



Red Hat Software Collections 2

Using Red Hat Software Collections Container Images

Basic Usage Instructions for Red Hat Software Collections 2.4 Container
images

Red Hat Software Collections 2 Using Red Hat Software Collections Container Images

Basic Usage Instructions for Red Hat Software Collections 2.4 Container images

Lenka Špačková
lspackova@redhat.com

Robert Krátký
rkratky@redhat.com

Vladimír Slávik
vslavik@redhat.com

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution-Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This document provides instructions for obtaining, configuring, and using container images that are shipped with Red Hat Software Collections.

Table of Contents

| | |
|------------------------------------------------------------------------------|-----------|
| PREFACE | 4 |
| CHAPTER 1. USING BASE IMAGES | 5 |
| CHAPTER 2. USING SOURCE-TO-IMAGE (S2I) | 6 |
| 2.1. BUILD PROCESS | 6 |
| 2.2. EXAMPLE: BUILDING A PYTHON APPLICATION FROM GIT USING S2I | 6 |
| CHAPTER 3. CONTAINER IMAGES BASED ON RED HAT SOFTWARE COLLECTIONS 2.4 | 8 |
| CHAPTER 4. CONTAINER IMAGES BASED ON RED HAT SOFTWARE COLLECTIONS 2.3 | 9 |
| CHAPTER 5. CONTAINER IMAGES BASED ON RED HAT SOFTWARE COLLECTIONS 2.2 | 10 |
| CHAPTER 6. CONTAINER IMAGES BASED ON RED HAT SOFTWARE COLLECTIONS 2.0 | 11 |
| CHAPTER 7. APPLICATION IMAGES | 12 |
| 7.1. NODE.JS | 12 |
| 7.1.1. Description | 12 |
| 7.1.2. Access | 12 |
| 7.1.3. Configuration | 12 |
| 7.2. PERL | 12 |
| 7.2.1. Description | 12 |
| 7.2.2. Access | 12 |
| 7.2.3. Configuration | 12 |
| 7.3. PHP | 13 |
| 7.3.1. Description | 13 |
| 7.3.2. Access | 13 |
| 7.3.3. Configuration | 13 |
| 7.4. PYTHON | 15 |
| 7.4.1. Description | 15 |
| 7.4.2. Access | 15 |
| 7.4.3. Configuration | 15 |
| 7.5. RUBY | 16 |
| 7.5.1. Description | 16 |
| 7.5.2. Access | 16 |
| 7.5.3. Configuration | 17 |
| 7.6. RUBY ON RAILS | 17 |
| 7.6.1. Description | 17 |
| 7.6.2. Access | 18 |
| 7.6.3. Configuration | 18 |
| 7.7. PHUSION PASSENGER | 18 |
| 7.7.1. Description | 18 |
| 7.7.2. Access | 18 |
| 7.7.3. Configuration | 18 |
| 7.8. THERMOSTAT AGENT | 19 |
| 7.8.1. Description | 19 |
| 7.8.2. Access | 19 |
| 7.8.3. Usage | 19 |
| 7.8.4. Configuration | 19 |
| 7.9. THERMOSTAT STORAGE | 20 |
| 7.9.1. Description | 20 |
| 7.9.2. Access | 20 |

| | |
|------------------------------------------------------------------------------------|-----------|
| 7.9.3. Usage | 20 |
| 7.9.4. Configuration | 21 |
| CHAPTER 8. DAEMON IMAGES | 22 |
| 8.1. APACHE HTTP SERVER | 22 |
| 8.1.1. Description | 22 |
| 8.1.2. Access | 22 |
| 8.1.3. Configuration | 22 |
| 8.2. NGINX | 22 |
| 8.2.1. Description | 22 |
| 8.2.2. Access | 23 |
| 8.2.3. Configuration | 23 |
| 8.3. VARNISH CACHE | 23 |
| 8.3.1. Description | 23 |
| 8.3.2. Access | 23 |
| 8.3.3. Configuration | 24 |
| CHAPTER 9. DATABASE IMAGES | 25 |
| 9.1. MYSQL | 25 |
| 9.1.1. Description | 25 |
| 9.1.2. Access and Usage | 25 |
| 9.1.3. Configuration | 25 |
| 9.2. MARIADB | 27 |
| 9.2.1. Description | 27 |
| 9.2.2. Access | 28 |
| 9.2.3. Usage and Configuration | 28 |
| 9.3. POSTGRESQL | 28 |
| 9.3.1. Description | 28 |
| 9.3.2. Access and Usage | 28 |
| 9.3.3. Configuration | 29 |
| 9.4. MONGODB | 30 |
| 9.4.1. Description | 30 |
| 9.4.2. Access and Usage | 30 |
| 9.4.3. Configuration | 30 |
| 9.4.4. Custom configuration file | 31 |
| 9.5. REDIS | 32 |
| 9.5.1. Description | 32 |
| 9.5.2. Access | 32 |
| 9.5.3. Configuration | 32 |
| CHAPTER 10. RED HAT DEVELOPER TOOLSET IMAGES | 33 |
| 10.1. RUNNING RED HAT DEVELOPER TOOLSET TOOLS FROM PRE-BUILT CONTAINER IMAGES | 33 |
| 10.2. USING CONTAINER IMAGES BUILT FROM DOCKERFILES | 34 |
| 10.2.1. Obtaining Dockerfiles | 34 |
| 10.2.2. Building Container Images | 34 |
| 10.2.3. Running Red Hat Developer Toolset Tools from Custom-Built Container Images | 35 |
| 10.3. RED HAT DEVELOPER TOOLSET TOOLCHAIN | 36 |
| 10.3.1. Description | 36 |
| 10.3.2. Access | 37 |
| 10.4. RED HAT DEVELOPER TOOLSET PERFORMANCE TOOLS | 37 |
| 10.4.1. Description | 37 |
| 10.4.2. Access | 38 |
| 10.4.3. Usage | 38 |

PREFACE

As a part of the Red Hat Software Collections offering, Red Hat provides a number of container images, which are based on the corresponding Software Collections. These include application, daemon, and database images. The provided images are detailed in tables

- [Container Images Based on Red Hat Software Collections 2.4](#)
- [Container Images Based on Red Hat Software Collections 2.3](#)
- [Container Images Based on Red Hat Software Collections 2.2](#)
- [Container Images Based on Red Hat Software Collections 2.0](#)

You can use these images in a containerized environment to build, deploy, and run your applications.

For more information on containers and container images, see the [Core Concepts of the OpenShift Enterprise 3.0 Architecture](#), which discusses core concepts and methods related to delivering containerized applications.

For more information on Software Collections, see the [Red Hat Software Collections](#) and [Red Hat Developer Toolset](#) documentation.



NOTE

Running Red Hat Software Collections container images is supported only on Red Hat Enterprise Linux 7 Server and Red Hat Enterprise Linux Atomic Host. You cannot run the images on Red Hat Enterprise Linux 7 Workstation or Red Hat Enterprise Linux 6 or earlier.

When using SELinux for controlling processes within a container, make sure that any content that is volume mounted into the container is readable, and potentially writable, depending on the use case. For more information, see [Using Volumes With the docker Container Can Cause Problems With SELinux](#).

There are two basic approaches that you can take to use the container images shipped with Red Hat Software Collections: using base images or using Source-to-Image.

CHAPTER 1. USING BASE IMAGES

To use container images provided by Red Hat as base images in your own Dockerfile, add the following line to it:

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

Working with Dockerfiles is covered in the [Red Hat Enterprise Linux Atomic Host 7 Getting Started with Containers](#) document. Detailed information on Dockerfiles can be found in the [Dockerfile reference document](#).

CHAPTER 2. USING SOURCE-TO-IMAGE (S2I)

Source-to-Image (S2I) is a framework and a tool that allows you to write images which use the application source code as an input and produce a new image that runs the assembled application as an output. The main advantage of using the S2I tool for building reproducible container images is the ease of use for developers.

To use the S2I tool on your system, subscribe to Red Hat Software Collections and run the following command to install the **source-to-image** package:

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

Use the RHSM channel: **rhel-server-rhsc1-7-rpms**. Note that the **source-to-image** package requires the **docker** package from the Red Hat Enterprise Linux Extras channel.

Alternatively, you can use the **rhel-x86_64-server-7-rhsc1-1** RHN channel, but note that the RHN channel is accessible only through Red Hat Satellite instances.

For details about subscribing to Red Hat Software Collections, see [Getting Access to Red Hat Software Collections](#).

More information about the S2I tool is available at [GitHub](#).



NOTE

Similarly to Red Hat Software Collections container images, the S2I tool runs only on Red Hat Enterprise Linux 7 Server, not on Red Hat Enterprise Linux 7 Workstation.

2.1. BUILD PROCESS

The build process consists of the following three fundamental elements, which are combined into a final container image:

- The source code of your application—written in a programming language or framework.
- Builder image—container image provided by Red Hat that supports building images using the S2I tool.
- S2I scripts that are part of the builder image.

During the build process, S2I creates a tar file that contains the source code and scripts, then streams that file into the builder image.

For more information on the Source-to-Image framework, see [S2I Requirements](#).

2.2. EXAMPLE: BUILDING A PYTHON APPLICATION FROM GIT USING S2I

This example shows how to build:

- A new container image from the **python-35-rhel7** builder image that is available in the Red Hat Container Registry, and

- A test application available from a public Git repository in the [GitHub sti-python](#) repository, in the **3.5/test/setup-test-app/** directory.

1. Install the S2I tool from the Red Hat Software Collections repository:

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

2. Pull the builder image:

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

3. Build the test application from the [GitHub sti-python](#) repository, in the **3.5/test/setup-test-app/** directory:

```
# s2i build https://github.com/openshift/sti-python.git --  
context-dir=3.5/test/setup-test-app/ rhsccl/python-35-rhel7  
python-35-rhel7-app
```

This produces a new application image, **python-35-rhel7-app**.

4. Run the resulting **python-35-rhel7-app** image:

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

5. Fetch a document from <http://localhost:8080/>:

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

The example document is returned.

6. Stop the container:

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

CHAPTER 3. CONTAINER IMAGES BASED ON RED HAT SOFTWARE COLLECTIONS 2.4

| Component | Description |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------|
| Application Images | |
| rhsc1/nodejs-6-rhel7 | Node.js 6 platform for building and running applications |
| rhsc1/python-27-rhel7 | Python 2.7 platform for building and running applications |
| rhsc1/ruby-24-rhel7 | Ruby 2.4 platform for building and running applications |
| rhsc1/ror-50-rhel7 | Ruby on Rails 5.0 platform for building and running applications |
| rhsc1/thermostat-16-agent-rhel7 | Thermostat 1.6 agent suitable for monitoring Java applications in other containers (EOL) |
| rhsc1/thermostat-16-storage-rhel7 | Thermostat 1.6 storage , a web endpoint for storing and retrieving data (EOL) |
| Daemon Images | |
| rhsc1/httpd-24-rhel7 | Apache HTTP 2.4 Server |
| rhsc1/nginx-110-rhel7 | nginx 1.10 server and a reverse proxy server |
| Red Hat Developer Toolset 6.1 Images | |
| rhsc1/devtoolset-6-toolchain-rhel7 | Red Hat Developer Toolset toolchain |
| rhsc1/devtoolset-6-perftools-rhel7 | Red Hat Developer Toolset perftools |

All images are based on components from Red Hat Software Collections. The images are available for Red Hat Enterprise Linux 7 through the Red Hat Container Registry.

For detailed information about components provided by Red Hat Software Collections 2.4, see the [Red Hat Software Collections 2.4 Release Notes](#)

For more information about the Red Hat Developer Toolset 6.1 components, see the [Red Hat Developer Toolset 6.1 User Guide](#).

EOL images are no longer supported, and no further details are provided in this document.

CHAPTER 4. CONTAINER IMAGES BASED ON RED HAT SOFTWARE COLLECTIONS 2.3

| Component | Description |
|-------------------------------|------------------------------------------------------------------------------|
| Application Images | |
| rhsc1/perl-524-rhel7 | Perl 5.24 platform for building and running applications |
| rhsc1/php-56-rhel7 | PHP 5.6 platform for building and running applications (EOL) |
| rhsc1/php-70-rhel7 | PHP 7.0 platform for building and running applications |
| rhsc1/python-35-rhel7 | Python 3.5 platform for building and running applications |
| rhsc1/ruby-23-rhel7 | Ruby 2.3 platform for building and running applications |
| Database Images | |
| rhsc1/mysql-57-rhel7 | MySQL 5.7 SQL database server |
| rhsc1/mongodb-32-rhel7 | MongoDB 3.2 NoSQL database server |
| rhsc1/redis-32-rhel7 | Redis 3.2 key-value store |

All images are based on components from Red Hat Software Collections. The images are available for Red Hat Enterprise Linux 7 through the Red Hat Container Registry.

For detailed information about components provided by Red Hat Software Collections 2.3, see the [Red Hat Software Collections 2.3 Release Notes](#)

CHAPTER 5. CONTAINER IMAGES BASED ON RED HAT SOFTWARE COLLECTIONS 2.2

| Component | Description |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------|
| Application Images | |
| rhsc1/nodejs-4-rhel7 | Node.js 4 platform for building and running applications (EOL) |
| rhsc1/ror-42-rhel7 | Ruby on Rails 4.2 platform for building and running applications |
| rhsc1/thermostat-1-agent-rhel7 | Thermostat 1.4 agent suitable for monitoring Java applications in other containers (EOL) |
| Daemon Images | |
| rhsc1/nginx-18-rhel7 | nginx 1.8 server and a reverse proxy server |
| rhsc1/varnish-4-rhel7 | Varnish Cache 4.0 HTTP reverse proxy |
| Database Images | |
| rhsc1/mariadb-101-rhel7 | MariaDB 10.1 SQL database server |
| rhsc1/postgresql-95-rhel7 | PostgreSQL 9.5 SQL database server |
| Red Hat Developer Toolset 4.1 Images | |
| rhsc1/devtoolset-4-toolchain-rhel7 | Red Hat Developer Toolset toolchain (EOL) |
| rhsc1/devtoolset-4-perftools-rhel7 | Red Hat Developer Toolset perftools (EOL) |

All images are based on components from Red Hat Software Collections. The images are available for Red Hat Enterprise Linux 7 through the Red Hat Container Registry.

For detailed information about components provided by Red Hat Software Collections 2.2, see the [Red Hat Software Collections 2.2 Release Notes](#)

For more information about the Red Hat Developer Toolset 4.1 components, see the [Red Hat Developer Toolset 4.1 User Guide](#).

EOL images are no longer supported, and no further details are provided in this document.

CHAPTER 6. CONTAINER IMAGES BASED ON RED HAT SOFTWARE COLLECTIONS 2.0

| Component | Description |
|----------------------------------|----------------------------------------------------------------------------------------|
| Application Images | |
| rhsc1/python-34-rhel7 | Python 3.4 platform for building and running applications (EOL) |
| rhsc1/perl-520-rhel7 | Perl 5.20 platform for building and running applications (EOL) |
| rhsc1/ruby-22-rhel7 | Ruby 2.2 platform for building and running applications (EOL) |
| rhsc1/ror-41-rhel7 | Ruby on Rails 4.1 platform for building and running applications (EOL) |
| rhsc1/passenger-40-rhel7 | Phusion Passenger 4.0 web server and application server (EOL) |
| Daemon Images | |
| rhsc1/nginx-16-rhel7 | nginx 1.6 server and a reverse proxy server (EOL) |
| Database Images | |
| rhsc1/mysql-56-rhel7 | MySQL 5.6 SQL database server (EOL) |
| rhsc1/mariadb-100-rhel7 | MariaDB 10.0 SQL database server (EOL) |
| rhsc1/postgresql-94-rhel7 | PostgreSQL 9.4 SQL database server (EOL) |
| rhsc1/mongodb-26-rhel7 | MongoDB 2.6 NoSQL database server (EOL) |

All images are based on components from Red Hat Software Collections. The images are available for Red Hat Enterprise Linux 7 through the Red Hat Container Registry.

For detailed information about components provided by Red Hat Software Collections 2.0, see the [Red Hat Software Collections 2.0 Release Notes](#)

EOL images are no longer supported, and no further details are provided in this document.

CHAPTER 7. APPLICATION IMAGES

7.1. NODE.JS

7.1.1. Description

The **rhsc1/nodejs-6-rhel7** image provides a Node.js 6 platform for building and running applications. The **rhsc1/nodejs-4-rhel7** image provides a Node.js 4 platform.

7.1.2. Access

To pull the **rhsc1/nodejs-6-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/nodejs-6-rhel7
```

To pull the **rhsc1/nodejs-4-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/nodejs-4-rhel7
```

7.1.3. Configuration

No further configuration is required.

7.2. PERL

7.2.1. Description

The **rhsc1/perl-524-rhel7** image provides a Perl 5.24 platform for building and running applications, the **rhsc1/perl-520-rhel7** image provides a Perl 5.20 platform. **Apache httpd 2.4** with **mod_perl** for deploying Perl web applications is preinstalled. The images also supports deploying Perl Web Server Gateway Interface (PSGI) applications.

7.2.2. Access

To pull the **rhsc1/perl-524-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/perl-524-rhel7
```

To pull the **rhsc1/perl-520-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/perl-520-rhel7
```

7.2.3. Configuration

To set environment variables, you can place them as a key-value pair into a **.sti/environment** file inside your source code repository.

| Variable Name | Description | Default |
|-------------------------|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| ENABLE_CPAN_TEST | Allows the installation of all specified cpan packages and the running of their tests | false |
| CPAN_MIRROR | Specifies a mirror URL which will be used by cpanminus to install dependencies | URL is not specified by default |
| PSGI_FILE | Specifies relative path to the PSGI application file; Use empty value to disable the PSGI autoconfiguration | Single *.psgi file in the top-level directory, if it exists |
| PSGI_URI_PATH | Specifies URI path that is handled by the PSGI application | / |

To install additional Perl modules from the Comprehensive Perl Archive Network (CPAN), create a **cpanfile** in the root directory of your application sources. The file must conform to the **cpanfile** format as defined in Module-CPANFile CPAN distribution. For detailed information about the cpanfile format, refer to the [cpanfile documentation](#).

To modify the **Apache httpd** behavior, drop the **.htaccess** file in the application sources tree where appropriate. For details about **.htaccess**, see the [Apache HTTP Server Tutorial](#).

7.3. PHP

7.3.1. Description

The **rhsc1/php-70-rhel7** image provides a PHP 7.0 platform for building and running applications, the **rhsc1/php-56-rhel7** image provides a PHP 5.6 platform.

7.3.2. Access

To pull the **rhsc1/php-70-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/php-70-rhel7
```

To pull the **rhsc1/php-56-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/php-56-rhel7
```

7.3.3. Configuration

To set environment variables, place them as a key-value pair into a **.sti/environment** file inside your source code repository.

The following environment variables set their equivalent property value in the **php.ini** file:

| Variable Name | Description | Default |
|-------------------------------|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| ERROR_REPORTING | Informs PHP of which errors, warnings and notices you would like it to take action for | E_ALL & ~E_NOTICE |
| DISPLAY_ERRORS | Controls whether or not and where PHP will output errors, notices and warnings | ON |
| DISPLAY_STARTUP_ERRORS | Cause display errors which occur during PHP's startup sequence to be handled separately from display errors | OFF |
| TRACK_ERRORS | Store the last error/warning message in \$php_errormsg (boolean) | OFF |
| HTML_ERRORS | Link errors to documentation related to the error | ON |
| INCLUDE_PATH | Path for PHP source files | ./opt/app-root/src:/opt/rh/php56/root/usr/share/pear |
| SESSION_PATH | Location for session data files | /tmp/sessions |

The following environment variable sets its equivalent property value in the **opcache.ini** file:

| Variable Name | Description | Default |
|-----------------------------------|----------------------------------------|---------|
| OPCACHE_MEMORY_CONSUMPTION | The OPcache shared memory storage size | 16M |

You can also override the entire directory used to load the PHP configuration by setting:

| Variable Name | Description |
|-------------------------|-----------------------------------------------------|
| PHPRC | Sets the path to the php.ini file |
| PHP_INI_SCAN_DIR | Path to scan for additional ini configuration files |

In case the DocumentRoot of the application is nested within the source directory

`/opt/app-root/src`, users can provide their own **.htaccess** file. This allows the overriding of Apache's behavior and specifies how application requests should be handled. The **.htaccess** file needs to be located at the root of the application source. For details about **.htaccess**, see the [Apache HTTP Server Tutorial](#).

7.4. PYTHON

7.4.1. Description

The **rhsc1/python-35-rhel7** image provides a Python 3.5 platform for building and running applications. The **rhsc1/python-34-rhel7** image includes a Python 3.4 platform, and the **rhsc1/python-27-rhel7** image provides a Python 2.7 platform.

7.4.2. Access

To pull the **rhsc1/python-35-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/python-35-rhel7
```

To pull the **rhsc1/python-34-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/python-34-rhel7
```

To pull the **rhsc1/python-27-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/python-27-rhel7
```

7.4.3. Configuration

To set environment variables, you can place them as a key-value pair into a **.sti/environment** file inside your source code repository.

| Variable Name | Description |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| APP_FILE | Used to run the application from a Python script. This should be a path to a Python file (defaults to app.py) that will be passed to the Python interpreter to start the application. |

| Variable Name | Description |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| APP_MODULE | Used to run the application with Gunicorn, as documented here. This variable specifies a WSGI callable with the pattern MODULE_NAME:VARIABLE_NAME , where MODULE_NAME is the full dotted path of a module, and VARIABLE_NAME refers to a WSGI callable inside the specified module. Gunicorn will look for a WSGI callable named application if not specified. If APP_MODULE is not provided, the run script will look for a wsgi.py file in your project and use it if it exists. If using setup.py for installing the application, the MODULE_NAME part can be read from there. For an example, see setup-test-app . |
| APP_CONFIG | Path to a valid Python file with a Gunicorn configuration file. |
| DISABLE_COLLECTSTATIC | Set this variable to a non-empty value to inhibit the execution of manage.py collectstatic during the build. This affects only Django projects. |
| DISABLE_MIGRATE | Set this variable to a non-empty value to inhibit the execution of manage.py migrate when the produced image is run. This affects only Django projects. |

7.5. RUBY

7.5.1. Description

The **rhsc/ruby-24-rhel7** image provides a Ruby 2.4 platform for building and running applications; Node.js 6 is preinstalled for assets compilation.

The **rhsc/ruby-23-rhel7** image provides a Ruby 2.3 platform for building and running applications; Node.js 4 is preinstalled for assets compilation.

The **rhsc/ruby-22-rhel7** image provides a Ruby 2.2 platform; Node.js 0.10 is preinstalled for assets compilation.

7.5.2. Access

To pull the **rhsc/ruby-24-rhel7** image, run the following command as **root**:

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

To pull the **rhsc/ruby-23-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/ruby-23-rhel7
```

To pull the **rhsc1/ruby-22-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/ruby-22-rhel7
```

7.5.3. Configuration

To set environment variables, you can place them as a key-value pair into a **.sti/environment** file inside your source code repository.

| Variable Name | Description |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RACK_ENV | This variable specifies the environment where the Ruby application will be deployed (unless overwritten) - production, development, test. Each level has different behaviors in terms of logging verbosity, error pages, Ruby gem installation, and other. Note that application assets will be compiled only if the RACK_ENV is set to production. |
| DISABLE_ASSET_COMPILATION | This variable indicates that the asset compilation process will be skipped. Because this only takes place when the application is run in the production environment, it should be used only when assets are already compiled. |
| PUMA_MIN_THREADS, PUMA_MAX_THREADS | These variables indicate the minimum and maximum threads that will be available in Puma's thread pool. |
| PUMA_WORKERS | This variable indicates the number of worker processes that will be launched. See documentation on Puma's clustered mode . |
| RUBYGEM_MIRROR | Set this variable to use a custom RubyGems mirror URL to download required gem packages during the build process. |

For S2I scripts to work, you need to include the **puma** or **rack** gem in the application's Gemfile.

7.6. RUBY ON RAILS

7.6.1. Description

The **rhsc1/ror-50-rhel7** provides a Ruby on Rails 5.0 platform for building and running applications and it contains Ruby 2.4, Ruby on Rails 5.0, and Node.js 6 preinstalled.

The **rhsc1/ror-42-rhel7** provides a Ruby on Rails 4.2 platform for building and running applications and it contains Ruby 2.3, Ruby on Rails 4.2, and Node.js 4 preinstalled.

The **rhsc1/ror-41-rhel7** provides a Ruby on Rails 4.1 platform and it contains Ruby 2.2, Ruby on Rails 4.1, and Node.js 0.10 preinstalled.

7.6.2. Access

To pull the **rhsc1/ror-50-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/ror-50-rhel7
```

To pull the **rhsc1/ror-42-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/ror-42-rhel7
```

To pull the **rhsc1/ror-41-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/ror-41-rhel7
```

7.6.3. Configuration

No further configuration is required.

The **rhsc1/ror-50-rhel7** image contains and enables the **rh-ruby24**, **rh-ror50**, and **rh-nodejs6** Software Collections. The **rhsc1/ror-42-rhel7** image contains and enables the **rh-ruby23**, **rh-ror42**, and **rh-nodejs4** Software Collections. The **rhsc1/ror-41-rhel7** image contains and enables the **rh-ruby22**, **rh-ror41**, and **nodejs010** Software Collections.

For automatic S2I builds, use the Ruby container.

7.7. PHUSION PASSENGER

7.7.1. Description

The **rhsc1/passenger-40-rhel7** image provides a Phusion Passenger 4.0 application server configured with Apache httpd web server. It also provides a Ruby 2.2 platform for building and running applications. Node.js 0.10 is preinstalled for assets compilation.

7.7.2. Access

To pull the **rhsc1/passenger-40-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/passenger-40-rhel7
```

7.7.3. Configuration

No further configuration is required; this image contains and enables the **rh-ruby22**, **rh-ror41**, **nodejs010**, **rh-passenger40**, and **httpd24** Software Collections. It is especially designed to support automatic S2I builds.

7.8. THERMOSTAT AGENT

7.8.1. Description

The **rhsc1/thermostat-16-agent-rhel7** image provides a **thermostat agent** suitable for monitoring Java applications in containers.

The **rhsc1/thermostat-16-agent-rhel7** and **rhsc1/thermostat-1-agent-rhel7** images are no longer supported.

7.8.2. Access

To pull the **rhsc1/thermostat-16-agent-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/thermostat-16-agent-rhel7
```

7.8.3. Usage

The image is intended to be used as a base image for builder and runtime images in your Dockerfile using:

```
FROM rhsc1/thermostat-16-agent-rhel7
```

See this [example](#) of a Dockerfile, which will have the Thermostat agent pre-installed.

Once the **rhsc1/thermostat-16-agent-rhel7** image has been introduced into the image hierarchy, the Thermostat agent can be started by setting the three required Thermostat environment variables. For example, an image called **rhsc1/thermostat-test** uses **rhsc1/thermostat-16-agent-rhel7** as its base image and runs Java app **foo** on deployment. A Thermostat agent can be started together with **foo** by setting the following environment variables:

- THERMOSTAT_AGENT_USERNAME
- THERMOSTAT_AGENT_PASSWORD
- THERMOSTAT_DB_URL

The **rhsc1/thermostat-16-storage-rhel7** image can be used to set up a storage endpoint for the agent to connect to.

7.8.4. Configuration

The image recognizes the following environment variables that you can set during initialization by passing **-e VAR=VALUE** to the **docker run** command:

| Variable Name | Description |
|---------------------------|-----------------------------------------------------------------|
| THERMOSTAT_AGENT_USERNAME | User name for the Thermostat agent to use connecting to storage |
| THERMOSTAT_AGENT_PASSWORD | Password for connecting to storage |

| Variable Name | Description |
|--------------------------|--------------------------------------------------------------------|
| THERMOSTAT_DB_URL | The URL for Thermostat storage |
| APP_USER | The application user the Java app Thermostat shall monitor runs as |

7.9. THERMOSTAT STORAGE

7.9.1. Description

The **rhsc1/thermostat-16-storage-rhel7** contains a Thermostat 1.6 storage, which provides a web endpoint for storing and retrieving data. This image is based on the **rh-thermostat16** Software Collection.

The **rhsc1/thermostat-16-storage-rhel7** image is no longer supported.

7.9.2. Access

To pull the **rhsc1/thermostat-16-storage-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/thermostat-16-storage-rhel7
```

7.9.3. Usage

To run Thermostat storage connected to some other MongoDB backend (for example, provided by another container), you need to provide the MongoDB URL, **mongo** user name and password, and agent and client user names and passwords. Run the following command:

```
# 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 \
  rhsc1/thermostat-16-storage-rhel7
```

This will run a container with the HTTP layer connected to the MongoDB URL using the provided **mongo** user name and password. The container can be accessed at <http://ip-address:8080/thermostat/storage> with the appropriate client or agent credentials that you specified with the environment variables. To find the IP address, run the following command:

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


The command from the example above creates the following agent users:

- **agentuser1** with password **agentpass1**
- **agentuser2** with password **agentpass2**

Either one of them, or both, can be used for Thermostat agent connections.

In addition, the following client users will be usable:

- **clientuser1** with password **clientpass1**
- **clientuser2** with password **clientpass2**

Either one of them, or both, can be used for Thermostat client connections.

7.9.4. Configuration

The **rhsci/thermostat-16-storage-rhel7** image recognizes the following environment variables that you can set during initialization by passing **-e VAR=VALUE** to the **docker run** command:

| Variable Name | Description |
|------------------------------------|---------------------------------------------------------------------------------|
| THERMOSTAT_AGENT_USERNAMES | Space-separated list of user names for Thermostat agents to connect to storage |
| THERMOSTAT_AGENT_PASSWORDS | Space-separated list of passwords for agents connecting to storage |
| THERMOSTAT_CLIENT_USERNAMES | Space-separated list of user names for Thermostat clients to connect to storage |
| THERMOSTAT_CLIENT_PASSWORDS | Space-separated list of passwords for clients connecting to storage |
| MONGO_USERNAME | User name for the MongoDB backing storage |
| MONGO_PASSWORD | Password for the MongoDB backing storage |
| MONGO_URL | MongoDB URL to connect to |

CHAPTER 8. DAEMON IMAGES

8.1. APACHE HTTP SERVER

8.1.1. Description

The **rhsc1/httpd-24-rhel7** image provides an Apache HTTP 2.4 Server. The image can be used as a base image for other applications based on Apache HTTP web server.

8.1.2. Access

To pull the **rhsc1/httpd-24-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/httpd-24-rhel7
```

The **rhsc1/httpd-24-rhel7** image supports using the S2I tool.

8.1.3. Configuration

The **Apache HTTP Server** container image supports the following configuration variable, which can be set by using the **-e** option with the **docker run** command:

| Variable Name | Description |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HTTPD_LOG_TO_VOLUME | By default, httpd logs into standard output, so the logs are accessible by using the docker logs command. When HTTPD_LOG_TO_VOLUME is set, httpd logs into /var/log/httpd24 , which can be mounted to host system using the Docker volumes. |



NOTE

The **rhsc1/httpd-24-rhel7** container image now uses **1001** as the default UID to work correctly within the source-to-image strategy in OpenShift. Additionally, the container image listens on port **8080** by default. Previously, the **rhsc1/httpd-24-rhel7** container image listened on port **80** by default and ran as UID **0**.

To run the **rhsc1/httpd-24-rhel7** container image as UID **0**, specify the **-u 0** option of the **docker run** command:

```
docker run -u 0 rhsc1/httpd-24-rhel7
```

8.2. NGINX

8.2.1. Description

The **rhsc1/nginx-110-rhel7** image provides an nginx 1.10 server and a reverse proxy server; the image can be used as a base image for other applications based on nginx 1.10 web server.

The **rhsc1/nginx-18-rhel7** image provides an nginx 1.8 server and a reverse proxy server; the image can be used as a base image for other applications based on nginx 1.8 web server.

The **rhsc1/nginx-16-rhel7** image is no longer supported.

8.2.2. Access

To pull the **rhsc1/nginx-110-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/nginx-110-rhel7
```

To pull the **rhsc1/nginx-18-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/nginx-18-rhel7
```

8.2.3. Configuration

The **nginx** container images support the following configuration variable, which can be set by using the **-e** option with the **docker run** command:

| Variable Name | Description |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NGINX_LOG_TO_VOLUME | By default, nginx logs into standard output, so the logs are accessible by using the docker logs command. When NGINX_LOG_TO_VOLUME is set, nginx logs into /var/log/nginx110 or /var/log/nginx18 (depending on the version used), which can be mounted to host system using the Docker volumes. |

The **rhsc1/nginx-110-rhel7** and **rhsc1/nginx-18-rhel7** images support using the S2I tool.

8.3. VARNISH CACHE

8.3.1. Description

The **rhsc1/varnish-4-rhel7** image provides Varnish Cache 4.0, an HTTP reverse proxy.

8.3.2. Access

To pull the **rhsc1/varnish-4-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/varnish-4-rhel7
```

8.3.3. Configuration

No further configuration is required.

The **rhscl/varnish-4-rhel7** image supports using the S2I tool. Note that the **default.vcl** configuration file in the directory accessed by S2I needs to be in the [VCL](#) format.

CHAPTER 9. DATABASE IMAGES

9.1. MYSQL

9.1.1. Description

The **rhsc1/mysql-56-rhel7** image provides a MySQL 5.6 SQL database server. The **rhsc1/mysql-57-rhel7** image provides a MySQL 5.7 SQL database server.

9.1.2. Access and Usage

To pull the **rhsc1/mysql-56-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/mysql-56-rhel7
```

To pull the **rhsc1/mysql-57-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/mysql-57-rhel7
```

To set only the mandatory environment variables and not store the database in a host directory, execute the following command:

```
# docker run -d --name mysql_database -e MYSQL_USER=<user> -e
  MYSQL_PASSWORD=<pass> \
  -e MYSQL_DATABASE=<db> -p 3306:3306 rhsc1/mysql-56-rhel7
```

Change the image name if you are using the **rhsc1/mysql-57-rhel7** image.

This will create a container named **mysql_database** running MySQL with database **db** and user with credentials **user:pass**. Port **3306** will be exposed and mapped to the host. If you want your database to be persistent across container executions, also add a **-v /host/db/path:/var/lib/mysql/data:Z** argument. The directory **/host/db/path** will be the MySQL data directory.

If the database directory is not initialized, the entrypoint script will first run **mysql_install_db** and set up necessary database users and passwords. After the database is initialized, or if it was already present, **mysqld** is executed and will run as **PID 1**. You can stop the detached container by running the **docker stop mysql_database** command.

9.1.3. Configuration

The image recognizes the following environment variables that you can set during initialization by passing **-e VAR=VALUE** to the **docker run** command:

| Variable Name | Description |
|-----------------------|-------------------------------------------|
| MYSQL_USER | User name for MySQL account to be created |
| MYSQL_PASSWORD | Password for the user account |

| Variable Name | Description |
|----------------------------|---------------------------------------|
| MYSQL_DATABASE | Database name |
| MYSQL_ROOT_PASSWORD | Password for the root user (optional) |

**NOTE**

The **root** user has no password set by default, only allowing local connections. You can set it by setting the **MYSQL_ROOT_PASSWORD** environment variable when initializing your container. This will allow you to login to the **root** account remotely. Local connections will still not require a password.

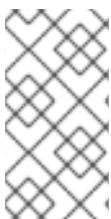
The following environment variables influence the MySQL configuration file and are all optional:

| Variable name | Description | Default |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|
| MYSQL_LOWER_CASE_TABLE_NAMES | Sets how the table names are stored and compared | 0 |
| MYSQL_MAX_CONNECTIONS | The maximum permitted number of simultaneous client connections | 151 |
| MYSQL_MAX_ALLOWED_PACKET | The maximum size of one packet or any generated/intermediate string | 200M |
| MYSQL_FT_MIN_WORD_LEN | The minimum length of the word to be included in a FULLTEXT index | 4 |
| MYSQL_FT_MAX_WORD_LEN | The maximum length of the word to be included in a FULLTEXT index | 20 |
| MYSQL_AIO | Controls the innodb_use_native_aio setting value in case the native AIO is broken. See http://help.directadmin.com/item.php?id=529 | 1 |
| MYSQL_TABLE_OPEN_CACHE | The number of open tables for all threads | 400 |
| MYSQL_KEY_BUFFER_SIZE | The size of the buffer used for index blocks | 32M (or 10% of available memory) |

| Variable name | Description | Default |
|--------------------------------------|----------------------------------------------------------------------------------|------------------------------------|
| MYSQL_SORT_BUFFER_SIZE | The size of the buffer used for sorting | 256K |
| MYSQL_READ_BUFFER_SIZE | The size of the buffer used for a sequential scan | 8M (or 5% of available memory) |
| MYSQL_INNODB_BUFFER_POOL_SIZE | The size of the buffer pool where InnoDB caches table and index data | 32M (or 50% of available memory) |
| MYSQL_INNODB_LOG_FILE_SIZE | The size of each log file in a log group | 8M (or 15% of available available) |
| MYSQL_INNODB_LOG_BUFFER_SIZE | The size of the buffer that InnoDB uses to write to the log files on disk | 8M (or 15% of available memory) |
| MYSQL_DEFAULTS_FILE | Point to an alternative configuration file | /etc/my.cnf |
| MYSQL_BINLOG_FORMAT | Set sets the binlog format, supported values are row and statement | statement |
| MYSQL_LOG_QUERIES_ENABLED | To enable query logging, set this variable to 1 | 0 |

You can also set the following mount point by passing the **-v /host:/container:Z** flag to Docker:

| Volume Mount Point | Description |
|----------------------------|----------------------|
| /var/lib/mysql/data | MySQL data directory |



NOTE

When mounting a directory from the host into the container, ensure that the mounted directory has the appropriate permissions and that the owner and group of the directory matches the user UID or name which is running inside the container, which is 27 by default.

9.2. MARIADB

9.2.1. Description

The **rhsc/mariadb-101-rhel7** image provides a MariaDB 10.1 SQL database server; the **rhsc/mariadb-100-rhel7** image provides a MariaDB 10.0 SQL database server.

9.2.2. Access

To pull the **rhsc1/mariadb-101-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/mariadb-101-rhel7
```

To pull the **rhsc1/mariadb-100-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/mariadb-100-rhel7
```

9.2.3. Usage and Configuration

The usage and configuration is the same as for the MySQL image. For details, see the [MySQL](#) section. Note that the name of the daemon is **mysqld** and all environment variables have the same names as in MySQL. See also [How to Extend the rhsc1/mariadb-101-rhel7 Container Image](#).

9.3. POSTGRESQL

9.3.1. Description

The **rhsc1/postgresql-95-rhel7** image provides a PostgreSQL 9.5 SQL database server; the **rhsc1/postgresql-94-rhel7** image provides a PostgreSQL 9.4 SQL database server.

9.3.2. Access and Usage

To pull the **rhsc1/postgresql-95-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/postgresql-95-rhel7
```

To pull the **rhsc1/postgresql-94-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/postgresql-94-rhel7
```

To set only the mandatory environment variables and not store the database in a host directory, execute the following command:

```
# docker run -d --name postgresql_database -e POSTGRESQL_USER=<user> \
-e POSTGRESQL_PASSWORD=<pass> -e POSTGRESQL_DATABASE=<db> \
-p 5432:5432 rhsc1/postgresql-94-rhel7
```

This will create a container named **postgresql_database** running PostgreSQL with database **db** and user with credentials **user:pass**. Port **5432** will be exposed and mapped to the host. If you want your database to be persistent across container executions, also add a **-v /host/db/path:/var/lib/pgsql/data** argument. This will be the PostgreSQL database cluster directory.

If the database cluster directory is not initialized, the entrypoint script will first run **initdb** and set up necessary database users and passwords. After the database is initialized, or if it was already present, **postgres** is executed and will run as **PID 1**. You can stop the detached container by running the **docker stop postgresql_database** command.

9.3.3. Configuration

The image recognizes the following environment variables that you can set during initialization by passing **-e VAR=VALUE** to the **docker run** command:

| Variable Name | Description |
|----------------------------------|----------------------------------------------------|
| POSTGRESQL_USER | User name for PostgreSQL account to be created |
| POSTGRESQL_PASSWORD | Password for the user account |
| POSTGRESQL_DATABASE | Database name |
| POSTGRESQL_ADMIN_PASSWORD | Password for the postgres admin account (optional) |



NOTE

The **postgres** administrator account has no password set by default, only allowing local connections. You can set it by setting the **POSTGRESQL_ADMIN_PASSWORD** environment variable when initializing your container. This will allow you to login to the **postgres** account remotely. Local connections will still not require a password.

The following environment variables influence the PostgreSQL configuration file and are both optional:

| Variable Name | Description | Default |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------|---------|
| POSTGRESQL_MAX_CONNECTIONS | The maximum number of client connections allowed. This also sets the maximum number of prepared transactions. | 100 |
| POSTGRESQL_SHARED_BUFFERS | Sets how much memory is dedicated to PostgreSQL to use for caching data | 32M |

You can also set the following mount point by passing the **-v /host:/container** flag to Docker:

| Volume Mount Point | Description |
|----------------------------|---------------------------------------|
| /var/lib/pgsql/data | PostgreSQL database cluster directory |

**NOTE**

When mounting a directory from the host into the container, ensure that the mounted directory has the appropriate permissions and that the owner and group of the directory matches the user UID or name which is running inside the container.

9.4. MONGODB

9.4.1. Description

The **rhsc/mongodb-32-rhel7** image provides a MongoDB 3.2 NoSQL database server. The **rhsc/mongodb-26-rhel7** image provides a MongoDB 2.6 NoSQL database server.

9.4.2. Access and Usage

To pull the **rhsc/mongodb-32-rhel7** image, run the following command as **root**:

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

To pull the **rhsc/mongodb-26-rhel7** image, run the following command as **root**:

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

To set only the mandatory environment variables and store the database in the **/home/user/database** directory on the host file system, execute the following command:

```
# 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 rhsc/mongodb-26-rhel7
```

Change the image name when appropriate.

If you are initializing the database and it is the first time you are using the specified shared volume, the database will be created with two users: **admin** and **MONGODB_USER**. After that, the MongoDB daemon will be started. If you are re-attaching the volume to another container, the creation of the database user and **admin** user will be skipped and only the MongoDB daemon will be started.

9.4.3. Configuration

The image recognizes the following environment variables that you can set during initialization by passing **-e VAR=VALUE** to the **docker run** command:

| Variable Name | Description |
|-------------------------|----------------------------------------------------|
| MONGODB_USER | User name for MONGODB account to be created |
| MONGODB_PASSWORD | Password for the user account |

| Variable Name | Description |
|-------------------------------|------------------------------------|
| MONGODB_DATABASE | Database name |
| MONGODB_ADMIN_PASSWORD | Password for the admin user |

**NOTE**

The administrator user name is set to **admin** and you have to specify the password by setting the **MONGODB_ADMIN_PASSWORD** environment variable. This process is done upon database initialization.

The following environment variables influence the MongoDB configuration file and are all optional:

| Variable Name | Description | Default |
|---------------------------|--------------------------------------------------------------------------|-------------|
| MONGODB_NOPREALLOC | Disable data file preallocation | true |
| MONGODB_SMALLFILES | Set MongoDB to use a smaller default data file size | true |
| MONGODB_QUIET | Runs MongoDB in a quiet mode that attempts to limit the amount of output | true |

**NOTE**

In the **rhsc1/mongodb-32-rhel7** image, the **MONGODB_NOPREALLOC** and **MONGODB_SMALLFILES** options are not effective.

You can also set the following mount point by passing the **-v /host:/container** flag to Docker:

| Volume Mount Point | Description |
|------------------------------|------------------------|
| /var/lib/mongodb/data | MongoDB data directory |

**NOTE**

When mounting a directory from the host into the container, ensure that the mounted directory has the appropriate permissions and that the owner and group of the directory matches the user UID or name which is running inside the container.

9.4.4. Custom configuration file

It is possible to use a custom configuration file for the **mongod** server. Providing a custom configuration file supersedes the environment variable values of an individual configuration.

A custom configuration file used in a container has to be mounted into **/etc/mongod.conf**. For example, to use a configuration file stored in the **/home/user** directory, use the following option for the **docker run** command: **-v /home/user/mongod.conf:/etc/mongod.conf:Z**.



NOTE

The custom configuration file does not affect the name of a replica set. The replica set name has to be set in the **MONGODB_REPLICA_NAME** environment variable.

9.5. REDIS

9.5.1. Description

The **rhsc/redis-32-rhel7** image provides Redis 3.2, an advanced key-value store. The image is based on the **rh-redis32** Software Collection.

9.5.2. Access

To pull the **rhsc/redis-32-rhel7** image, run the following command as **root**:

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

9.5.3. Configuration

The following environment variable influences the Redis configuration file and is optional:

| Variable Name | Description |
|-----------------------|--------------------------------|
| REDIS_PASSWORD | Password for the server access |

You can also set the following mount point by passing the **-v /host:/container** flag to Docker:

| Volume Mount Point | Description |
|----------------------------|----------------------|
| /var/lib/redis/data | Redis data directory |



NOTE

When mounting a directory from the host into the container, ensure that the mounted directory has the appropriate permissions and that the owner and group of the directory matches the user UID or name that is running inside the container. The default UID for this container is **1001**.

CHAPTER 10. RED HAT DEVELOPER TOOLSET IMAGES

10.1. RUNNING RED HAT DEVELOPER TOOLSET TOOLS FROM PRE-BUILT CONTAINER IMAGES

To display general usage information for pre-built Red Hat Developer Toolset docker-formatted container images that you have already pulled to your local machine, run the following command as **root**:

```
# docker run image_name usage
```

To launch an interactive shell within a pre-built docker-formatted container image, run the following command as **root**:

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

In both of the above commands, substitute the *image_name* parameter with the name of the container image you pulled to your local system and now want to use.

For example, to launch an interactive shell within the container image with selected toolchain components, run the following command as **root**:

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

Example 10.1. Using GCC in the Pre-Built Red Hat Developer Toolset Toolchain Image

This example illustrates how to obtain and launch the pre-built docker-formatted container image with selected toolchain components of the Red Hat Developer Toolset and how to run the **gcc** compiler within that image.

1. Make sure you have a **Docker** environment set up properly on your system by following instructions at [Getting Docker in RHEL 7](#).
2. Pull the pre-built toolchain Red Hat Developer Toolset container image from the official Red Hat Container Registry:

```
# docker pull rhsc/ devtoolset-6-toolchain-rhel7
```

3. To launch the container image with an interactive shell, issue the following command:

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

4. To launch the container as a regular (non-root) user, use the **sudo** command. To map a directory from the host system to the container file system, include the **-v** (or **--volume**) option in the **docker** command:

```
$ sudo docker run -v ~/Source:/src -ti rhsc/ devtoolset-6-toolchain-rhel7 /bin/bash -l
```

In the above command, the host's **~/Source/** directory is mounted as the **/src/** directory within the container.

- Once you are in the container's interactive shell, you can run Red Hat Developer Toolset tools as expected. For example, to verify the version of the **gcc** compiler, run:

```
bash-4.2$ gcc -v
[...]
gcc version 6.3.1 20170216 (Red Hat 6.3.1-3) (GCC)
```

10.2. USING CONTAINER IMAGES BUILT FROM DOCKERFILES

Dockerfiles are available for selected Red Hat Developer Toolset components. Dockerfiles are text files that contain instructions for automated building of docker-formatted container images.

Red Hat Developer Toolset 6.1 for Red Hat Enterprise Linux 7 is shipped with the following Dockerfiles:

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

10.2.1. Obtaining Dockerfiles

The Red Hat Developer Toolset Dockerfiles are provided by the package **devtoolset-6-dockerfiles**. The package contains individual Dockerfiles for building docker-formatted container images with individual components and a meta-Dockerfile for building a docker-formatted container image with all the components offered. To be able to use the Dockerfiles, install this package by executing:

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

Use the RHSM channel **rhel-server-rhsc1-7-rpms**. In order to enable it, follow the instructions at [Getting Access to Red Hat Developer Toolset](#)

10.2.2. Building Container Images

Change to the directory where the Dockerfile is installed and run the following command as **root**:

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

Replace *image_name* with the desired name for the new image.

Example 10.2. Building a Container Image with a Red Hat Developer Toolset Component

To build a docker-formatted container image for deploying the **perftools** tools in a container, follow the instructions below:

1. Make sure you have a **Docker** environment set up properly on your system by following instructions at [Getting Docker in RHEL 7](#).
2. Install the package containing the Red Hat Developer Toolset Dockerfiles:

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

3. Determine where the Dockerfile for the required component is located:

```
# 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. Change to the directory where the required Dockerfile is installed:

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

5. Build the container image:

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

Replace *devtoolset-6-my-perftools* with the name you wish to assign to your resulting container image.

10.2.3. Running Red Hat Developer Toolset Tools from Custom-Built Container Images

To display general usage information for images built from Red Hat Developer Toolset Dockerfiles (see [Section 10.2.2, “Building Container Images”](#)), run the following command as **root**:

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

To launch an interactive shell within a docker-formatted container image you built, run the following command as **root**:

```
docker run -ti image_name /bin/bash -l
```

In both of the above commands, substitute the *image_name* parameter with the name of the container image you chose when building it.

Example 10.3. Using elfutils in a Custom-Built Red Hat Developer Toolset Image

This example illustrates how to launch a custom-built docker-formatted container image with the **elfutils** component and how to run the **eu-size** tool within that image.

1. To launch the container image with an interactive shell, issue the following command:

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

- To launch the container as a regular (non-root) user, use the **sudo** command. To map a directory from the host system to the container file system, include the **-v** (or **--volume**) option in the **docker** command:

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

In the above command, the host's **~/Source/** directory is mounted as the **/src/** directory within the container.

- Once you are in the container's interactive shell, you can run Red Hat Developer Toolset tools as expected. For example, to verify the version of the **eu-size** tool, run:

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

10.3. RED HAT DEVELOPER TOOLSET TOOLCHAIN

10.3.1. Description

The Red Hat Developer Toolset Toolchain images provide the GNU Compiler Collection (GCC) and GNU Debugger (GDB).

The **rhsc/devtoolset-6-toolchain-rhel7** image provides content corresponding to the following packages:

| Component | Version | Package |
|-----------------|---------|--------------------------|
| gcc | 6.3.1 | devtoolset-6-gcc |
| g++ | | devtoolset-6-gcc-c++ |
| gfortran | | devtoolset-6-gcc-fortran |
| gdb | 7.12.1 | devtoolset-6-gdb |

The **rhsc/devtoolset-4-toolchain-rhel7** image provides content corresponding to the following packages:

| Component | Version | Package |
|------------|---------|----------------------|
| gcc | 5.3.1 | devtoolset-4-gcc |
| g++ | | devtoolset-4-gcc-c++ |

| Component | Version | Package |
|-----------------|---------|--------------------------|
| | | |
| gfortran | | devtoolset-4-gcc-fortran |
| gdb | 7.11 | devtoolset-4-gdb |

10.3.2. Access

To pull the **rhsc1/devtoolset-6-toolchain-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/devtoolset-6-toolchain-rhel7
```

To pull the **rhsc1/devtoolset-4-toolchain-rhel7** image, run the following command as **root**:

```
# docker pull registry.access.redhat.com/rhsc1/devtoolset-4-toolchain-rhel7
```

10.4. RED HAT DEVELOPER TOOLSET PERFORMANCE TOOLS

10.4.1. Description

The Red Hat Developer Toolset Performance Tools images provide a number of profiling and performance measurement tools.

The **rhsc1/devtoolset-6-perftools-rhel7** image provides the following components:

| Component | Version | Package |
|------------------|---------|------------------------|
| 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 |

The **rhsc1/devtoolset-4-perftools-rhel7** image provides the following components:

| Component | Version | Package |
|------------------|---------|------------------------|
| 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. Access

To pull the **rhsc/devtoolset-6-perftools-rhel7** image, run the following command as **root**:

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

To pull the **rhsc/devtoolset-4-perftools-rhel7** image, run the following command as **root**:

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

10.4.3. Usage

Using the SystemTap Tool from Container Images

When using the **SystemTap** tool from a container image, additional configuration is required, and the container needs to be run with special command-line options.

The following three conditions need to be met:

1. The image needs to be run with super-user privileges. To do this, run the image using the following command:

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

To use the pre-built **perftools** image, substitute the image name for **devtoolset-6-perftools-rhel7** in the above command.

2. The following kernel packages need to be installed in the container:

- **kernel**
- **kernel-devel**
- **kernel-debuginfo**

The version and release numbers of the above packages must match the version

and release numbers of the kernel running on the host system. Run the following command to determine the version and release numbers of the hosts system's kernel:

```
~]$ uname -r  
3.10.0-514.10.2.el7.x86_64
```

Note that the **kernel-debuginfo** package is only available from the *Debug* channel. Enable the **rhel-7-server-debug-rpms** repository as described in TODO WHERE. For more information on how to get access to debuginfo packages, see <https://access.redhat.com/site/solutions/9907>.

To install the required packages with the correct version, use the **yum** package manager and the output of the **uname** command. For example, to install the correct version of the **kernel** package, run the following command as **root**:

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

3. Save the container to a reusable image by executing the **docker commit** command. To save a custom-built **SystemTap** container:

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