Red Hat Software Collections 2

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

Red Hat Software Collections 2.4 コンテナーイメージの基本的な使用手順

Last Updated: 2022-11-28

Red Hat Software Collections 2 Red Hat Software Collections コンテナーイメージの使用

Red Hat Software Collections 2.4 コンテナーイメージの基本的な使用手順

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書では、Red Hat Software Collections に同梱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

前書き

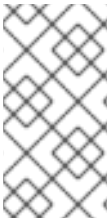
Red Hat Software Collections オファリングの一環として、Red Hat は、対応する Software Collections をベースとした多くのコンテナーイメージを提供しています。これには、アプリケーション、デーモン、およびデータベースイメージが含まれます。提供されるイメージについては、以下の表で詳しく説明します。

- Red Hat Software Collections 2.4 に基づくコンテナーイメージ
- Red Hat Software Collections 2.3 に基づくコンテナーイメージ
- Red Hat Software Collections 2.2 に基づくコンテナーイメージ
- Red Hat Software Collections 2.0 に基づくコンテナーイメージ

これらのイメージをコンテナー化された環境で使用して、アプリケーションをビルドし、デプロイし、実行できます。

コンテナーおよびコンテナーイメージの詳細については、コンテナー化されたアプリケーションの配信に関連するコア概念および方法について説明している OpenShift Enterprise 3.0 Architecture のコア概念 を参照してください。

Software Collections 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。



注記

Red Hat Software Collections コンテナーイメージの実行は、Red Hat Enterprise Linux 7 Server および Red Hat Enterprise Linux Atomic Host でのみサポートされます。Red Hat Enterprise Linux 7 Workstation または Red Hat Enterprise Linux 6 以前ではイメージを実行することはできません。

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、Using Volumes With the docker Container Can Cause Problems With SELinux を参照してください。

ベースイメージの使用または Source-to-Image の使用という、2 つの基本的な方法では、Red Hat Software Collections に同梱されているコンテナーイメージを使用できます。

第1章 ベースイメージの使用

Red Hat が提供するコンテナーイメージを独自の Dockerfile のベースイメージとして使用するには、以下の行を追加します。

```
FROM registry.access.redhat.com/rhscl/python-35-rhel7
```

Dockerfile の使用は、Red Hat Enterprise Linux Atomic Host 7 Getting Started with Containers で説明しています。Dockerfile の詳細は、Dockerfile reference を参照してください。

第2章 SOURCE-TO-IMAGE (S2I) の使用

Source-to-Image (S2I) はフレームワークで、アプリケーションのソースコードを入力として使用するイメージの作成を可能にするツールで、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できます。再現可能なコンテナーイメージをビルドするために S2I ツールを使用する主な利点は、開発者が使いやすくすることです。

システムで S2I ツールを使用するには、Red Hat Software Collections にサブスクライブし、以下のコマンドを実行して **source-to-image** パッケージをインストールします。

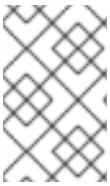
```
# yum install source-to-image
```

RHSM チャネル **rhel-server-rhscl-7-rpms** を使用します。**source-to-image** パッケージには、Red Hat Enterprise Linux Extras チャネルの **docker** パッケージが必要なことに注意してください。

または、**rhel-x86_64-server-7-rhscl-1** RHN チャネルを使用できますが、RHN チャネルは Red Hat Satellite インスタンスを通してのみアクセス可能であることに注意してください。

Red Hat Software Collections へのサブスクライブの詳細は、Red Hat Software Collections へのアクセス を参照してください。

S2I ツールの詳細は GitHub で参照できます。



注記

Red Hat Software Collections コンテナーイメージと同様に、S2I ツールは Red Hat Enterprise Linux 7 Workstation ではなく、Red Hat Enterprise Linux 7 Server でのみ実行されます。

2.1. ビルドプロセス

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。
- ビルダーイメージ: S2I ツールを使用してイメージのビルドをサポートする Red Hat が提供する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。

ビルドプロセス時に、S2I 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。

Source-to-Image フレームワークの詳細は、S2I 要件 を参照してください。

2.2. 例: S2I を使用して GIT から PYTHON アプリケーションのビルド

以下の例は、ビルド方法を示しています。

- Red Hat コンテナーレジストリーで利用可能な **python-35-rhel7** ビルダーイメージからの新規コンテナーイメージ

- **3.5/test/setup-test-app/** ディレクトリーの GitHub sti-python リポジトリーのパブリック Git リポジトリーから利用できるテストアプリケーション。

1. Red Hat Software Collections リポジトリーから S2I ツールをインストールします。

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

2. ビルダーイメージをプルします。

```
# docker pull registry.access.redhat.com/rhscl/python-35-rhel7
```

3. **3.5/test/setup-test-app/** ディレクトリーの GitHub sti-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/openshift/sti-python.git --context-dir=3.5/test/setup-test-app/
rhscl/python-35-rhel7 python-35-rhel7-app
```

これにより、新しいアプリケーションイメージ **python-35-rhel7-app** が作成されます。

4. 作成された **python-35-rhel7-app** イメージを実行します。

```
# docker run -d -p 8080:8080 --name example-app python-35-rhel7-app
```

5. **http://localhost:8080/** からドキュメントを取得します。

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

ドキュメントの例が返されます。

6. コンテナーを停止します。

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

第3章 RED HAT SOFTWARE COLLECTIONS 2.4 に基づくコンテナーイメージ

コンポーネント	説明
アプリケーションイメージ	
rhscl/nodejs-6-rhel7	アプリケーションを構築して実行する Node.js 6 プラットフォーム (EOL)
rhscl/python-27-rhel7	アプリケーションを構築して実行する Python 2.7 プラットフォーム (EOL)
rhscl/ruby-24-rhel7	アプリケーションを構築して実行する Ruby 2.4 プラットフォーム (EOL)
rhscl/ror-50-rhel7	アプリケーションを構築して実行する Ruby on Rails 5.0 プラットフォーム (EOL)
rhscl/thermostat-16-agent-rhel7	他のコンテナー内の Java アプリケーションの監視に適した Thermostat 1.6 エージェント (EOL)
rhscl/thermostat-16-storage-rhel7	データを保存および取得するための Web エンドポイント Thermostat 1.6 ストレージ (EOL)
デーモンイメージ	
rhscl/httpd-24-rhel7	Apache HTTP 2.4 サーバー
rhscl/nginx-110-rhel7	nginx 1.10 サーバーおよびリバースプロキシーサーバー (EOL)
Red Hat Developer Toolset 6.1 イメージ	
rhscl/devtoolset-6-toolchain-rhel7	Red Hat Developer Toolset ツールチェーン (EOL)
rhscl/devtoolset-6-perftools-rhel7	Red Hat Developer Toolset perftools (EOL)

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 2.4 が提供するコンポーネントの詳細は、Red Hat Software Collections 2.4 リリースノート を参照してください。

Red Hat Developer Toolset 6.1 のコンポーネントに関する詳細は、Red Hat Developer Toolset 6.1 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 2.3 に基づくコンテナーイメージ

コンポーネント	説明
アプリケーションイメージ	
rhscl/perl-524-rhel7	アプリケーションを構築して実行する Perl 5.24 プラットフォーム (EOL)
rhscl/php-56-rhel7	アプリケーションを構築して実行する PHP 5.6 プラットフォーム (EOL)
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
rhscl/python-35-rhel7	アプリケーションを構築して実行する Python 3.5 プラットフォーム (EOL)
rhscl/ruby-23-rhel7	アプリケーションを構築して実行する Ruby 2.3 プラットフォーム (EOL)
データベースイメージ	
rhscl/mysql-57-rhel7	MySQL 5.7 SQL データベースサーバー (EOL)
rhscl/mongodb-32-rhel7	MongoDB 3.2 NoSQL データベースサーバー (EOL)
rhscl/redis-32-rhel7	Redis 3.2 キー値ストア (EOL)

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 2.3 が提供するコンポーネントの詳細は、Red Hat Software Collections 2.3 リリースノート 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 2.2 に基づくコンテナーイメージ

コンポーネント	説明
アプリケーションイメージ	
rhscl/nodejs-4-rhel7	アプリケーションを構築して実行する Node.js 4 プラットフォーム (EOL)
rhscl/ror-42-rhel7	アプリケーションを構築して実行する Ruby on Rails 4.2 プラットフォーム (EOL)
rhscl/thermostat-1-agent-rhel7	他のコンテナー内の Java アプリケーションの監視に適した Thermostat 1.4 エージェント (EOL)
デーモンイメージ	
rhscl/nginx-18-rhel7	nginx 1.8 サーバーおよびリバースプロキシーサーバー (EOL)
rhscl/varnish-4-rhel7	Varnish Cache 4.0 HTTP リバースプロキシー (EOL)
データベースイメージ	
rhscl/mariadb-101-rhel7	MariaDB 10.1 SQL データベースサーバー (EOL)
rhscl/postgresql-95-rhel7	PostgreSQL 9.5 SQL データベースサーバー (EOL)
Red Hat Developer Toolset 4.1 イメージ	
rhscl/devtoolset-4-toolchain-rhel7	Red Hat Developer Toolset ツールチェーン (EOL)
rhscl/devtoolset-4-perftools-rhel7	Red Hat Developer Toolset perftools (EOL)

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 2.2 が提供するコンポーネントの詳細は、Red Hat Software Collections 2.2 リリースノート を参照してください。

Red Hat Developer Toolset 4.1 のコンポーネントに関する詳細は、Red Hat Developer Toolset 4.1 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6章 RED HAT SOFTWARE COLLECTIONS 2.0 に基づくコンテナーイメージ

コンポーネント	説明
アプリケーションイメージ	
rhscl/python-34-rhel7	アプリケーションを構築して実行する Python 3.4 プラットフォーム (EOL)
rhscl/perl-520-rhel7	アプリケーションを構築して実行する Perl 5.20 プラットフォーム (EOL)
rhscl/ruby-22-rhel7	アプリケーションを構築して実行する Ruby 2.2 プラットフォーム (EOL)
rhscl/ror-41-rhel7	アプリケーションを構築して実行する Ruby on Rails 4.1 プラットフォーム (EOL)
rhscl/passenger-40-rhel7	Phusion Passenger 4.0 Web サーバーおよびアプリケーションサーバー (EOL)
デーモンイメージ	
rhscl/nginx-16-rhel7	nginx 1.6 サーバーおよびリバースプロキシーサーバー (EOL)
データベースイメージ	
rhscl/mysql-56-rhel7	MySQL 5.6 SQL データベースサーバー (EOL)
rhscl/mariadb-100-rhel7	MariaDB 10.0 SQL データベースサーバー (EOL)
rhscl/postgresql-94-rhel7	PostgreSQL 9.4 SQL データベースサーバー (EOL)
rhscl/mongodb-26-rhel7	MongoDB 2.6 NoSQL データベースサーバー (EOL)

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 2.0 が提供するコンポーネントの詳細は、Red Hat Software Collections 2.0 リリースノート 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 アプリケーションイメージ

7.1. NODE.JS

7.1.1. 説明

rhscl/nodejs-6-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 6 プラットフォームを提供します。**rhscl/nodejs-4-rhel7** イメージは Node.js 4 プラットフォームを提供します。

7.1.2. アクセス

rhscl/nodejs-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。

```
# docker pull registry.access.redhat.com/rhscl/nodejs-6-rhel7
```

rhscl/nodejs-4-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。

```
# docker pull registry.access.redhat.com/rhscl/nodejs-4-rhel7
```

7.1.3. 設定

その他の設定は必要ありません。

7.2. PERL

7.2.1. 説明

rhscl/perl-524-rhel7 イメージは、アプリケーションを構築して実行する Perl 5.24 プラットフォームを提供します。**rhscl/perl-520-rhel7** イメージは Perl 5.20 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** を備えた **Apache httpd 2.4** が事前にインストールされています。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もサポートします。

7.2.2. アクセス

rhscl/perl-524-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。

```
# docker pull registry.access.redhat.com/rhscl/perl-524-rhel7
```

rhscl/perl-520-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。

```
# docker pull registry.access.redhat.com/rhscl/perl-520-rhel7
```

7.2.3. 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**
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。
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 自動設定を無効にします	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。

7.3. PHP

7.3.1. 説明

rhscl/php-70-rhel7 イメージは、アプリケーションを構築して実行するための PHP 7.0 プラットフォームを提供し、**rhscl/php-56-rhel7** イメージは PHP 5.6 プラットフォームを提供します。

7.3.2. アクセス

rhscl/php-70-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。

```
# docker pull registry.access.redhat.com/rhscl/php-70-rhel7
```

rhscl/php-56-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。

```
# docker pull registry.access.redhat.com/rhscl/php-56-rhel7
```

7.3.3. 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/php56/root/usr/share/pear**
SESSION_PATH	セッションデータファイルの場所	**/tmp/sessions**

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 共有メモリーのストレージサイズ	16M

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

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。

7.4. PYTHON

7.4.1. 説明

rhscl/python-35-rhel7 イメージは、アプリケーションを構築して実行する Python 3.5 プラットフォームを提供します。**rhscl/python-34-rhel7** イメージには Python 3.4 プラットフォームが含まれており、**rhscl/python-27-rhel7** イメージには Python 2.7 プラットフォームが含まれています。

7.4.2. アクセス

rhscl/python-35-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.access.redhat.com/rhscl/python-35-rhel7
```

rhscl/python-34-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.access.redhat.com/rhscl/python-34-rhel7
```

rhscl/python-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.access.redhat.com/rhscl/python-27-rhel7
```

7.4.3. 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	説明
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。

変数名	説明
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_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_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_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。

7.5. RUBY

7.5.1. 説明

rhscl/ruby-24-rhel7 イメージは、アプリケーションを構築および実行するための Ruby 2.4 プラットフォームを提供します。アセットのコンパイル用に Node.js 6 が事前にインストールされています。

rhscl/ruby-23-rhel7 イメージは、アプリケーションを構築および実行するための Ruby 2.3 プラットフォームを提供します。アセットのコンパイル用に Node.js 4 が事前にインストールされています。

rhscl/ruby-22-rhel7 イメージは Ruby 2.2 プラットフォームを提供します。アセットのコンパイル用に Node.js 0.10 が事前にインストールされています。

7.5.2. アクセス

rhscl/ruby-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。

```
# docker pull registry.access.redhat.com/rhscl/ruby-24-rhel7
```

rhscl/ruby-23-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。

```
# docker pull registry.access.redhat.com/rhscl/ruby-23-rhel7
```

rhscl/ruby-22-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。

```
# docker pull registry.access.redhat.com/rhscl/ruby-22-rhel7
```

7.5.3. 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	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に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。

7.6. RUBY ON RAILS

7.6.1. 説明

rhscl/ror-50-rhel7 は、アプリケーションを構築および実行するための Ruby on Rails 5.0 プラットフォームを提供します。これには、Ruby 2.4、Ruby on Rails 5.0、および Node.js 6 が事前にインストールされています。

rhscl/ror-42-rhel7 は、アプリケーションを構築および実行するための Ruby on Rails 4.2 プラットフォームを提供します。これには、Ruby 2.3、Ruby on Rails 4.2、および Node.js 4 が事前にインストールされています。

rhscl/ror-41-rhel7 は Ruby on Rails 4.1 プラットフォームを提供します。これには、Ruby 2.2、Ruby on Rails 4.1、および Node.js 0.10 が事前にインストールされています。

7.6.2. アクセス

rhscl/ror-50-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。

```
# docker pull registry.access.redhat.com/rhscl/ror-50-rhel7
```

rhscl/ror-42-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。

```
# docker pull registry.access.redhat.com/rhscl/ror-42-rhel7
```

rhscl/ror-41-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。

```
# docker pull registry.access.redhat.com/rhscl/ror-41-rhel7
```

7.6.3. 設定

その他の設定は必要ありません。

rhscl/ror-50-rhel7 イメージには、**rh-ruby24**、**rh-ror50**、および **rh-nodejs6** Software Collections が含まれ、有効化されます。**rhscl/ror-42-rhel7** イメージには、**rh-ruby23**、**rh-ror42**、および **rh-nodejs4** Software Collections が含まれ、有効化されます。 **rhscl/ror-41-rhel7** イメージには、**rh-ruby22**、**rh-ror41**、および **nodejs010** Software Collections が含まれ、有効化されます。

自動 S2I ビルドの場合は、Ruby コンテナーを使用します。

7.7. PHUSION PASSENGER

7.7.1. 説明

rhscl/passenger-40-rhel7 イメージは、Apache httpd Web サーバーで設定された Phusion Passenger 4.0 アプリケーションサーバーを提供します。また、アプリケーションを構築および実行するための Ruby 2.2 プラットフォームも提供します。Node.js 0.10 は、アセットのコンパイル用にが事前にインストールされています。

7.7.2. アクセス

rhscl/passenger-40-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。

```
# docker pull registry.access.redhat.com/rhscl/passenger-40-rhel7
```

7.7.3. 設定

これ以上の設定は必要ありません。このイメージには、**rh-ruby22**、**rh-ror41**、**nodejs010**、**rh-passenger40**、および **httpd24** Software Collections が含まれ、有効化されます。特に、自動 S2I ビルドをサポートするように設計されています。

7.8. THERMOSTAT エージェント

7.8.1. 説明

rhscl/thermostat-16-agent-rhel7 イメージは、コンテナー内の Java アプリケーションの監視に適した **thermostat エージェント** を提供します。

rhscl/thermostat-16-agent-rhel7 および **rhscl/thermostat-1-agent-rhel7** イメージはサポートされなくなりました。

7.8.2. アクセス

rhscl/thermostat-16-agent-rhel7 イメージをプルするには、**root** として次のコマンドを実行します。

```
# docker pull registry.access.redhat.com/rhscl/thermostat-16-agent-rhel7
```

7.8.3. 使用方法

このイメージは、以下を使用して、Dockerfile 内のビルダーおよびランタイムイメージのベースイメージとして使用することを目的としています。

```
FROM rhscl/thermostat-16-agent-rhel7
```

この Dockerfile の 例 を参照してください。この Dockerfile には、Thermostat エージェントが事前にインストールされています。

rhscl/thermostat-16-agent-rhel7 イメージがイメージ階層に導入されると、必要な 3 つの Thermostat 環境変数を設定することで、Thermostat エージェントを開始できます。たとえば、**rhscl/thermostat-test** というイメージは、ベースイメージとして **rhscl/thermostat-16-agent-rhel7** を使用し、デプロイメント時に Java アプリケーション **foo** を実行します。次の環境変数を設定することで、Thermostat エージェントを **foo** と一緒に開始できます。

- THERMOSTAT_AGENT_USERNAME
- THERMOSTAT_AGENT_PASSWORD
- THERMOSTAT_DB_URL

rhscl/thermostat-16-storage-rhel7 イメージを使用して、エージェントが接続するストレージエンドポイントを設定できます。

7.8.4. 設定

イメージは、**-e VAR=VALUE** を **docker 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
THERMOSTAT_AGENT_USERNAME	ストレージへの接続に使用する Thermostat エージェントのユーザー名
THERMOSTAT_AGENT_PASSWORD	ストレージに接続するためのパスワード

変数名	説明
THERMOSTAT_DB_URL	Thermostat ストレージの URL
APP_USER	Java アプリケーションの Thermostat が実行を監視するアプリケーションユーザーは、次のように実行されます。

7.9. THERMOSTAT ストレージ

7.9.1. 説明

rhscl/thermostat-16-storage-rhel7 には、データを保存および取得するための Web エンドポイントを提供する Thermostat 1.6 ストレージが含まれています。このイメージは、**rh-thermostat16** Software Collection に基づいています。

rhscl/thermostat-16-storage-rhel7 イメージはサポートされなくなりました。

7.9.2. アクセス

rhscl/thermostat-16-storage-rhel7 イメージをプルするには、**root** として次のコマンドを実行します。

```
# docker pull registry.access.redhat.com/rhscl/thermostat-16-storage-rhel7
```

7.9.3. 使用方法

他の MongoDB バックエンド (別のコンテナーによって提供されるものなど) に接続された Thermostat ストレージを実行するには、MongoDB URL、**mongo** ユーザー名とパスワード、およびエージェントとクライアントのユーザー名とパスワードを提供する必要があります。次のコマンドを実行します。

```
# docker run -d \
  -e MONGO_URL=mongodb://172.17.0.1:27017 \
  -e MONGO_USERNAME=<mongouser> \
  -e MONGO_PASSWORD=<mongopass> \
  -e THERMOSTAT_AGENT_USERNAMES=<agentuser1,agentuser2> \
  -e THERMOSTAT_AGENT_PASSWORDS=<agentpass1,agentpass2> \
  -e THERMOSTAT_CLIENT_USERNAMES=<clientuser1,clientuser2> \
  -e THERMOSTAT_CLIENT_PASSWORDS=<clientpass1,clientpass2> \
  --name thermostat16-storage \
  rhscl/thermostat-16-storage-rhel7
```

これにより、提供された **mongo** ユーザー名とパスワードを使用して、MongoDB URL に接続された HTTP レイヤーでコンテナーが実行されます。コンテナーには、環境変数で指定した適切なクライアントまたはエージェントの認証情報を使用して、**http://ip-address:8080/thermostat/storage** でアクセスできます。IP アドレスを見つけるには、次のコマンドを実行します。

```
# docker inspect --format '\{\{ .NetworkSettings.IPAddress }}' thermostat16-storage
```

上記の例のコマンドは、次のエージェントユーザーを作成します。

- **agentuser1** とパスワード **agentpass1**
- **agentuser2** とパスワード **agentpass2**

それらのいずれか、または両方を、Thermostat エージェントの接続に使用できます。

さらに、次のクライアントユーザーが使用可能になります。

- **clientuser1** とパスワード **clientpass1**
- **clientuser2** とパスワード **clientpass2**

それらのいずれか、または両方を、Thermostat クライアント接続に使用できます。

7.9.4. 設定

rhscl/thermostat-16-storage-rhel7 イメージは、**docker run** コマンドに **-e VAR=VALUE** を渡すことで、初期化中に設定できる次の環境変数を認識します。

変数名	説明
THERMOSTAT_AGENT_USERNAMES	Thermostat エージェントがストレージに接続するためのスペース区切りのユーザー名のリスト
THERMOSTAT_AGENT_PASSWORDS	ストレージに接続するエージェントのパスワードのスペース区切りリスト
THERMOSTAT_CLIENT_USERNAMES	Thermostat クライアントがストレージに接続するためのスペース区切りのユーザー名のリスト
THERMOSTAT_CLIENT_PASSWORDS	ストレージに接続するクライアントのパスワードのスペース区切りリスト
MONGO_USERNAME	MongoDB バッキングストレージのユーザー名
MONGO_PASSWORD	MongoDB バッキングストレージのパスワード
MONGO_URL	接続先の MongoDB URL

第8章 デーモンイメージ

8.1. APACHE HTTP サーバー

8.1.1. 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。

8.1.2. 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.access.redhat.com/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。

8.1.3. 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
```

8.2. NGINX

8.2.1. 説明

rhscl/nginx-110-rhel7 イメージは、nginx 1.10 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 1.10 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-18-rhel7 イメージは、nginx 1.8 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 1.8 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-16-rhel7 イメージはサポートされなくなりました。

8.2.2. アクセス

rhscl/nginx-110-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。

```
# docker pull registry.access.redhat.com/rhscl/nginx-110-rhel7
```

rhscl/nginx-18-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。

```
# docker pull registry.access.redhat.com/rhscl/nginx-18-rhel7
```

8.2.3. 設定

nginx コンテナーイメージは、**docker 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、**docker 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** が設定されている場合、nginx は **/var/log/nginx110** または **/var/log/nginx18** (使用するバージョンに応じて) にログインし、Docker 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-110-rhel7 イメージおよび **rhscl/nginx-18-rhel7** イメージは、S2I ツールの使用をサポートしています。

8.3. VARNISH CACHE

8.3.1. 説明

rhscl/varnish-4-rhel7 イメージは、HTTP リバースプロキシーである Varnish Cache 4.0 を提供します。

8.3.2. アクセス

rhscl/varnish-4-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。

```
# docker pull registry.access.redhat.com/rhscl/varnish-4-rhel7
```

8.3.3. 設定

その他の設定は必要ありません。

rhscl/varnish-4-rhel7 イメージは、S2I ツールの使用をサポートしています。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 形式でなければならない点に注意してください。

第9章 データベースイメージ

9.1. MYSQL

9.1.1. 説明

rhscl/mysql-56-rhel7 イメージは、MySQL 5.6 SQL データベースサーバーを提供します。**rhscl/mysql-57-rhel7** イメージは、MySQL 5.7 SQL データベースサーバーを提供します。

9.1.2. アクセスと使用方法

rhscl/mysql-56-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。

```
# docker pull registry.access.redhat.com/rhscl/mysql-56-rhel7
```

rhscl/mysql-57-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。

```
# docker pull registry.access.redhat.com/rhscl/mysql-57-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。

```
# docker run -d --name mysql_database -e MYSQL_USER=<user> -e
MYSQL_PASSWORD=<pass> \
  -e MYSQL_DATABASE=<db> -p 3306:3306 rhscl/mysql-56-rhel7
```

rhscl/mysql-57-rhel7 イメージを使用している場合は、イメージ名を変更します。

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は、**-v /host/db/path:/var/lib/mysql/data:Z** 引数も追加してください。**/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。データベースが初期化された後、またはすでに存在していた場合は、**mysqld** が実行され、**PID 1** として実行されます。**docker stop mysql_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。

9.1.3. 設定

イメージは、**-e VAR=VALUE** を **docker 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

変数名	説明
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。

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

変数名	説明	デフォルト
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 (または使用可能量の 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

-v /host:/container:Z フラグを Docker に渡すことで、次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 (デフォルトでは 27)。

9.2. MARIADB

9.2.1. 説明

rhscl/mariadb-101-rhel7 イメージは MariaDB 10.1 SQL データベースサーバーを提供します。**rhscl/mariadb-100-rhel7** イメージは MariaDB 10.0 SQL データベースサーバーを提供します。

9.2.2. アクセス

rhscl/mariadb-101-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。

```
# docker pull registry.access.redhat.com/rhscl/mariadb-101-rhel7
```

rhscl/mariadb-100-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。

```
# docker pull registry.access.redhat.com/rhscl/mariadb-100-rhel7
```

9.2.3. 使用法と設定

使用方法と設定は MySQL イメージと同じです。詳細については、MySQL セクションを参照してください。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。How to Extend the rhscl/mariadb-101-rhel7 Container Image も参照してください。

9.3. POSTGRESQL

9.3.1. 説明

rhscl/postgresql-95-rhel7 イメージは、PostgreSQL 9.5 SQL データベースサーバーを提供します。**rhscl/postgresql-94-rhel7** イメージは PostgreSQL 9.4 SQL データベースサーバーを提供します。

9.3.2. アクセスと使用方法

rhscl/postgresql-95-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。

```
# docker pull registry.access.redhat.com/rhscl/postgresql-95-rhel7
```

rhscl/postgresql-94-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。

```
# docker pull registry.access.redhat.com/rhscl/postgresql-94-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。

```
# docker run -d --name postgresql_database -e POSTGRESQL_USER=<user> \
  -e POSTGRESQL_PASSWORD=<pass> -e POSTGRESQL_DATABASE=<db> \
  -p 5432:5432 rshcl/postgresql-94-rhel7
```

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。**docker 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。

9.3.3. 設定

イメージは、**-e VAR=VALUE** を **docker 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。

以下の環境変数は PostgreSQL の設定ファイルに影響を与え、いずれもオプションとなります。

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_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

-v /host:/container フラグを Docker に渡すことで、次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。

9.4. MONGODB

9.4.1. 説明

rhscl/mongodb-32-rhel7 イメージは、MongoDB 3.2 NoSQL データベースサーバーを提供します。**rhscl/mongodb-26-rhel7** イメージは、MongoDB 2.6 NoSQL データベースサーバーを提供します。

9.4.2. アクセスと使用方法

rhscl/mongodb-32-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。

```
# docker pull registry.access.redhat.com/rhscl/mongodb-32-rhel7
```

rhscl/mongodb-26-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。

```
# docker pull registry.access.redhat.com/rhscl/mongodb-26-rhel7
```

必須の環境変数のみを設定し、そのデータベースをホストファイルシステムの **/home/user/database** ディレクトリーに保存するには、次のコマンドを実行します。

```
# docker run -d -e MONGODB_USER=<user> -e MONGODB_PASSWORD=<password> \
  -e MONGODB_DATABASE=<database> -e
MONGODB_ADMIN_PASSWORD=<admin_password> \
  -v /home/user/database:/var/lib/mongodb/data rhscl/mongodb-26-rhel7
```

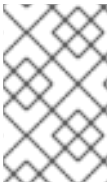
必要に応じてイメージ名を変更します。

データベースを初期化中で、指定した共有ボリュームを初めて使用している場合は、**admin** および **MONGODB_USER** という 2 つのユーザーでデータベースが作成されます。その後、MongoDB デーモンが起動します。ボリュームを別のコンテナーに再アタッチする場合は、データベースユーザーと **admin** ユーザーの作成は省略され、MongoDB デーモンのみが開始されます。

9.4.3. 設定

イメージは、**-e VAR=VALUE** を **docker 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
MONGODB_USER	作成される **MONGODB** アカウントのユーザー名
MONGODB_PASSWORD	ユーザーアカウントのパスワード
MONGODB_DATABASE	データベース名
MONGODB_ADMIN_PASSWORD	**admin** ユーザーのパスワード



注記

管理者ユーザー名は **admin** に設定され、**MONGODB_ADMIN_PASSWORD** 環境変数を設定してパスワードを指定する必要があります。このプロセスは、データベースの初期化時に行われます。

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

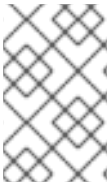
変数名	説明	デフォルト
MONGODB_NOPREALLOC	データファイルの事前割り当てを無効にします。	**true**
MONGODB_SMALLFILES	MongoDB がより小さなデフォルトのデータファイルサイズを使用するように設定します。	**true**
MONGODB_QUIET	出力量の制限を試みる quiet モードで MongoDB を実行します。	**true**

注記

rhscl/mongodb-32-rhel7 イメージでは、**MONGODB_NOPREALLOC** および **MONGODB_SMALLFILES** オプションは無効です。

-v /host:/container フラグを Docker に渡すことで、次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/mongodb/data	MongoDB 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。

9.4.4. カスタム設定ファイル

mongod サーバー用のカスタム設定ファイルを使用することができます。カスタム設定ファイルを提供すると、個々の設定の環境変数値が置き換えられます。

コンテナーで使用されるカスタム設定ファイルは、**/etc/mongod.conf** にマウントする必要があります。たとえば、**/home/user** ディレクトリーに保存されている設定ファイルを使用するには、**docker run** コマンドに **-v /home/user/mongod.conf:/etc/mongod.conf:Z** のオプションを使用します。

注記

カスタム設定ファイルは、レプリカセットの名前に影響を与えません。レプリカセット名は、**MONGODB_REPLICA_NAME** 環境変数に設定する必要があります。

9.5. REDIS

9.5.1. 説明

rhscl/redis-32-rhel7 イメージは、高度なキー/値ストアである Redis 3.2 を提供します。このイメージは、**rh-redis32** Software Collection をベースにしています。

9.5.2. アクセス

rhscl/redis-32-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。

```
# docker pull registry.access.redhat.com/rhscl/redis-32-rhel7
```

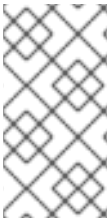
9.5.3. 設定

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

-v /host:/container フラグを Docker に渡すことで、次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/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。このコンテナーのデフォルトの UID は **1001** です。

第10章 RED HAT DEVELOPER TOOLSET イメージ

10.1. 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 docker 形式の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。

```
# docker run image_name usage
```

ビルド済みの docker 形式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。

```
# docker 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。

```
# docker run -ti rhscl/devtoolset-6-toolchain-rhel7 /bin/bash -l
```

例10.1 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 docker 形式の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. RHEL 7 での Docker の取得 の手順に従って、**Docker** 環境がシステムに適切にセットアップされていることを確認します。
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。

   ```
   # docker pull rhscl/devtoolset-6-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。

   ```
   # docker run -ti rhscl/devtoolset-6-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、**docker** コマンドに **-v** (または **--volume**) オプションを追加します。

   ```
   $ sudo docker run -v ~/Source:/src -ti rhscl/devtoolset-6-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 6.3.1 20170216 (Red Hat 6.3.1-3) (GCC)
```

10.2. DOCKERFILE から構築されるコンテナーイメージの使用

Dockerfile は、一部の Red Hat Developer Toolset コンポーネントで利用できます。Dockerfile は、docker 形式のコンテナーイメージの自動ビルド手順が含まれるテキストファイルです。

Red Hat Developer Toolset 6.1 for Red Hat Enterprise Linux 7 には、以下の Dockerfile が同梱されています。

- devtoolset-6-toolchain
- devtoolset-6-perftools

10.2.1. Dockerfile の取得

Red Hat Developer Toolset Dockerfiles は、パッケージ **devtoolset-6-dockerfiles** で提供されます。このパッケージには、個々のコンポーネントを含む docker 形式のコンテナーイメージを構築するための個別の Dockerfiles と、提供されるすべてのコンポーネントを含む docker 形式のコンテナーイメージを構築するための meta-Dockerfile が含まれています。Dockerfiles を使用できるようにするには、次のコマンドを実行してこのパッケージをインストールします。

```
# yum install devtoolset-6-dockerfiles
```

RHSM チャネル **rhel-server-rhscl-7-rpms** を使用します。これを有効にするには、Red Hat Developer Toolset へのアクセスの取得 の指示に従ってください。

10.2.2. コンテナーイメージのビルド

Dockerfile がインストールされているディレクトリーに移動し、**root** で以下のコマンドを実行します。

```
# docker build -t image_name
```

image_name を、新しいイメージの名前に置き換えます。

例10.2 Red Hat Developer Toolset コンポーネントを使用したコンテナーイメージの構築

perftools ツールをコンテナーにデプロイするための docker 形式のコンテナーイメージをビルドするには、次の手順に従います。

1. RHEL 7 での Docker の取得 の手順に従って、**Docker** 環境がシステムに適切にセットアップされていることを確認します。
2. Red Hat Developer Toolset Dockerfiles を含むパッケージをインストールします。

```
# yum install devtoolset-6-dockerfiles
```

3. 必要なコンポーネントの Dockerfile がある場所を特定します。

```
# rpm -ql devtoolset-6-dockerfiles | grep "perftools-docker/Dockerfile"
/opt/rh/devtoolset-6/root/usr/share/devtoolset-6-dockerfiles/rhel7/devtoolset-6-perftools-docker/Dockerfile
```

4. 必要な Dockerfile がインストールされているディレクトリーに移動します。

```
# cd /opt/rh/devtoolset-6/root/usr/share/devtoolset-6-dockerfiles/rhel7/devtoolset-6-perftools-docker/
```

5. コンテナーイメージをビルドします。

```
# docker build -t devtoolset-6-my-perftools .
```

devtoolset-6-my-perftools を生成されるコンテナーイメージに割り当てる名前に置き換えます。

10.2.3. Custom-Built コンテナーイメージからの Red Hat Developer Toolset ツールの実行

Red Hat Developer Toolset Dockerfile (「コンテナーイメージのビルド」 を参照) からビルドされた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。

```
docker run image_name container-usage
```

ビルドした docker 形式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。

```
docker 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、ビルド時に選択したコンテナーイメージの名前に置き換えます。

例10.3 Custom-Built Red Hat Developer Toolset イメージでの elfutils の使用

この例では、**elfutils** コンポーネントを使用してカスタムビルドされた docker 形式のコンテナーイメージを起動する方法と、そのイメージ内で **eu-size** ツールを実行する方法を示しています。

1. 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。

```
# docker run -ti devtoolset-6-my-perftools /bin/bash -l
```

2. 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、**docker** コマンドに **-v** (または **--volume**) オプションを追加します。

```
$ sudo docker run -v ~/Source:/src -ti devtoolset-6-my-perftools /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。

3. 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、**eu-size** ツールのバージョンを確認するには、以下のコマンドを実行します。

```
bash-4.2$ eu-size -V
size (elfutils) 0.168
[...]
```

10.3. RED HAT DEVELOPER TOOLSET TOOLCHAIN

10.3.1. 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-6-toolchain-rhel7 イメージは、以下のパッケージに対応するコンテンツを提供します。

コンポーネント	バージョン	パッケージ
gcc	6.3.1	devtoolset-6-gcc
g++		devtoolset-6-gcc-c++
gfortran		devtoolset-6-gcc-fortran
gdb	7.12.1	devtoolset-6-gdb

rhscl/devtoolset-4-toolchain-rhel7 イメージは、以下のパッケージに対応するコンテンツを提供します。

コンポーネント	バージョン	パッケージ
gcc	5.3.1	devtoolset-4-gcc
g++		devtoolset-4-gcc-c++
gfortran		devtoolset-4-gcc-fortran
gdb	7.11	devtoolset-4-gdb

10.3.2. アクセス

rhscl/devtoolset-6-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。

```
# docker pull registry.access.redhat.com/rhscl/devtoolset-6-toolchain-rhel7
```

rhscl/devtoolset-4-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。

```
# docker pull registry.access.redhat.com/rhscl/devtoolset-4-toolchain-rhel7
```

10.4. RED HAT DEVELOPER TOOLSET パフォーマンスツール

10.4.1. 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-6-perftools-rhel7 イメージは、以下のコンポーネントを提供します。

コンポーネント	バージョン	パッケージ
OProfile	1.1.0	devtoolset-6-oprofile
SystemTap	3.0	devtoolset-6-systemtap
Valgrind	3.12.0	devtoolset-6-valgrind
Dyninst	9.2.0	devtoolset-6-dyninst
elfutils	0.168	devtoolset-6-elfutils

rhscl/devtoolset-4-perftools-rhel7 イメージは、以下のコンポーネントを提供します。

コンポーネント	バージョン	パッケージ
OProfile	1.1.0	devtoolset-4-oprofile
SystemTap	2.9	devtoolset-4-systemtap
Valgrind	3.11.0	devtoolset-4-valgrind
Dyninst	9.1.0	devtoolset-4-dyninst
elfutils	0.166	devtoolset-4-elfutils

10.4.2. アクセス

rhscl/devtoolset-6-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。

```
# docker pull registry.access.redhat.com/rhscl/devtoolset-6-perftools-rhel7
```

rhscl/devtoolset-4-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。

```
# docker pull registry.access.redhat.com/rhscl/devtoolset-4-perftools-rhel7
```

10.4.3. 使用方法

コンテナーイメージからの SystemTap ツールの使用

コンテナーイメージから **SystemTap** ツールを使用する場合は、追加の設定が必要で、コンテナーを特別なコマンドラインオプションを指定して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。

   ```
   ~]$ docker run --ti --privileged --ipc=host --net=host --pid=host devtoolset-6-my-perftools
   /bin/bash -l
   ```

   ビルド済みの **perftools** イメージを使用するには、上記のコマンドでイメージ名を **devtoolset-6-perftools-rhel7** 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
     3.10.0-514.10.2.el7.x86_64
     ```

     kernel-debuginfo パッケージは、**Debug** チャネルでのみ利用できることに注意してください。TODO WHERE の説明に従って **rhel-7-server-debug-rpms** リポジトリーを有効にします。debuginfo パッケージへのアクセス方法は、https://access.redhat.com/site/solutions/9907 を参照してください。

     正しいバージョンで必要なパッケージ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** パッケージ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. **docker 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。

   ```
   ~]$ docker commit devtoolset-6-systemtap-$(uname -r)
   ```