第16章 COMPILER TOOLSET イメージ

Red Hat Developer Tools のコンテナーイメージは、AMD64 および Intel 64、64 ビット IBM Z、および IBM POWER のリトルエンディアンアーキテクチャーで、以下のコンパイラーツールセットで利用できます。

- Clang and LLVM Toolset
- Rust Toolset
- Go Toolset

詳細は、Red Hat Developer Tools のドキュメント を参照してください。

第17章 改訂履歴

バージョン	日付	Change	変更点
0.2-7	2023 年 12 月 20 日	古いリンクを削除しました。	Lenka Špačková
0.2-6	2023 年 7 月 3 日	**rhscl/mariadb-103-rhel7** コンテナーイメージは EOL です。	Lenka Špačková
0.2-5	2023 年 5 月 23 日	Red Hat Developer Toolset 12.1 のリリースで更新します。	Lenka Špačková
0.2-4	2022 年 11 月 22 日	Red Hat Developer Toolset 12.0 のリリースで更新します。	Lenka Špačková
0.2-3	2021 年 11 月 15 日	Red Hat Software Collections 3.8 コンテナーイメージの使用のリリース	Lenka Špačková
0.2-2	2021 年 10 月 11 日	Red Hat Software Collections 3.8 Beta コンテナーイメージの使用のリリース	Lenka Špačková
0.2-1	2021 年 6 月 3 日	Red Hat Software Collections 3.7 コンテナーイメージの使用のリリース	Lenka Špačková
0.2-0	2021 年 5 月 3 日	Red Hat Software Collections 3.7 Beta コンテナーイメージの使用のリリース	Lenka Špačková
0.1-9	2021 年 4 月 6 日	サポートされているアーキテクチャーが改善されました。	Lenka Špačková
0.1-8	2021 年 1 月 13 日	概要に関する章とアプリケーションイメージのビルドに関する拡張情報が改善されました。	Lenka Špačková
0.1-7	2020 年 12 月 1 日	Red Hat Software Collections 3.6 コンテナーイメージの使用のリリース	Lenka Špačková
0.1-6	2020 年 10 月 29 日	Red Hat Software Collections 3.6 Beta コンテナーイメージの使用のリリース	Lenka Špačková
0.1-5	2020 年 5 月 26 日	Red Hat Software Collections 3.5 コンテナーイメージの使用のリリース	Lenka Špačková
0.1-4	Apr 21 2020	Red Hat Software Collections 3.5 Beta コンテナーイメージの使用のリリース	Lenka Špačková
0.1-3	2019 年 12 月 10 日	Red Hat Software Collections 3.4 コンテナーイメージの使用のリリース	Lenka Špačková

バージョン	日付	Change	変更点
0.1-2	2019 年 11 月 7 日	Red Hat Software Collections 3.4 Beta コンテナーイメージの使用のリリース	Lenka Špačková
0.1-1	2019 年 6 月 11 日	Red Hat Software Collections 3.3 コンテナーイメージの使用のリリース	Lenka Špačková
0.1-0	Apr 16 2019	Red Hat Software Collections 3.3 Beta コンテナーイメージの使用のリリース	Lenka Špačková
0.0-9	2018 年 11 月 13 日	Red Hat Software Collections 3.2 コンテナーイメージの使用のリリース	Lenka Špačková
0.0-8	2018 年 10 月 23 日	Red Hat Software Collections 3.2 Beta コンテナーイメージの使用のリリース	Lenka Špačková
0.0-8	2018 年 8 月 29 日	devtoolset-6-perftools の SystemTap に関連する既知の問題を追加。	Lenka Špačková
0.0-7	2018 年 5 月 10 日	MongoDB イメージドキュメントを拡張	Lenka Špačková
0.0-6	2018 年 5 月 3 日	Red Hat Software Collections 3.1 コンテナーイメージの使用のリリース	Lenka Špačková
0.0-5	Apr 04 2018	Red Hat Software Collections 3.1 Beta コンテナーイメージの使用のリリース	Lenka Špačková
0.0-3	2017 年 11 月 29 日	既存のコンテナーイメージの拡張セクションを追加。	Lenka Špačková
0.0-2	2017 年 10 月 24 日	Red Hat Software Collections 3.0 コンテナーイメージの使用のリリース	Lenka Špačková
0.0-1	2017 年 10 月 3 日	Red Hat Software Collections 3.0 Beta コンテナーイメージの使用のリリース	Lenka Špačková