Red Hat Software Collections 2.x
2.0 Release Notes

Release Notes for Red Hat Software Collections 2.0

Lenka Špačková Jaromír Hradílek Eliška Slobodová

Red Hat Software Collections 2.x 2.0 Release Notes

Release Notes for Red Hat Software Collections 2.0

Lenka Špačková
Red Hat Customer Content Services
lspackova@redhat.com

Jaromír Hradílek
Red Hat Customer Content Services
jhradilek@redhat.com

Eliška Slobodová
Red Hat Customer Content Services

Abstract

The Red Hat Software Collections 2.0 Release Notes document the major features and contain important information about known problems in Red Hat Software Collections 2.0. The Red Hat Developer Toolset collection is documented in the Red Hat Developer Toolset Release Notes and the Red Hat Developer Toolset User Guide.

Chapter 1. Red Hat Software Collections 2.0

This chapter serves as an overview of the Red Hat Software Collections 2.0 content set. It provides a list of components and their descriptions, sums up changes in this version, documents relevant compatibility information, and lists known issues.

1.1. About Red Hat Software Collections

For certain applications, more recent versions of some software components are often needed in order to use their latest new features. **Red Hat Software Collections** is a Red Hat offering that provides a set of dynamic programming languages, database servers, and various related packages that are either more recent than their equivalent versions included in the base Red Hat Enterprise Linux system, or are available for this system for the first time. For a complete list of components that are distributed as part of Red Hat Software Collections and a brief summary of their features, see Section 1.2, “Main Features”.

Red Hat Software Collections does not replace the default system tools provided with Red Hat Enterprise Linux 6 or Red Hat Enterprise Linux 7. Instead, a parallel set of tools is installed in the **`/opt/`** directory and can be optionally enabled per application by the user using the supplied **`scl`** utility. The default versions of Perl or PostgreSQL, for example, remain those provided by the base Red Hat Enterprise Linux system.

All Red Hat Software Collections components are fully supported under Red Hat Enterprise Linux Subscription Level Agreements, are functionally complete, and are intended for production use. Important bug fix and security errata are issued to Red Hat Software Collections subscribers in a similar manner to Red Hat Enterprise Linux for at least three years from the release of each major version. A new major version of Red Hat Software Collections is released approximately every 18 months, and in each major release stream, each version of a selected component remains backward compatible. For detailed information about length of support for individual components, refer to the Red Hat Software Collections Product Life Cycle document.

Red Hat Developer Toolset is now part of Red Hat Software Collections, included as a separate Software Collection. For more information about Red Hat Developer Toolset, refer to the Red Hat Developer Toolset Release Notes and the Red Hat Developer Toolset User Guide.

1.2. Main Features

Red Hat Software Collections 2.0 provides recent stable versions of the tools listed in Table 1.1, “Red Hat Software Collections 2.0 Components”.

Table 1.1. Red Hat Software Collections 2.0 Components

Component	Software Collection	Description
Red Hat Developer Toolset 3.1	*devtoolset-3*	Red Hat Developer Toolset is designed for developers working on the Red Hat Enterprise Linux platform. It provides current versions of the **GNU Compiler Collection**, **GNU Debugger**, **Eclipse** development platform, and other development, debugging, and performance monitoring tools. For a complete list of components, see the Red Hat Developer Toolset Components table in the *Red Hat Developer Toolset User Guide*.

Component	Software Collection	Description
Perl 5.20.1	*rh-perl520*	A release of Perl, a high-level programming language that is commonly used for system administration utilities and web programming. The *rh-perl520* Software Collection provides additional utilities, scripts, and *database connectors for MySQL and PostgreSQL*. Also, it includes the **DateTime** Perl module and the **mod_perl** Appache httpd module, which is supported only with the *httpd24* Software Collection.
PHP 5.4.40	*php54*	A release of PHP with **PEAR 1.9.4** and a number of additional extensions. PHP 5.4 provides a number of *language and interface improvements*. The **memcache** and **Zend OPcache** extensions are also included.
PHP 5.5.21	*php55*	A release of PHP with **PEAR 1.9.4** and enhanced language features including *better exception handling, generators,* and **Zend OPcache**. The **memcache** and **mongodb** extensions are also included.
PHP 5.6.5	*rh-php56*	A release of PHP with **PEAR 1.9.5** and enhanced language features including *constant expressions, variadic functions, arguments unpacking, and the interactive debugger*. The **memcache**, **mongo**, and **XDebug** extensions are also included.
Python 2.7.8	*python27*	A release of Python 2.7 with a number of additional utilities. This Python version provides various new features and enhancements, including a new ordered dictionary type, faster I/O operations, and improved forward compatibility with Python 3. The *python27* Software Collections contains the *Python 2.7.8 interpreter*, a set of extension libraries useful for programming web applications and **mod_wsgi** (only supported with the *httpd24* Software Collection), MySQL and PostgreSQL database connectors, and **numpy** and **scipy**.
Python 3.4.2	*rh-python34*	A release of Python 3 with a number of additional utilities. This Software Collection gives developers on Red Hat Enterprise Linux access to Python 3 and allows them to benefit from various advantages and new features of this version. The *rh-python34* Software Collection contains *Python 3.4.2 interpreter*, a set of extension libraries useful for programming web applications and **mod_wsgi** (only supported with the *httpd24* Software Collection), PostgreSQL database connector, and **numpy** and **scipy**.
Ruby 2.2.2	*rh-ruby22*	A release of Ruby 2.2. This version provides substantial *performance and reliability improvements, including incremental and symbol garbage collection* and many others, while maintaining source level backward compatibility with Ruby 2.0.0 and Ruby 1.9.3.

Component	Software Collection	Description
Ruby on Rails 4.1.5	*rh-ror41*	A release of Ruby on Rails 4.1, a web application development framework written in the Ruby language. This version provides a number of new features including *Spring application preloader, config/secrets.yml, Action Pack variants, and Action Mailer previews*. This Software Collection is supported together with the *rh-ruby22* Collection.
MariaDB 10.0.17	*rh-mariadb100*	A release of MariaDB, *an alternative to MySQL* for users of Red Hat Enterprise Linux. For all practical purposes, MySQL is binary compatible with MariaDB and can be replaced with it without any data conversions. This version adds the PAM authentication plugin to MariaDB.
MongoDB 2.6.9	*rh-mongodb26*	A release of MongoDB, a cross-platform *document-oriented database system classified as a NoSQL database*. This Software Collection includes the *mongo-java-driver* package.
MySQL 5.6.24	*rh-mysql56*	A release of MySQL, which provides a number of new features and enhancements, including improved performance.
PostgreSQL 9.4.1	*rh-postgresql94*	A release of PostgreSQL, which provides a number of enhancements, including *improved scalability* (bi-directonal replication, cascading replication), increased flexibility of native JSON support, and improved performance.
Node.js 0.10	*nodejs010*	A release of Node.js with **npm 1.4.28** and *support for the SPDY protocol version 3.1*. This Software Collection gives users of Red Hat Enterprise Linux access to this programming platform.
nginx 1.6.2	*nginx16*	A release of nginx, a web and proxy server with a focus on high concurrency, performance and low memory usage. This version introduces a number of new features, including various *SSL improvements*, support for *SPDY 3.1, cache revalidation with conditional requests*, and *authentication request module*.
Apache httpd 2.4.12	*httpd24*	A release of the Apache HTTP Server (httpd), including a high performance *event-based processing model, enhanced SSL module and FastCGI support*. The **mod_auth_kerb** module is also included.
Thermostat 1.2.0	*thermostat1*	A release of Thermostat, a monitoring and instrumentation tool for the *OpenJDK HotSpot JVM*, with support for monitoring *multiple JVM instances*. This Software Collection depends on the *rh-mongodb26* and *rh-java-common* components.

Component	Software Collection	Description
DevAssistant 0.9.3	*devassist09*	A release of DevAssistant, a tool designed to assist developers with *creating and setting up basic projects* in various programming languages, installing dependencies, setting up a development environment, and working with source control. DevAssistant supports the C, C++, Java, and Python programming languages but it is able to support working with any other language, framework, or tool due to its modular architecture.
Maven 3.0.5	*maven30*	A release of Maven, a *software project management and comprehension tool* used primarily for Java projects. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting, and documentation from a central piece of information.
Passenger 4.0.50	*rh-passenger40*	A release of Phusion Passenger, a web and application server, designed to be fast, robust, and lightweight. It supports **Ruby** using the *ruby193*, *ruby200*, or *rh-ruby22* Software Collections together with **Ruby on Rails** using the *ror40* or *rh-ror41* Collections. It can also be used with **nginx 1.6** from the *nginx16* Software Collection and with **Apache httpd** from the *httpd24* Software Collection.
Common Java Packages 1.1	*rh-java-common*	This Software Collection provides *common Java libraries and tools* used by other collections. The *rh-java-common* Software Collection is required by the *devtoolset-3*, *maven30*, *rh-mongodb26*, and *thermostat1* components.
V8 3.14.5.10	*v8314*	This Software Collection provides the *V8 JavaScript engine* and is supported only as a dependency for the *mongodb24*, *rh-mongodb26*, *ruby193*, *ror40*, *rh-ror41*, and *nodejs010* Software Collections.

Previously released Software Collections remain available in the same distribution channels. For example, the *git19* Software Collection, which provides **Git 1.9.4**, has not been updated since Red Hat Software Collections 1.2 but still can be installed along with the Red Hat Software Collections 2.0 components or other previously released components.

All currently available Software Collections are listed in the Table 1.2, “All Available Software Collections”. For detailed information regarding components that have not been updated since Red Hat Software Collections 1, refer to the Red Hat Software Collections 1.2 Release Notes. See the Red Hat Software Collections Product Life Cycle document for information on the length of support for individual components.

Table 1.2. All Available Software Collections

Component	Software Collection
Components New in Red Hat Software Collections 2.0	
Perl 5.20.1	*rh-perl520*
PHP 5.6.5	*rh-php56*
Python 3.4.2	*rh-python34*
Ruby 2.2.2	*rh-ruby22*

Component	Software Collection
Components New in Red Hat Software Collections 2.0	
Ruby on Rails 4.1.5	*rh-ror41*
MariaDB 10.0.17	*rh-mariadb100*
MongoDB 2.6.9	*rh-mongodb26*
MySQL 5.6.24	*rh-mysql56*
PostgreSQL 9.4.1	*rh-postgresql94*
Passenger 4.0.50	*rh-passenger40*
Common Java Packages 1.1	*rh-java-common*
Components Updated in Red Hat Software Collections 2.0	
Red Hat Developer Toolset 3.1	*devtoolset-3*
PHP 5.4.40	*php54*
PHP 5.5.21	*php55*
Python 2.7.8	*python27*
Node.js 0.10	*nodejs010*
nginx 1.6.2	*nginx16*
Apache httpd 2.4.12	*httpd24*
Thermostat 1.2.0	*thermostat1*
DevAssistant 0.9.3	*devassist09*
Maven 3.0.5	*maven30*
V8 3.14.5.10	*v8314*
Components Not Updated since Red Hat Software Collections 1	
Git 1.9.4	*git19*
Perl 5.16.3	*perl516*
Python 3.3.2	*python33*
Ruby 1.9.3	*ruby193*
Ruby 2.0.0	*ruby200*
Ruby on Rails 4.0.2	*ror40*
MariaDB 5.5.37	*mariadb55*
MongoDB 2.4.9	*mongodb24*
MySQL 5.5.37	*mysql55*
PostgreSQL 9.2.8	*postgresql92*

1.3. Changes in Red Hat Software Collections 2.0

1.3.1. Overview

New Software Collections

Red Hat Software Collections 2.0 adds these new Software Collections:

- *rh-java-common* — this Software Collection provides common Java libraries and tools used by other collections. The *rh-java-common* component is required by the *devtoolset-3*, *maven30*, *rh-mongodb26*, and *thermostat1* Software Collections.
- *rh-mariadb100* — see Section 1.3.8, “Changes in MariaDB”
- *rh-mongodb26* — see Section 1.3.9, “Changes in MongoDB”

- *rh-mysql56* — see Section 1.3.10, “Changes in MySQL”
- *rh-passenger40* — see Section 4.7, “Passenger”
- *rh-perl520* — see Section 1.3.3, “Changes in Perl”
- *rh-php56* — see Section 1.3.4, “Changes in PHP”
- *rh-postgresql94* — see Section 1.3.11, “Changes in PostgreSQL”
- *rh-python34* — see Section 1.3.5, “Changes in Python”
- *rh-ruby22* — see Section 1.3.6, “Changes in Ruby”
- *rh-ror41* — see Section 1.3.7, “Changes in Ruby on Rails”

Updated Software Collections

The following components have been updated in Red Hat Software Collections 2.0:

- *devtoolset-3* — see Section 1.3.2, “Changes in Red Hat Developer Toolset”
- *php54* — see Section 1.3.4, “Changes in PHP”
- *php55* — see Section 1.3.4, “Changes in PHP”
- *python27* — see Section 1.3.5, “Changes in Python”
- *nodejs010* — see Section 1.3.12, “Changes in Node.js”
- *nginx16* — see Section 1.3.13, “Changes in nginx”
- *httpd24* — see Section 1.3.14, “Changes in Apache httpd”
- *thermostat1* — see Section 1.3.15, “Changes in Thermostat”
- *devassist09* — see Section 1.3.16, “Changes in DevAssistant”

The further detailed sections describe changes since Red Hat Software Collections 1.2.

1.3.2. Changes in Red Hat Developer Toolset

Red Hat Software Collections 2.0 is released with Red Hat Developer Toolset 3.1. The following components have been upgraded in this release:

- **Eclipse** to version 4.4.2
- **GCC** to version 4.9.2
- **elfutils** to version 0.161
- **GDB** to version 7.8.2
- **SystemTap** to version 2.6
- **Valgrind** to version 3.10.1
- **Dyninst** to version 8.2.1

The Red Hat Developer Toolset 3.1 release also includes a bug fix update of **ltrace** and enhancement update of **memstomp**. For detailed information on changes in Red Hat Developer Toolset, see Red Hat Developer Toolset User Guide.

Red Hat Developer Toolset 3.1 introduces the *devtoolset-3-dockerfiles* subpackage for Red Hat Enterprise Linux 7. This package contains Dockerfiles for selected Red Hat Developer Toolset components, including their Red Hat Enterprise Linux 6 versions, which can be deployed only on Red Hat Enterprise Linux 7 Server. For details, see Red Hat Developer Toolset User Guide.

1.3.3. Changes in Perl

Perl 5.20.1, shipped in the new *rh-perl520* Software Collection, introduces various changes and improvements, for example:

- Hashes have been randomized by default; the order in which keys and values are returned from a hash now changes on each **Perl** run
- Enabling locale now affects the character type
- Support for Unicode 6.3 has been added
- New hash slices have been added.

1.3.4. Changes in PHP

PHP 5.4

The *php54* Software Collection has been upgraded to version 5.4.40, which provides a number of bug fixes over the version shipped in Red Hat Software Collections 1.

PHP 5.5

The updated *php55* Software Collection includes **PHP 5.5.21** with multiple bug fixes over the version shipped in Red Hat Software Collections 1.

PHP 5.6

The new *rh-php56* Software Collection includes **PHP 5.6.5** with **PEAR 1.9.5** and the **memcache**, **mongo**, and **XDebug** extensions. This version provides a number of language and interface improvements. Refer to the upstream documentation on migration and the documentation for the PHP Interactive Debugger, which is provided by the *rh-php56-php-dbg package*.

1.3.5. Changes in Python

Python 2

The *python27* Software Collection has been upgraded to version 2.7.8, which provides numerous security and bug fixes. This Software Collection now includes the **python-wheel** and **python-pip** modules.

Python 3

The new *rh-python34* Software Collection includes **Python 3.4.2**, which provides numerous security fixes and several new features. Among others:

- The **pathlib** module providing object-oriented file system path
- Enumerated type (enum) is now part of the Python standard library (PEP 435)
- Import-related standard library module changes
- A new statistics module
- The **asyncio** module, which enables writing code that concurrently handles asynchronous network based interactions.

This update also includes several changes to improve security, for example:

- Certificates are now verified by default in the **httplib** module
- TLSv1.1 and TLSv1.2 support for SSL has been added
- Server-side Server Name Indication (SNI) support for SSL has been added.

1.3.6. Changes in Ruby

The new *rh-ruby22* Software Collection contains **Ruby 2.2.2**, which provides substantial performance and reliability improvements, including:

- A new incremental garbage collection (GC) algorithm has been included
- Symbols are now garbage collectable
- Minor improvements on the core classes and the standard library have been introduced.

Ruby 2.2 is backward compatible with **Ruby 2.0.0** and **Ruby 1.9.3**. The *ruby193* and *ruby200* Software Collections are still available. For information about length of support for these components, refer to the Red Hat Software Collections Product Life Cycle document. Note that upstream development of **Ruby 1.9.3** has been terminated and it is advisable to migrate to the *rh-ruby22* Software Collection.

1.3.7. Changes in Ruby on Rails

Ruby on Rails 4.1.5, shipped in the new *rh-ror41* Software Collection, provides the following major new features:

- Spring Application Preloader to speed up development
- The **`config/secrets.yml`** file, which can be used to store multiple secrets and access keys
- Action Pack Variants to render different templates for phones, tablets, and browsers
- Action Mailer Previews for email viewing
- Active Record enums
- Message Verifiers to generate and verify signed messages
- A new **`Module#concerning`** to separate responsibilities within a class
- Cross-site request forgery (CSRF) protection from remote <script> tags.

The *rh-ror41* Software Collection is supported together with the *rh-ruby22* Collection.

1.3.8. Changes in MariaDB

The new *rh-mariadb100* Software Collection includes **MariaDB 10.0.17**, which provides a number of bug fixes, performance improvements, and enhancements over the version shipped in Red Hat Software Collections 1. The most notable changes are:

- Parallel replication, which enables **MariaDB** to execute queries on the slave in parallel
- Global transaction ID, which allows to easily change a slave server to connect to and a master server to replicate from; the state of the slave is recorded in a crash-safe way
- Multi-source replication, which means that one server has multiple masters from which it replicates
- New NoSQL features that add access to diverse data sources dynamically
- New sharding features that allow database tables to be split across servers.

For more information regarding features in **MariaDB 10.0**, refer to the upstream resources. For information about migrating to the *rh-mariadb100* Software Collection, see Section 5.1, "Migrating to MariaDB 10.0".

For all practical purposes, **MariaDB** is a binary drop in replacement of the same **MySQL** version. For example, **My SQL5.5** is compatible with **MariaDB 5.5** and also in practice with **MariaDB 10.0**. For more information about **MariaDB** and **MySQL** compatibility, see the MariaDB documentation. Incompatibilities between **MariaDB 10.0** and **MySQL 5.6** are described in this section.

1.3.9. Changes in MongoDB

MongoDB 2.6.9, included in the new *rh-mongodb26* Software Collection, provides a number of bug fixes and enhancements over the version shipped in Red Hat Software Collections 1. For example:

- Aggregation enhancements — the aggregation pipeline adds the ability to return result sets of any size, either by returning a cursor or writing the output to a collection
- Text search integration — text search is now enabled by default and the query system includes the **`$text`** operator, which resolves text-search queries
- Improvements to the update and insert systems, which include additional operations and improvements that increase consistency of modified data
- A new authorization model that provides the ability to create custom User-Defined Roles and the ability to specify user privileges at a collection-level granularity.

For detailed information on changes in **MongoDB 2.6**, refer to the MongoDB documentation. For information about migrating to the *rh-mongodb26* Software Collection, see Section 5.2, "Migrating to MongoDB 2.6".

1.3.10. Changes in MySQL

The new *rh-mysql56* Software Collection includes **MySQL 5.6.24**, which provides a number of bug fixes, performance improvements, and enhancements over the version shipped in Red Hat Software Collections 1. Among others:

- Parallel replication, which enables **MySQL** to execute queries on the slave in parallel
- Global transaction ID, which allows to easily change a slave server to connect to and a master server to replicate from; the state of the slave is recorded in a crash-safe way
- InnoDB memcached plug-in, which enables direct access to InnoDB tables using the memcached

protocol and client libraries

- New NoSQL-style memcached APIs
- Optimizer improvements for all-around query performance
- Partitioning improvements for querying and managing huge tables
- Improved performance monitoring using the Performance Schema.

For more information about changes in **MySQL 5.6**, refer to the MySQL documentation. For information about migrating to the *rh-mysql56* Software Collection, see Section 5.3, “Migrating to MySQL 5.6”.

1.3.11. Changes in PostgreSQL

PostgreSQL 9.4.1, provided by the new *rh-postgresql94* Software Collection, includes the following most notable changes:

- Increased flexibility with the new JSONB datatype, which enables users to use both relational and non-relational data stores at the same time
- Increased scalability with Logical Decoding that supplies a new API for reading, filtering and manipulating the PostgreSQL replication stream. This interface is the foundation for new replication tools, such as Bi-Directional Replication.
- Increased performance with improvements to GIN indexes, concurrently updatable Materialized Views for faster, more up-to-date reporting, parallel writing to the transaction log, and support for Linux huge pages.
- Event trigger support for DDL
- Improved materialized view, which can, for example, be refreshed without blocking concurrent reads
- Updatable views

For detailed changes, see the PostgreSQL 9.3 Release Notes and the PostgreSQL 9.4 Release Notes. For information about migrating to the *rh-postgresql94* Software Collection, see Section 5.4, “Migrating to PostgreSQL 9.4”.

1.3.12. Changes in Node.js

The *nodejs010* Software Collection has been upgraded to upstream version 0.10.35, which provides a number of bug fixes and enhancements over the version shipped in Red Hat Software Collections 1. Among others:

- Support for the SPDY protocol version 3.1 has been included for both Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7
- The *nodejs010* Software Collection is now fully supported.

1.3.13. Changes in nginx

The *nginx16* Software Collection has been upgraded to version 1.6.2, which provides several bug fixes and enhancements over the version shipped in Red Hat Software Collections 1. For example:

- Support for **Passenger** has been added — see Section 4.7, “Passenger” for details

- This update includes support for SPDY 3.1 for both Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7.

1.3.14. Changes in Apache httpd

The *httpd24* Software Collection has been upgraded to version 2.4.12, which provides numerous bug fixes and enhancements over the version shipped in Red Hat Software Collections 1. Among others:

- Support for **Passenger** has been added — see Section 4.7, “Passenger” for details
- Support for Elliptic curve Diffie–Hellman (ECDH) has been added
- Support for Unix Domain Socket (UDS) in the **mod_proxy_fcgi** module has been improved
- Installation of the **mod_ssl** module in FIPS mode has been fixed.

1.3.15. Changes in Thermostat

The *thermostat1* Software Collection has been upgraded to version 1.2.0, which introduces several new features:

- A new instrumenting profiler plug-in has been added
- The setup of secured **Thermostat** using web storage has been simplified; the default setup has been changed to use HTTP-based storage
- Various improvements have been introduced, for example, in the Swing client GUI and in a number of charts.

Note that data migration and automatic user plug-in migration is not supported from **Thermostat 1.0.4** to **Thermostat 1.2**. For details, refer to the Thermostat documentation.

1.3.16. Changes in DevAssistant

The *devassist09* Software Collection has been upgraded to version 0.9.3, which provides various bug fixes and several minor improvements, for example:

- GitHub token is no longer logged for security reasons
- Icons in the PNG format are supported in the DevAssistant GUI
- Error messages can now be ignored when running commands in assistants.

For information about possible incompatibility with the previous version of **DevAssistant**, see Section 4.5.4, “Backward Compatibility in DevAssistant”.

1.4. Compatibility Information

Red Hat Software Collections 2.0 is available for all supported releases of Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 on AMD64 and Intel 64 architectures.

1.5. Known Issues

`rh-mysql56`, `rh-mariadb100` components, BZ#1194611

The *rh-mysql56-mysql-server* and *rh-mariadb100-mariadb-server* packages no longer provide the **test** database by default. Although this database is not created during initialization, the grant tables are prefilled with the same values as when **test** was created by default. As a consequence, upon a later creation of the **test** or **test_*** databases, these databases have less restricted access rights than is default for new databases.

Additionally, when running benchmarks, the **run-all-tests** script no longer works out of the box with example parameters. You need to create a test database before running the tests and specify the database name in the **--database** parameter. If the parameter is not specified, **test** is taken by default but you need to make sure the **test** database exist.

httpd24 component, BZ#1224763

When using the **mod_proxy_fcgi** module with FastCGI Process Manager (PHP-FPM), **httpd** uses port **8000** for the FastCGI protocol by default instead of the correct port **9000**. To work around this problem, specify the correct port explicitly in configuration.

rh-passanger40 component, BZ#1196555

When Passenger from the *rh-passenger40* Software Collection is run as a module for **httpd**, the functionality is restricted by SELinux policy. To work around this problem, switch the passenger domain to permissive mode by running the following command as **root**:

```
semanage permissive -a passenger_t
```

Standalone server and **nginx** integration are not affected by this issue.

mongodb24 component

The *mongodb24* Software Collection from Red Hat Software Collections 1.2 cannot be rebuilt with the *rh-java-common* and *maven30* Software Collections shipped with Red Hat Software Collections 2.0. Additionally, the *mongodb24-build* and *mongodb24-scldevel* packages cannot be installed with Red Hat Software Collections 2.0 due to unsatisfied requires on the *maven30-javapackages-tools* and *maven30-maven-local packages*. When the *mongodb24-scldevel* package is installed, broken dependencies are reported and the **yum --skip-broken** command skips too many packages. Users are advised to update to the *rh-mongodb26* Software Collection.

perl component

When the user tries to use the **mod_perl** module from both the *rh-perl520* and *perl516* Software Collections, a conflict in the **/opt/rh/httpd24/root/usr/lib64/httpd/modules/mod_perl.so** file occurs. As a consequence, it is impossible to use **mod_perl** from more than one **Perl** Software Collection.

nodejs010 component

Shared libraries provided by the *nodejs010* Software Collection, namely **libcares**, **libhttp_parser**, and **libuv**, are not properly prefixed with the Collection name. As a consequence, conflicts with the corresponding system libraries might occur.

nodejs-hawk component

The *nodejs-hawk* package uses an implementation of the SHA-1 and SHA-256 algorithms adopted from the CryptoJS project. In this release, the client-side JavaScript is obfuscated. The future fix will involve using crypto features directly from the CryptoJS library.

postgresql component

The *rh-postgresql94* and *postgresql92* packages for Red Hat Enterprise Linux 6 do not provide the **sepgsql** module as this feature requires installation of *libselinux* version 2.0.99, which is not available in Red Hat Enterprise Linux 6.

httpd, mariadb, mongodb, mysql, nodejs, perl, php55, rh-php56, python, ruby, ror, thermostat, and v8314 components, BZ#1072319

When uninstalling the *httpd24*, *mariadb55*, *rh-mariadb100*, *mongodb24*, *rh-mongodb26*, *mysql55*, *rh-mysql56*, *nodejs010*, *perl516*, *rh-perl520*, *php55*, *rh-php56*, *python27*, *python33*, *rh-python34*, *ruby193*, *ruby200*, *rh-ruby22*, *ror40*, *rh-ror41*, *thermostat1*, or *v8314* packages, the order of uninstalling can be relevant due to ownership of dependent packages. As a consequence, some directories and files might not be removed properly and might remain on the system.

mariadb, mysql, postgresql, mongodb components

Red Hat Software Collections 2.0 contains the **MySQL 5.6**, **MariaDB 10.0**, **PostgreSQL 9.4** and **MongoDB 2.6** databases. The core Red Hat Enterprise Linux 6 provides earlier versions of the **MySQL** and **PostgreSQL** databases (client library and daemon). The core Red Hat Enterprise Linux 7 provides earlier versions of the **MariaDB** and **PostgreSQL** databases (client library and daemon). Client libraries are also used in database connectors for dynamic languages, libraries, and so on.

The client library packaged in the Red Hat Software Collections database packages in the **PostgreSQL** component is not supposed to be used, as it is included only for purposes of server utilities and the daemon. Users are instead expected to use the system library and the database connectors provided with the core system.

A protocol, which is used between the client library and the daemon, is stable across database versions, so, for example, using the **PostgreSQL 9.2** client library with the **PostgreSQL 9.4** daemon works as expected.

The core Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 do not include the client library for **MongoDB**. In order to use this client library for your application, you should use the client library from Red Hat Software Collections and always use the **scl enable ...** call every time you run an application linked against this **MongoDB** client library.

mariadb, mysql, mongodb components

MariaDB, MySQL, and MongoDB do not make use of the **/opt/*provider*/*collection*/root** prefix when creating log files. Note that log files are saved in the **/var/opt/*provider*/*collection*/log/** directory, not in **/opt/*provider*/*collection*/root/var/log/**.

httpd component

Compiling external applications against the Apache Portable Runtime (APR) and APR-util libraries from the *httpd24* Software Collection is not supported. The LD_LIBRARY_PATH is not set in *httpd24* because it is not required by any application in this Software Collection.

httpd, ruby193 components, BZ#1071145

In Red Hat Enterprise Linux 6.5 and earlier versions, **httpd** is unable to execute the binary files in the **mod_passenger** module, namely **PassengerWatchdog**, **PassengerHelperAgent**, **PassengerLoggingAgent**, and **SpawnPreparer** in the **/opt/rh/ruby193/root/usr/lib64/gems/exts/passenger-4.0.18/agents/** directory. To work around this problem, disable SELinux by running the following command as **root**:

```
setenforce 0
```

nginx component, BZ#1045041

In Red Hat Enterprise Linux 6.5 and earlier versions, no SELinux policy is applied for the **nginx** daemon.

python27 component

In Red Hat Enterprise Linux 7, when the user tries to install the *python27-python-debuginfo* package, the **/usr/src/debug/Python-2.7.5/Modules/socketmodule.c** file conflicts with the corresponding file from the *python-debuginfo* package installed on the core system. Consequently, installation of the *python27-python-debuginfo* fails. To work around this problem, uninstall the *python-debuginfo* package and then install the *python27-python-debuginfo* package.

devassist component

When the user tries to rebuild the *devassist09-PyYAML* package on Red Hat Enterprise Linux 6, the build fails due to a soft dependency, if the Pyrex or Cython programming languages are detected. To work around this problem, make sure the *pyrex* or *cython* packages are not installed on your system.

Other Notes

rh-ruby22, rh-python34, rh-php56 components

Using Software Collections on a read-only NFS has several limitations.

- Ruby gems cannot be installed while the *rh-ruby22* Software Collection is on a read-only NFS. Consequently, for example, when the user tries to install the ab gem using the **gem install ab** command, the following error message is displayed:

```
ERROR:  While executing gem ... (Errno::EROFS)
    Read-only file system @ dir_s_mkdir - /opt/rh/rh-
ruby22/root/usr/local/share/gems
```

  The same problem occurs when the user tries to update or install gems from an external source by running the **bundle update** or **bundle install** commands.

- When installing Python packages on a read-only NFS using the Python Package Index (PyPI), running the **pip** command fails with an error message similar to this:

```
Read-only file system: '/opt/rh/rh-
python34/root/usr/lib/python3.4/site-packages/ipython-
3.1.0.dist-info'
```

- Installing packages from PHP Extension and Application Repository (PEAR) on a read-only NFS using the **pear** command fails with the error message:

```
Cannot install, php_dir for channel "pear.php.net" is not
writeable by the current user
```

This is an expected behavior.

thermostat component

Previously, it was sufficient to start the **Thermostat** storage and agent back ends by running the **thermostat service** command. With this update, it is necessary to first run the **thermostat-setup** command and then configure the agent manually with credentials in the **agent.auth** file. For details, refer to the Thermostat User Guide.

thermostat component

The **thermostat1-thermostat-tomcat start** command, which starts the Thermostat web storage endpoint, can be used only on Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7.0. On Red Hat Enterprise Linux 7.1 and later versions, use **service tomcat@thermostat start** instead.

httpd component

Language modules for Apache are supported only with the Red Hat Software Collections version of **Apache httpd** and not with the Red Hat Enterprise Linux system versions of **httpd**. For example, the **mod_wsgi** module from the *rh-python34* Collection can be used only with the *httpd24* Collection.

all components

Since Red Hat Software Collections 2.0, configuration files, variable data, and runtime data of individual Collections are stored in different directories than in previous versions of Red Hat Software Collections.

coreutils component

Some utilities, for example, **su**, **login**, or **screen**, do not export environment settings in all cases, which can lead to unexpected results. It is therefore recommended to use **sudo** instead of **su** and set the **env_keep** environment variable in the **/etc/sudoers** file. Alternatively, you can run commands in a reverse order; for example:

```
su -l postgres -c "scl enable rh-postgresql94 psql"
```

instead of

```
scl enable rh-postgresql94 bash
su -l postgres -c psql
```

When using tools like **screen** or **login**, you can use the following command to preserve the environment settings:

```
source /opt/rh/<collection_name>/enable
```

php54 component

Note that **Alternative PHP Cache (APC)** in Red Hat Software Collections is provided for user data cache only. For opcode cache, **Zend OPcache** is provided.

python component

When the user tries to install more than one *scldevel* package from the *python27*, *python33*, and *rh-python34* Software Collections, a transaction check error message is returned. This is an expected behavior because the user can install only one set of the macro files provided by the packages (**`%scl_python`**, **`%scl_`*`prefix`*`_python`**).

php component

When the user tries to install more than one *scldevel* package from the *php54*, *php55*, and *rh-php56* Software Collections, a transaction check error message is returned. This is an expected behavior because the user can install only one set of the macro files provided by the packages (**`%scl_php`**, **`%scl_`*`prefix`*`_php`**).

ruby component

When the user tries to install more than one *scldevel* package from the *ruby193*, *ruby200*, and *rh-ruby22* Software Collections, a transaction check error message is returned. This is an expected behavior because the user can install only one set of the macro files provided by the packages (**`%scl_ruby`**, **`%scl_`*`prefix`*`_ruby`**).

perl component

When the user tries to install more than one *scldevel* package from the *perl516* and *rh-perl520* Software Collections, a transaction check error message is returned. This is an expected behavior because the user can install only one set of the macro files provided by the packages (**`%scl_perl`**, **`%scl_`*`prefix`*`_perl`**).

nodejs component

When installing the *nodejs010* Software Collection, *nodejs010* installs **GCC** in the base Red Hat Enterprise Linux system as a dependency, unless the *gcc* packages are already installed.

Chapter 2. Installation

This chapter describes in detail how to get access to the content set, install Red Hat Software Collections 2.0 on the system, and rebuild Red Hat Software Collections.

2.1. Getting Access to Red Hat Software Collections

The Red Hat Software Collections content set is available to customers with Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 subscriptions listed at https://access.redhat.com/solutions/472793. Depending on the subscription management service with which you registered your Red Hat Enterprise Linux system, you can either enable Red Hat Software Collections by using Red Hat Subscription Management, or by using RHN Classic. For detailed instructions on how to enable Red Hat Software Collections using RHN Classic or Red Hat Subscription Management, see the respective section below. For information on how to register your system with one of these subscription management services, see Using and Configuring Red Hat Subscription Manager.

2.1.1. Using Red Hat Subscription Management

If your system is registered with Red Hat Subscription Management, complete the following steps to attach the subscription that provides access to the repository for Red Hat Software Collections and enable the repository:

1. Display a list of all subscriptions that are available for your system and determine the pool ID of a subscription that provides Red Hat Software Collections. To do so, type the following at a shell prompt as **`root`**:

   ```
   subscription-manager list --available
   ```

   For each available subscription, this command displays its name, unique identifier, expiration date, and other details related to it. The pool ID is listed on a line beginning with **`Pool Id`**.

2. Attach the appropriate subscription to your system by running the following command as **`root`**:

   ```
   subscription-manager attach --pool=pool_id
   ```

   Replace *pool_id* with the pool ID you determined in the previous step. To verify the list of subscriptions your system has currently attached, type as **`root`**:

   ```
   subscription-manager list --consumed
   ```

3. Display the list of available Yum list repositories to retrieve repository metadata and determine the exact name of the Red Hat Software Collections repositories. As **`root`**, type:

   ```
   subscription-manager repos --list
   ```

   Or alternatively, run **`yum repolist all`** for a brief list.

   The repository names depend on the specific version of Red Hat Enterprise Linux you are using and are in the following format:

```
rhel-variant-rhscl-6-rpms
rhel-variant-rhscl-6-debug-rpms
rhel-variant-rhscl-6-source-rpms

rhel-server-rhscl-6-eus-rpms
rhel-server-rhscl-6-eus-source-rpms
rhel-server-rhscl-6-eus-debug-rpms

rhel-variant-rhscl-7-rpms
rhel-variant-rhscl-7-debug-rpms
rhel-variant-rhscl-7-source-rpms

rhel-server-rhscl-7-eus-rpms
rhel-server-rhscl-7-eus-source-rpms
rhel-server-rhscl-7-eus-debug-rpms
```

Replace *variant* with the Red Hat Enterprise Linux system variant, that is, **`server`** or **`workstation`**. Note that Red Hat Software Collections is supported neither on the **`Client`** nor on the **`ComputeNode`** variant.

4. Enable the appropriate repository by running the following command as **`root`**:

```
subscription-manager repos --enable repository
```

Once the subscription is attached to the system, you can install Red Hat Software Collections as described in Section 2.2, “Installing Red Hat Software Collections”. For more information on how to register your system using Red Hat Subscription Management and associate it with subscriptions, see Using and Configuring Red Hat Subscription Manager.

2.1.2. Using RHN Classic

If your system is registered with RHN Classic, complete the following steps to subscribe to Red Hat Software Collections:

1. Display a list of all channels that are available to you and determine the exact name of the Red Hat Software Collections channel. To do so, type the following at a shell prompt as **`root`**:

```
rhn-channel --available-channels
```

The name of the channel depends on the specific version of Red Hat Enterprise Linux you are using and is in the following format, where *variant* is the Red Hat Enterprise Linux system variant (**`server`** or **`workstation`**):

```
rhel-x86_64-variant-6-rhscl-1

rhel-x86_64-server-6.5.z-rhscl-1
rhel-x86_64-server-6.6.z-rhscl-1

rhel-x86_64-variant-7-rhscl-1

rhel-x86_64-server-7.1.eus-rhscl-1
```

Red Hat Enterprise Linux 7 channels are accessible only through Red Hat Satellite instances.

Note

Red Hat Software Collections 2.x are distributed in the same channels as Red Hat Software Collections 1.x.

2. Subscribe the system to the Red Hat Software Collections channel by running the following command as **root**:

```
rhn-channel --add --channel=channel_name
```

   Replace *channel_name* with the name you determined in the previous step.

3. Verify the list of channels you are subscribed to. As **root**, type:

```
rhn-channel --list
```

When the system is subscribed, you can install Red Hat Software Collections as described in Section 2.2, “Installing Red Hat Software Collections”. For more information on how to register your system with RHN Classic, see Using and Configuring Red Hat Subscription Manager.

2.1.3. Packages from the Optional Channel

Some of the Red Hat Software Collections 2.0 packages require the **Optional** channel to be enabled in order to complete the full installation of these packages. For detailed instructions on how to subscribe your system to this channel, see the relevant Knowledgebase articles on Red Hat Customer Portal: https://access.redhat.com/solutions/392003 for Red Hat Subscription Management or https://access.redhat.com/solutions/70019 if your system is registered with RHN Classic.

Packages from Software Collections for Red Hat Enterprise Linux 6 that require the **Optional** channel to be enabled are listed in the following table.

Table 2.1. Packages Requiring Enabling of the Optional Channel in Red Hat Enterprise Linux 6

Package from a Software Collection	Required Package from the Optional Channel
devtoolset-3-dyninst-testsuite	glibc-static
git19-git-cvs	cvsps
git19-perl-Git-SVN	perl-YAML, subversion-perl
mariadb55-mariadb-bench	perl-GD
mysql55-mysql-bench	perl-GD
php54-php-imap	libc-client
php54-php-recode	recode
php54-php-imap	libc-client
php54-php-recode	recode
php55-php-imap	libc-client
php55-php-recode	recode
rh-mariadb100-mariadb-bench	perl-GD
rh-mysql56-mysql-bench	perl-GD
rh-php56-php-imap	libc-client
rh-php56-php-recode	recode

Software Collections packages that require the **`Optional`** channel in Red Hat Enterprise Linux 7 are listed in the table below.

Table 2.2. Packages Requiring Enabling of the Optional Channel in Red Hat Enterprise Linux 7

Package from a Software Collection	Required Package from the Optional Channel
devassist09-devassistant	python-jinja2
devtoolset-3-build	scl-utils-build
devtoolset-3-dyninst-testsuite	glibc-static
devtoolset-3-easymock	cglib, objectweb-asm
devtoolset-3-eclipse-platform	sac
devtoolset-3-gcc-plugin-devel	libmpc-devel
devtoolset-3-icu4j-javadoc	java-1.7.0-openjdk-javadoc
devtoolset-3-jsch	jzlib
devtoolset-3-lucene-replicator	jetty-continuation, jetty-http, jetty-io, jetty-jmx, jetty-security, jetty-server, jetty-servlet, jetty-util
devtoolset-3-lucene-solr-grandparent	aether-api, aether-connector-wagon, aether-impl, aether-spi, aether-util, aopalliance, apache-commons-compress, apache-commons-net, apache-parent, apache-resource-bundles, aqute-bndlib, bsf, bsh, buildnumber-maven-plugin, cdi-api, cglib, felix-bundlerepository, felix-framework, felix-osgi-compendium, felix-osgi-core, felix-osgi-foundation, felix-osgi-obr, felix-shell, felix-utils, google-guice, guava, jboss-ejb-3.1-api, jboss-jaxrpc-1.1-api, jboss-servlet-3.0-api, jsch, jsoup, jzlib, kxml, maven, maven-archiver, maven-artifact, maven-artifact-manager, maven-artifact-resolver, maven-dependency-tree, maven-enforcer-api, maven-enforcer-plugin, maven-enforcer-rules, maven-file-management, maven-filtering, maven-model, maven-monitor, maven-plugin-bundle, maven-plugin-registry, maven-profile, maven-project, maven-scm, maven-settings, maven-shared-io, maven-wagon, modello, nekohtml, objectweb-asm, plexus-archiver, plexus-build-api, plexus-cipher, plexus-classworlds, plexus-compiler, plexus-i18n, plexus-interpolation, plexus-io, plexus-resources, plexus-sec-dispatcher, plexus-utils, plexus-velocity, sisu-inject-bean, sisu-inject-plexus, woodstox-core, xbean, xz-java
devtoolset-3-mockito	cglib, objectweb-asm
devtoolset-3-tika-parsers-epub	apache-commons-compress, xz-java
git19-git-cvs	cvsps
git19-perl-Git-SVN	subversion-perl
httpd24-mod_ldap	apr-util-ldap
php54-php-pspell	aspell
php55-php-pspell	aspell
python27-python-debug	scl-utils-build, tix
python27-python-devel	scl-utils-build
python27-tkinter	tix
rh-perl520-perl-Pod-Perldoc	groff
rh-php56-php-pspell	aspell
rh-python34-python-devel	scl-utils-build
rh-python34-python-sphinx	texlive-threeparttable, texlive-wrapfig

Note that packages from the **Optional** channel are not supported. For details, see the Knowledgebase article https://access.redhat.com/articles/1150793.

2.2. Installing Red Hat Software Collections

Red Hat Software Collections is distributed as a collection of RPM packages that can be installed, updated, and uninstalled by using the standard package management tools included in Red Hat Enterprise Linux. Note that a valid subscription is required to install Red Hat Software Collections on your system. For detailed instructions on how to associate your system with an appropriate subscription and get access to Red Hat Software Collections, see Section 2.1, “Getting Access to Red Hat Software Collections”.

Use of Red Hat Software Collections 2.0 requires the removal of any earlier pre-release versions, including Beta releases. If you have installed any previous version of Red Hat Software Collections 2.0, uninstall it from your system and install the new version as described in the Section 2.3, “Uninstalling Red Hat Software Collections” and Section 2.2.1, “Installing Individual Software Collections” sections.

The in-place upgrade from Red Hat Enterprise Linux 6 to Red Hat Enterprise Linux 7 is not supported by Red Hat Software Collections. As a consequence, the installed Software Collections might not work correctly after the upgrade. If you want to upgrade from Red Hat Enterprise Linux 6 to Red Hat Enterprise Linux 7, it is strongly recommended to remove all Red Hat Software Collections packages, perform the in-place upgrade, update the Red Hat Software Collections repository, and install the Software Collections packages again. It is advisable to back up all data before upgrading.

2.2.1. Installing Individual Software Collections

To install any of the Software Collections that are listed in Table 1.1, “Red Hat Software Collections 2.0 Components”, install the corresponding meta package by typing the following at a shell prompt as **root**:

```
yum install software_collection...
```

Replace *software_collection* with a space-separated list of Software Collections you want to install. For example, to install *php54* and *rh-mariadb100*, type as **root**:

```
~]# yum install php54 rh-mariadb100
```

This installs the main meta package for the selected Software Collection and a set of required packages as its dependencies. For information on how to install additional packages such as additional modules, see Section 2.2.2, “Installing Optional Packages”.

2.2.2. Installing Optional Packages

Each component of Red Hat Software Collections is distributed with a number of optional packages that are not installed by default. To list all packages that are part of a certain Software Collection but are not installed on your system, type the following at a shell prompt:

```
yum list available software_collection-\*
```

To install any of these optional packages, type as **root**:

```
yum install package_name...
```

Replace *package_name* with a space-separated list of packages that you want to install. For example, to install the *rh-perl520-perl-CPAN* and *rh-perl520-perl-Archive-Tar*, type:

```
~]# yum install rh-perl520-perl-CPAN rh-perl520-perl-Archive-Tar
```

2.2.3. Installing Debugging Information

To install debugging information for any of the Red Hat Software Collections packages, make sure that the *yum-utils* package is installed and type the following command as **root**:

```
debuginfo-install package_name
```

For example, to install debugging information for the *rh-ruby22-ruby* package, type:

```
~]# debuginfo-install rh-ruby22-ruby
```

Note that in order to use this command, you need to have access to the repository with these packages. If your system is registered with Red Hat Subscription Management, enable the **rhel-*variant*-rhscl-6-debug-rpms** or **rhel-*variant*-rhscl-7-debug-rpms** repository as described in Section 2.1.1, "Using Red Hat Subscription Management". If your system is registered with RHN Classic, subscribe the system to the **rhel-x86_64-*variant*-6-rhscl-1-debuginfo** or **rhel-x86_64-*variant*-7-rhscl-1-debuginfo** channel as described in Section 2.1.2, "Using RHN Classic". For more information on how to get access to debuginfo packages, see https://access.redhat.com/solutions/9907.

2.3. Uninstalling Red Hat Software Collections

To uninstall any of the Software Collections components, type the following at a shell prompt as **root**:

```
yum remove software_collection\*
```

Replace *software_collection* with the Software Collection component you want to uninstall.

Note that uninstallation of the packages provided by Red Hat Software Collections does not affect the Red Hat Enterprise Linux system versions of these tools.

2.4. Rebuilding Red Hat Software Collections

<collection>-build packages are not provided by default. If you wish to rebuild a collection and do not want or cannot use the **rpmbuild --define 'scl foo'** command, you first need to rebuild the metapackage, which provides the *<collection>-build* package.

Note that existing collections should not be rebuilt with different content. To add new packages into an existing collection, you need to create a new collection containing the new packages and make it dependent on packages from the original collection. The original collection has to be used without changes.

For detailed information on building Software Collections, refer to the Red Hat Software Collections Packaging Guide.

Chapter 3. Usage

This chapter describes the necessary steps for rebuilding and using Red Hat Software Collections 2.0, and deploying applications that use Red Hat Software Collections.

3.1. Using Red Hat Software Collections

3.1.1. Running an Executable from a Software Collection

To run an executable from a particular Software Collection, type the following command at a shell prompt:

```
scl enable software_collection... 'command...'
```

Or, alternatively, use the following command:

```
scl enable software_collection... -- command...
```

Replace *software_collection* with a space-separated list of Software Collections you want to use and *command* with the command you want to run. For example, to execute a Perl program stored in a file named **hello.pl** with the Perl interpreter from the *perl516* Software Collection, type:

```
~]$ scl enable perl516 'perl hello.pl'
Hello, World!
```

You can execute any command using the **scl** utility, causing it to be run with the executables from a selected Software Collection in preference to their possible Red Hat Enterprise Linux system equivalents. For a complete list of Software Collections that are distributed with Red Hat Software Collections, see Table 1.1, “Red Hat Software Collections 2.0 Components”.

3.1.2. Running a Shell Session with a Software Collection as Default

To start a new shell session with executables from a selected Software Collection in preference to their Red Hat Enterprise Linux equivalents, type the following at a shell prompt:

```
scl enable software_collection... bash
```

Replace *software_collection* with a space-separated list of Software Collections you want to use. For example, to start a new shell session with the *python27* and *postgresql92* Software Collections as default, type:

```
~]$ scl enable python27 postgresql92 bash
```

The list of Software Collections that are enabled in the current session is stored in the **$X_SCLS** environment variable, for instance:

```
~]$ echo $X_SCLS
python27 postgresql92
```

For a complete list of Software Collections that are distributed with Red Hat Software Collections, see Table 1.1, “Red Hat Software Collections 2.0 Components”.

3.1.3. Running a System Service from a Software Collection

Software Collections that include system services install corresponding init scripts in the **/etc/rc.d/init.d/** directory. To start such a service in the current session, type the following at a shell prompt as **root**:

```
service software_collection-service_name start
```

Replace *software_collection* with the name of the Software Collection and *service_name* with the name of the service you want to start. To configure this service to start automatically at boot time, type the following command as **root**:

```
chkconfig software_collection-service_name on
```

For example, to start the **postgresql** service from the *postgresql92* Software Collection and enable it in runlevels 2, 3, 4, and 5, type as **root**:

```
~]# service postgresql92-postgresql start
Starting postgresql92-postgresql service:                  [  OK  ]
~]# chkconfig postgresql92-postgresql on
```

For more information on how to manage system services in Red Hat Enterprise Linux 6, refer to the Red Hat Enterprise Linux 6 Deployment Guide. For a complete list of Software Collections that are distributed with Red Hat Software Collections, see Table 1.1, “Red Hat Software Collections 2.0 Components”.

3.2. Accessing a Manual Page from a Software Collection

Every Software Collection contains a general manual page that describes the content of this component. Each manual page has the same name as the component and it is located in the**/opt/rh** directory.

To read a manual page for a Software Collection, type the following command:

```
scl enable software_collection 'man software_collection'
```

Replace *software_collection* with the particular Red Hat Software Collections component. For example, to display the manual page for *mariadb55*, type:

```
~]$ scl enable mariadb55 "man mariadb55"
```

3.3. Deploying Applications That Use Red Hat Software Collections

In general, you can use one of the following two approaches to deploy an application that depends on a component from Red Hat Software Collections in production:

- Install all required Software Collections and packages manually and then deploy your application, or
- Create a new Software Collection for your application and specify all required Software Collections and other packages as dependencies.

For more information on how to manually install individual Red Hat Software Collections components, see Section 2.2, “Installing Red Hat Software Collections”. For further details on how to use Red Hat Software Collections, see Section 3.1, “Using Red Hat Software Collections”. For a detailed explanation of how to create a custom Software Collection or extend an existing one, read the Red Hat Software Collections Packaging Guide.

3.4. Dockerfiles for Red Hat Software Collections

Red Hat Software Collections 2.0 is shipped with Dockerfiles for the following Software Collections:

- *httpd24*
- *mariadb55*
- *mongodb24*
- *mysql55*
- *nginx16*
- *nodejs010*
- *perl516*
- *php54*
- *php55*
- *postgresql92*
- *python27*
- *python33*
- *rh-mariadb100*
- *rh-mongodb26*
- *rh-mysql56*
- *rh-passenger40*
- *rh-perl520*
- *rh-php56*
- *rh-postgresql94*
- *rh-python34*
- *rh-ror41*
- *rh-ruby22*
- *ror40*
- *ruby193*
- *ruby200*

The Dockerfiles are included in the *rhscl-dockerfiles* package distributed with Red Hat Software Collections. Dockerfiles are text files that define how a Docker image is created.

Note

The *docker* package, which contains the **`Docker`** daemon, command line tool, and other necessary components for building and using docker-formatted container images, is currently only available for the Server variant of the Red Hat Enterprise Linux 7 product. Red Hat Software Collections Dockerfiles are distributed for Red Hat Enterprise Linux 6 as well, but the images built using them can only be deployed on Red Hat Enterprise Linux 7 Server.

Each Dockerfile creates a minimal Docker image from Red Hat Enterprise Linux 6 or Red Hat Enterprise Linux 7 plus the Software Collection. Each Dockerfile will create an image which:

- Installs the basic set of packages from each Software Collection,
- Exposes some TCP ports; for example, port **`80`** and **`443`** for the *httpd24* collection.

The Dockerfiles are provided as examples, using which customers can build more complex containers.

Dockerfiles are available also for previously released Software Collections. For detailed information about them, refer to the Red Hat Software Collections documentation and the Red Hat Software Collections Product Life Cycle document.

3.4.1. Installation and Usage

To install the *rhscl-dockerfiles* package, type the following command as **`root`**:

```
yum install rhscl-dockerfiles
```

Use these Dockerfiles to create Docker images for the covered Software Collections.

For more information about building an image from a Dockerfile, see the Get Started with Docker Formatted Container Images on Red Hat Systems Knowledgebase article, or particularly the Building an Image from a Dockerfile section.

3.4.2. Deploying Software Collections Dependent on the Red Hat Software Collections Docker Images

You can use a Red Hat Software Collections Docker image as a base image and create your own containerized Software Collection on top of it as a separate image.

For more information about creating a new Docker image, see the Creating Docker Images section in the relevant Knowledgebase article.

Chapter 4. Specifics of Individual Software Collections

This chapter is focused on the specifics of certain Software Collections and provides additional details concerning these components.

4.1. Red Hat Developer Toolset

Red Hat Developer Toolset is designed for developers working on the Red Hat Enterprise Linux platform. Red Hat Developer Toolset provides current versions of the **GNU Compiler Collection**, **GNU Debugger**, **Eclipse** development platform, and other development, debugging, and performance monitoring tools. Similarly to other Software Collections, an additional set of tools is installed into the **/opt/** directory. These tools are enabled by the user on demand using the supplied **scl** utility. Similarly to other Software Collections, these do not replace the Red Hat Enterprise Linux system versions of these tools, nor will they be used in preference to those system versions unless explicitly invoked using the **scl** utility.

For a list of features, refer to the Main Features section of the *Red Hat Developer Toolset Release Notes*.

For a complete list of components, see the Red Hat Developer Toolset Components table in the *Red Hat Developer Toolset User Guide*.

Note that since Red Hat Software Collections 2.0, Red Hat Developer Toolset requires the *rh-java-common* Software Collection.

4.2. Thermostat 1

The **Thermostat** Software Collection provides a monitoring and instrumentation tool for the OpenJDK HotSpot JVM, with support for monitoring multiple JVM instances. The system is made up of two components: an **Agent**, which collects data, and a **Client**, which allows users to visualize collected data. These components communicate via a storage layer: either directly via **MongoDB** or indirectly via a Web layer for increased security. A pluggable agent and GUI framework allows for collection and visualization of performance data beyond what is included out of the box.

To install the *thermostat1* collection, type the following command as **root**:

```
yum install thermostat1
```

Note that since Red Hat Software Collections 2.0, the *thermostat1* Software Collection requires the *rh-java-common* Collection.

To enable the *thermostat1* collection, type the following command at a shell prompt:

```
scl enable thermostat1 bash
```

For more information, please refer to the Thermostat User Guide. In order to deploy Thermostat securely, see the Configuration and Administration Guide.

4.3. Ruby on Rails 4.1

This Software Collection adds the *rh-ruby22* package together with the *rh-ror41* package. The **Ruby on Rails** Collection can be enabled by the following command, which will automatically enable *rh-ruby22*:

```
scl enable rh-ror41 bash
```

These two collections are supported together.

4.4. MongoDB 2.6

To install the *rh-mongodb26* collection, type the following command as **root**:

```
yum install rh-mongodb26
```

Note that since Red Hat Software Collections 2.0, the *rh-mongodb26* Software Collection requires the *rh-java-common* Collection.

To run the **MongoDB** shell utility, type the following command:

```
scl enable rh-mongodb26 'mongo'
```

4.4.1. MongoDB 2.6 on Red Hat Enterprise Linux 6

If you are using Red Hat Enterprise Linux 6, the following instructions apply to your system.

To start the **MongoDB** daemon, type the following command as **root**:

```
service rh-mongodb26-mongod start
```

To start the **MongoDB** daemon on boot, type this command as **root**:

```
chkconfig rh-mongodb26-mongod on
```

To start the **MongoDB** sharding server, type this command as **root**:

```
service rh-mongodb26-mongos start
```

To start the **MongoDB** sharding server on boot, type the following command as **root**:

```
chkconfig rh-mongodb26-mongos on
```

Note that the **MongoDB** sharding server does not work unless the user starts at least one configuration server and specifies it in the **mongos.conf** file.

4.4.2. MongoDB 2.6 on Red Hat Enterprise Linux 7

When using Red Hat Enterprise Linux 7, the following commands are applicable.

To start the **MongoDB** daemon, type the following command as **root**:

```
systemctl start rh-mongodb26-mongod.service
```

To start the **MongoDB** daemon on boot, type this command as **root**:

```
systemctl enable rh-mongodb26-mongod.service
```

To start the **MongoDB** sharding server, type the following command as **root**:

```
systemctl start rh-mongodb26-mongos.service
```

To start the **MongoDB** sharding server on boot, type this command as **root**:

```
systemctl enable rh-mongodb26-mongos.service
```

Note that the **MongoDB** sharding server does not work unless the user starts at least one configuration server and specifies it in the **mongos.conf** file.

4.5. DevAssistant

DevAssistant is a tool designed to assist developers with creating and setting up basic projects in various programming languages, installing dependencies, setting up a development environment, and working with source control. The *devassist09* Software Collection supports several programming languages, namely C, C++, Java, and Python. Additionally, DevAssistant is able to support working with any other language, framework, or tool due to its modular architecture.

DevAssistant is a framework that runs plug-ins called *assistants*. Each assistant can have several subassistants.

4.5.1. Getting Started with DevAssistant

To install the *devassist09* Software Collection, type the following command as **root**:

```
yum install devassist09
```

To enable this collection, type the following command at a shell prompt:

```
scl enable devassist09 bash
```

To get help for DevAssistant, use the following command:

```
devassistant --help
```

or the shorter variant of the same command:

```
da -h
```

It is advisable to use the **--help** option on each level to list your possible next steps, until you reach the level of an executable subassistant (see Example 4.1, “Creating a New Python Library Project”).

To access the graphical user interface, type this command at a shell prompt:

```
devassistant-gui
```

or the shortened variant:

```
da-gui
```

Please note that the GUI is available only if you install the *devassist09* Software Collection on Red Hat Enterprise Linux 7. The functionalities and procedures are the same as when using the command line interface.

Note that the **devassistant** and **da** commands are equal. Further in the text, we will use only the shorter variant, the **da** command.

4.5.2. Running Assistants

DevAssistant provides the following functionalities: **create**, **modify**, **prepare**, and **task**. To run an assistant, use the following command:

```
da [--debug] {create,modify,prepare,task} [assistant [arguments]] ...
```

The four basic commands and descriptions related to these functionalities are listed in the following table:

Table 4.1. Functionalities of DevAssistant

Command	Shortened Command	Description
da create	**da crt**	Creating a new project from scratch
da modify	**da mod**	Working with an existing project
da prepare	**da prep**	Preparing a development environment for an upstream project
da task		Performing a custom task not related to a specific project

The *devassist09* Software Collection does not include any assistants for the **modify**, **prepare**, and **task** functionalities. These categories are available for users who want to create their own assistants.

4.5.3. Creating Projects with DevAssistant

The *devassist09* Software Collection includes the following assistants for creating projects:

Table 4.2. Assistants for Creating Projects

Assistant	Subassistant	Description
c	app	An application in C
	lib	A dynamically linked library in C
cpp	app	An application in C++
	lib	A dynamically linked library in C++
java	maven	A simple project using Maven
python	lib	A simple library for Python

The following example demonstrates creating a new Python library project by following instructions displayed by the **--help** option.

Example 4.1. Creating a New Python Library Project

To create a new Python library project, complete the following steps:

1. Enable the *devassist09* Software Collection by running this command:

```
~]$ scl enable devassist09 bash
```

2. Display help about DevAssistant by using the **--help** option:

```
~]$ da --help
You can either run assistants with:
da [--debug] {create,modify,prepare,task} [ASSISTANT [ARGUMENTS]]
...

Where:
create   used for creating new projects
modify   used for working with existing projects
prepare  used for preparing environment for upstream projects
task     used for performing custom tasks not related to a
specific project
You can shorten "create" to "crt", "modify" to "mod" and
"prepare" to "prep".

Or you can run a custom action:
da [--debug] [ACTION] [ARGUMENTS]

Available actions:
help     Print detailed help
version  Print version
```

3. List the possible next steps for creating a project by typing:

```
~]$ da create --help
usage:  create [-h] [--deps-only] {c,cpp,java,python} ...

Kickstart new projects easily with DevAssistant.

optional arguments:
  -h, --help           show this help message and exit
  --deps-only          Only install dependencies

subassistants:
  Following subassistants will help you with setting up your
project.

  {c,cpp,java,python}
```

4. Display help on the **python** assistant by typing at a shell prompt:

```
~]$ da create python --help
usage: create python [-h] {lib} ...

This is a base Python assistant, you have to select a
subassistant.

optional arguments:
  -h, --help  show this help message and exit
```

```
subassistants:
  Following subassistants will help you with setting up your
project.

  {lib}
```

5. List your choices for the only **python** subassistant, **lib**, by running this command:

```
~]$ da create python lib --help
usage: create python lib [-h] [-e [ECLIPSE]] -n NAME

Scaffolds a simple Python library project.

optional arguments:
  -h, --help            show this help message and exit
  -e [ECLIPSE], --eclipse [ECLIPSE]
                        Configure as Eclipse project (uses
~/workspace or
                        specified directory)
  -n NAME, --name NAME  Name of project to create
```

6. Run the assistant to create your new Python library project named **mypythonlib** by using the following command:

```
~]$ da create python lib -n mypythonlib
```

To get more information about the upstream version of **DevAssistant**, refer to the DevAssistant User Documentation. Please note that though the basic concept of the upstream application is the same as in the *devassist09* Software Collection, individual plug-ins and their functionalities might differ.

4.5.4. Backward Compatibility in DevAssistant

The updated version of **DevAssistant** can cause incompatibility in assistants that have not been provided by the *devassist09-devassistant-assistants-dts* package, that is, in your own assistants.

- Since **DevAssistant 0.9.3**, the variable names in the assistant files are no longer derived from the argument flags but from the argument names. In the following example, the **$foo** variable is initialized instead of the **$bar** variable:

```
args:
   foo:
     ...
     flags: [-b, --bar]
     ...
```

- Unknown attributes in the arguments section in the assistant file are no longer allowed. Since **DevAssistant 0.9.3**, an error message is returned in the following example because the **unknown_attribute** is not known to the parser:

```
args:
   foo:
     ...
     unknown_attribute: foo bar baz
```

```
      ...
```

4.6. Maven

The *maven30* Software Collection provides a software project management and comprehension tool. Based on the concept of a project object model (POM), **Maven** can manage a project's build, reporting, and documentation from a central piece of information.

To install the *maven30* Collection, type the following command as **root**:

```
yum install maven30
```

Note that since Red Hat Software Collections 2.0, the *maven30* Software Collection requires the *rh-java-common* Collection.

To enable this collection, type the following command at a shell prompt:

```
scl enable maven30 bash
```

Global Maven settings, such as remote repositories or mirrors, can be customized by editing the **/opt/rh/maven30/root/etc/maven/settings.xml** file.

For more information about using Maven, refer to the Maven documentation. To find documentation regarding individual plug-ins, please see the index of plug-ins.

4.7. Passenger

The *rh-passenger40* Software Collection provides **Phusion Passenger**, a web and application server designed to be fast, robust and lightweight.

The *rh-passenger40* Collection supports multiple versions of **Ruby**, particularly the *ruby193*, *ruby200*, and *rh-ruby22* Software Collections together with **Ruby on Rails** using the *ror40* or *rh-ror41* Collections. Prior to using **Passenger** with any of the **Ruby** Software Collections, install the corresponding package from the *rh-passenger40* Collection: the *rh-passenger-ruby193*, *rh-passenger-ruby200*, or *rh-passenger-ruby22* package.

The *rh-passenger40* Software Collection can also be used with **Apache httpd** from the *httpd24* Software Collection. To do so, install the *rh-passenger40-mod_passenger* package. Refer to the default configuration file **/opt/rh/httpd24/root/etc/httpd/conf.d/passenger.conf** for an example of **Apache httpd** configuration, which shows how to use multiple **Ruby** versions in a single **Apache httpd** instance.

Additionally, the *rh-passenger40* Software Collection can be used with the **nginx 1.6** web server from the *nginx16* Software Collection. To use **nginx 1.6** with *rh-passenger40*, you can run **Passenger** in Standalone mode using the following command in the web appplication's directory:

```
scl enable nginx16 rh-passenger40 'passenger start'
```

Alternatively, edit the *nginx16* configuration files as described in the upstream Passenger documentation.

Chapter 5. Migration

This chapter provides information on migrating to versions of components included in Red Hat Software Collections 2.0.

5.1. Migrating to MariaDB 10.0

Red Hat Enterprise Linux 6 contains **MySQL 5.1** as the default **MySQL** implementation. Red Hat Enterprise Linux 7 includes **MariaDB 5.5** as the default **MySQL** implementation. **MariaDB** is a community-developed drop-in replacement for **MySQL**. In addition to these basic versions, **MariaDB 5.5** has been available for Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 as a Software Collection since Red Hat Software Collections 1.0.

The *rh-mariadb100* Software Collection available for both Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 does not conflict with the *mysql* or *mariadb* packages from the core systems, so it is possible to install the *rh-mariadb100* Software Collection together with the *mysql* or *mariadb* packages. It is also possible to run both versions at the same time, however, the port number and the socket in the **`my.cnf`** files need to be changed to prevent these specific resources from conflicting.

Note that it is possible to upgrade to **MariaDB 10.0** only from **MariaDB 5.5** or **MySQL 5.5**. If you need to upgrade from an earlier version, upgrade to **MariaDB 5.5** or **MySQL 5.5** first. Instructions how to upgrade to **MariaDB 5.5** or **MySQL 5.5** are available in the Red Hat Software Collections 1.2 Release Notes.

5.1.1. Notable Differences Between the *mariadb55* and *rh-mariadb100* Software Collections

MariaDB 10.0 is built on the **MariaDB 5.5** series with backported features from **MySQL 5.6** and with entirely new features unavailable elsewhere. The *rh-mariadb100* Software Collection introduces the following notable changes:

- The service has been renamed to **`rh-mariadb100-mariadb`** in both Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7
- The **`test`** database is no longer created by default
- Configuration files for the *rh-mariadb100* Software Collection are the **`/etc/opt/rh/rh-mariadb100/my.cnf`** file and in the **`/etc/opt/rh/rh-mariadb100/my.cnf.d/`** directory
- Variable files including the database files for the *rh-mariadb100* Software Collection are located in the **`/var/opt/rh/rh-mariadb100/lib/`** directory
- The log file for the MariaDB daemon is **`/var/opt/rh/rh-mariadb100/log/mariadb/mariadb.log`**
- The pid file for the daemon is **`/var/run/rh-mariadb100-mariadb/mariadb.pid`**

Note that the *rh-mariadb100* Software Collection supports neither mounting over NFS nor dynamical registering using the **`scl register`** command.

For detailed changes, refer to the MariaDB documentation.

If you are going to upgrade from **MySQL**, refer to the articles about compatibility and features differences.

5.1.2. Upgrading to the *rh-mariadb100* Software Collection

Important

Prior to upgrading, back-up all your data, including any MariaDB or MySQL databases.

Upgrading can be performed either by using the **mysqldump** and **mysqlimport** utilities or using an in-place upgrade.

- In the former scenario, the whole dump of all databases from one database is generated and **mysql** is run with the dump file as an input using the **mysqlimport** or **LOAD DATA INFILE SQL** command within the other database. At the same time, the appropriate daemons have to be running during both dumping and restoring. You can use the **--all-databases** option in the **mysqldump** call to include all databases in the dump. The **--routines**, **--triggers**, and **--events** options can also be used if needed.

- During the in-place upgrade, the data files are copied from one database directory to another database directory. The daemons must not be running at the time of copying. Set appropriate permissions and SELinux context for the copied files.

After upgrading, start the server and run the **mysql_upgrade** command. Running **mysql_upgrade** is necessary to check and repair internal tables.

In case the **root** user has a non-empty password defined (it should have a password defined), it is necessary to call the **mysql_upgrade** utility with the **-p** option and specify the password.

Service names and paths bellow depend on which version you are upgrading from.

Example 5.1. Dump and Restore Upgrade

1. Create a backup from **MariaDB**.

   - If you are upgrading from **MariaDB 5.5** from base Red Hat Enterprise Linux 7:

   ```
   ~]# service mariadb start
   Starting mariadb:                                          [
   OK  ]
   ~]# mysqldump --all-databases --routines --events > dump.sql
   ~]# service mariadb stop
   Stopping mariadb:                                          [
   OK  ]
   ```

   - If you are upgrading from the *mariadb55* Software Collection in Red Hat Enterprise Linux 6:

   ```
   ~]# service mariadb55-mysqld start
   Starting mariadb55-mysqld:                                 [
   OK  ]
   ~]# scl enable mariadb55 -- mysqldump --all-databases --
   routines --events > dump.sql
   ~]# service mariadb55-mysqld stop
   Stopping mariadb55-mysqld:                                 [
   OK  ]
   ```

   - For upgrading from the *mariadb55* Software Collection in Red Hat Enterprise Linux 7, use **mariadb55-mariadb** as the service name.

- For upgrading from the *mysql55* Software Collection, use **`mysql55-mysqld`** as the service name.

2. Import the dumped database into the *rh-mariadb100* Software Collection:

```
~]# service rh-mariadb100-mariadb start
Starting rh-mariadb100-mariadb:                              [  OK
]
~]# scl enable rh-mariadb100 'mysql' < dump.sql
~]# scl enable rh-mariadb100 'mysql_upgrade -u root -p'
Enter password:
Looking for 'mysql' as: mysql
Looking for 'mysqlcheck' as: mysqlcheck
Running 'mysqlcheck with default connection arguments
Running 'mysqlcheck with default connection arguments
a.t1                                                 OK
mysql.columns_priv                                   OK
<skipped tables list>
mysql.user                                           OK
Running 'mysql_fix_privilege_tables'...
OK
```

Example 5.2. In-place Upgrade from MariaDB 5.5

If you are upgrading from **MariaDB 5.5** from base Red Hat Enterprise Linux 7, perform the upgrade as shown in the following example:

```
~]# service mariadb stop
Stopping mariadb:                                            [  OK  ]
~]# service rh-mariadb100-mariadb stop
Stopping rh-mariadb100-mariadb:                              [  OK  ]
~]# rm -rf /var/opt/rh/rh-mariadb100/lib/mysql/
~]# cp -r /var/lib/mysql/ /var/opt/rh/rh-mariadb100/lib/mysql/
~]# chown -R mysql:mysql /var/opt/rh/rh-mariadb100/lib/mysql/
~]# restorecon -R /var/opt/rh/rh-mariadb100/lib/mysql/
~]# service rh-mariadb100-mariadb start
Starting rh-mariadb100-mariadb:                              [  OK  ]
~]# scl enable rh-mariadb100 'mysql_upgrade -u root -p'
Enter password:
Looking for 'mysql' as: mysql
Looking for 'mysqlcheck' as: mysqlcheck
Running 'mysqlcheck with default connection arguments
Running 'mysqlcheck with default connection arguments
a.t1                                                 OK
mysql.columns_priv                                   OK
<skipped tables list>
mysql.user                                           OK
Running 'mysql_fix_privilege_tables'...
OK
```

For upgrading from the *mariadb55* Software Collection{;, use the **`/opt/rh/mariadb55/root/var/lib/mysql/`** as a source when copying the data.

For upgrading from the *mysql55* Software Collection, use the **`/opt/rh/mysql55/root/var/lib/mysql/`** as a source when copying the data.

For further details, refer to the articles about upgrading from MariaDB 5.5 or upgrading from MySQL 5.5.

5.2. Migrating to MongoDB 2.6

MongoDB 2.4 has been available since Red Hat Software Collections 1.1 as the *mongodb24* Software Collection. Red Hat Software Collections 2.0 is shipped with **MongoDB 2.6** provided by the *rh-mongodb26* Software Collection.

5.2.1. Notable Differences Between MongoDB 2.4 and MongoDB 2.6

General Changes

The *rh-mongodb26* Software Collection introduces several general changes listed below.

- Service files have been renamed:
  - The **`/etc/rc.d/init.d/mongodb24-mongodb`** service file for the MongoDB daemon has been renamed to **`/etc/rc.d/init.d/rh-mongodb26-mongod`**
  - The **`/etc/rc.d/init.d/mongodb24-mongodb-shard`** service file for the MongoDB sharding server has been renamed to **`/etc/rc.d/init.d/rh-mongodb26-mongos`**
- Configuration and system configuration files have been renamed:
  - The **`mongod`** daemon uses the **`/etc/opt/rh/rh-mongodb26/mongod.conf`** and **`/etc/opt/rh/rh-mongodb26/sysconfig/mongod`** configuration files
  - The **`mongos`** sharding server uses the **`/etc/opt/rh/rh-mongodb26/mongos.conf`** and **`/etc/opt/rh/rh-mongodb26/sysconfig/mongos`** configuration files
- The log files have been relocated:
  - The **`mongod`** daemon now writes log to the **`/var/opt/rh/rh-mongodb26/log/mongodb/mongod.log`** file
  - The **`mongos`** sharding server writes log to the **`/var/opt/rh/rh-mongodb26/log/mongodb/mongos.log`** file
- The default **`mongos`** port number has been changed from **`27019`** to **`27017`**
- The *rh-mongodb26-mongodb-test* package, which contains the **MongoDB** test suite, has been added. For more information about usage, install this package and read the **`/opt/rh/rh-mongodb26/root/usr/share/mongodb-test/README`** file.
- The *rh-mongodb26* Software Collection supports neither mounting over NFS nor dynamical registering using the **`scl register`** command.

Compatibility Changes

MongoDB 2.6 includes various minor changes that can affect compatibility with previous versions of **MongoDB**. For a brief list of compatibility changes in **MongoDB 2.6**, refer to the Knowledgebase article on the Red Hat Customer Portal. For details on compatibility changes, see the MongoDB documentation.

Authentication Changes

MongoDB 2.6 authorization model introduces changes in the way **MongoDB** stores and manages user privilege information:

- **MongoDB 2.6** requires at least one user in the **admin** database with the **userAdminAnyDatabase** role. Make sure that this user exists before you upgrade.
- You will not be able to create or modify users or create user-defined roles in **MongoDB** versions that use previous authorization models.

For details on authentication changes, see the MongoDB documentation.

5.2.2. Upgrading from the *mongodb24* to the *rh-mongodb26* Software Collection

Note that once upgraded to **MongoDB 2.6**, you cannot downgrade to any version earlier than **MongoDB 2.4**. If you created text or **2dsphere** indexes while running **MongoDB 2.6**, you can downgrade only to **MongoDB 2.4.10** or later versions.

Important

Before migrating from the *mongodb24* to the *rh-mongodb26* Software Collection, back up all your data, including any MongoDB databases, which are by default stored in the **/opt/rh/mongodb24/root/var/lib/mongodb/** directory.

To upgrade to the *rh-mongodb26* Software Collection, perform the following steps as **root**.

1. Install the MongoDB server from the *rh-mongodb26* Software Collection:

   ```
   yum install rh-mongodb26
   ```

2. Stop the **mongodb24** server in Red Hat Enterprise Linux 6:

   ```
   service mongodb24-mongodb stop
   ```

   Use the **systemctl stop mongodb24-mongodb.service** command instead if you are using Red Hat Enterprise Linux 7.

3. Copy your data into the new location:

   ```
   cp -a /opt/rh/mongodb24/root/var/lib/mongodb/* /var/opt/rh/rh-
   mongodb26/lib/mongodb
   ```

4. Change the **dbpath** variable in the **/opt/rh/mongodb24/root/etc/mongodb.conf** file to **/var/opt/rh/rh-mongodb26/lib/mongodb/**.

5. Start the **mongodb24** server in Red Hat Enterprise Linux 6:

   ```
   service mongodb24-mongodb start
   ```

   Use the **systemctl start mongodb24-mongodb.service** command if instead you are using Red Hat Enterprise Linux 7.

6. Install the **mongo** shell from the *rh-mongodb26* Software Collection:

```
yum install rh-mongodb26-mongodb
```

7. Connect the **mongo** shell from the *rh-mongodb26* Software Collection to your **mongodb24** server (for example, running on **localhost**, port **27017**; you do not need **root** privileges for this step):

   ```
   scl enable rh-mongodb26 'mongo --host localhost --port 27017
   admin'
   ```

8. In the **mongo** shell, run the **db.upgradeCheckAllDBs()** function to check your data set for compatibility:

   ```
   db.upgradeCheckAllDBs()
   ```

   See the MongoDB documentation for more information about the **db.upgradeCheckAllDBs()** function.

9. Resolve all issues identified by **db.upgradeCheckAllDBs()** and compatibility issues mentioned above that affect your application.

10. Stop the **mongodb24** server in Red Hat Enterprise Linux 6:

    ```
    service mongodb24-mongodb stop
    ```

    Use the **systemctl stop mongodb24-mongodb.service** command instead if you are using Red Hat Enterprise Linux 7.

11. Make the *mongodb24* Software Collection runnable after the upgrade by changing the **dbpath** variable back to the previous value (**/opt/rh/mongodb24/root/var/lib/mongodb/** by default) in the **/opt/rh/mongodb24/root/etc/mongodb.conf** file.

12. Configure the **rh-mongodb26-mongod** daemon in the **/etc/opt/rh/rh-mongodb26/mongod.conf** configuration file.

13. Start the MongoDB server from the *rh-mongodb26* Collection in Red Hat Enterprise Linux 6:

    ```
    service rh-mongodb26-mongod start
    ```

    Use the **systemctl start rh-mongodb26-mongod.service** instead if you are using Red Hat Enterprise Linux 7.

14. Upgrade the authorization model as described in the MongoDB documentation. Note that it is recommended to run your MongoDB deployment for a day or two before you upgrade the user authorization model because downgrades are more difficult after the user authorization model has been upgraded. Before you upgrade the authorization model, you will not be able to create or modify users or to use user-defined roles.

For detailed information about upgrading, refer to the MongoDB documentation, or particularly about upgrading a Replica Set or a Sharded Cluster.

5.3. Migrating to MySQL 5.6

Red Hat Enterprise Linux 6 contains **MySQL 5.1** as the default **MySQL** implementation. Red Hat Enterprise Linux 7 includes **MariaDB 5.5** as the default **MySQL** implementation. In addition to these basic versions, **MySQL 5.5** has been available as a Software Collection for Red Hat Enterprise Linux 6 since Red Hat Software Collections 1.0 and for Red Hat Enterprise Linux 7 since Red Hat Software Collections 1.1.

The *rh-mysql56* Software Collection available for both Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 does not conflict with the *mysql* or *mariadb* packages from the core systems, so it is possible to install the *rh-mysql56* Software Collection together with the *mysql* or *mariadb* packages. It is also possible to run both versions at the same time, however, the port number and the socket in the **`my.cnf`** files need to be changed to prevent these specific resources from conflicting.

Note that it is possible to upgrade to **MySQL 5.6** only from **MySQL 5.5**. If you need to upgrade from an earlier version, upgrade to **MySQL 5.5** first. Instructions how to upgrade to **MySQL 5.5** are available in the Red Hat Software Collections 1.2 Release Notes.

5.3.1. Notable Differences Between MySQL 5.5 and MySQL 5.6

The *rh-mysql56* Software Collection introduces the following notable changes:

- The service has been renamed to **`rh-mysql56-mysqld`** in both Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7
- The **`test`** database is no longer created by default
- Configuration files for the *rh-mysql56* Software Collection are the **`/etc/opt/rh/rh-mysql56/my.cnf`** file and in the **`/etc/opt/rh/rh-mysql56/my.cnf.d/`** directory
- Variable files including the database files for the *rh-mysql56* Software Collection are located in the **`/var/opt/rh/rh-mysql56/lib/`** directory
- The log file for the MySQL daemon is **`/var/opt/rh/rh-mysql56/log/mysql/mysqld.log`**
- The pid file for the daemon is **`/var/run/rh-mysql56-mysqld/mysqld.pid`**

Note that the *rh-mysql56* Software Collection supports neither mounting over NFS nor dynamical registering using the **`scl register`** command.

For detailed changes, refer to the MySQL documentation.

5.3.2. Upgrading to the *rh-mysql56* Software Collection

Important

Prior to upgrading, back-up all your data, including any MySQL databases.

Upgrading can be performed either by using the **mysqldump** and **mysqlimport** utilities or using an in-place upgrade.

- In the former scenario, the whole dump of all databases from one database is generated and **`mysql`** is run with the dump file as an input using the **`mysqlimport`** or **`LOAD DATA INFILE SQL`** command within the other database. At the same time, the appropriate daemons have to be running during both dumping and restoring. You can use the **`--all-databases`** option in the **`mysqldump`** call to include all databases in the dump. The **`--routines`**, **`--triggers`**, and **`--events`** options can also be used if needed.

- During the in-place upgrade, the data files are copied from one database directory to another database directory. The daemons must not be running at the time of copying. Set appropriate permissions and SELinux context for the copied files.

After upgrading, start the server and run the **`mysql_upgrade`** command. Running **`mysql_upgrade`** is necessary to check and repair internal tables.

In case the **`root`** user has a non-empty password defined (it should have a password defined), it is necessary to call the **`mysql_upgrade`** utility with the **`-p`** option and specify the password.

Service names and paths bellow depend on which version you are upgrading from.

Example 5.3. Dump and Restore Upgrade

1. Create a backup from the *mysql55* Software Collection:

```
~]# service mysql55-mysqld start
Starting mysql55-mysqld:                                   [  OK
]
~]# scl enable mysql55 -- mysqldump --all-databases --routines
--events > dump.sql
~]# service mysql55-mysqld stop
Stopping mysql55-mysqld:                                   [  OK
]
```

   For upgrading from the *mariadb55* Software Collection in Red Hat Enterprise Linux 6, use **`mariadb55-mysqld`** as the service name.

   For upgrading from the *mariadb55* Software Collection in Red Hat Enterprise Linux 7, use **`mariadb55-mariadb`** as the service name.

   For upgrading from **MariaDB 5.5** from base Red Hat Enterprise Linux 7, use **`mariadb`** as the service name and do not use **`scl enable mysql55 --`** when creating the dump.

2. Import the dumped database into the *rh-mysql56* Software Collection:

```
~]# service rh-mysql56-mysqld start
Starting rh-mysql56-mysqld:                                [  OK
]
~]# scl enable rh-mysql56 'mysql' < dump.sql
~]# scl enable rh-mysql56 'mysql_upgrade -u root -p'
Enter password:
Looking for 'mysql' as: mysql
Looking for 'mysqlcheck' as: mysqlcheck
Running 'mysqlcheck with default connection arguments
Running 'mysqlcheck with default connection arguments
a.t1                                               OK
mysql.columns_priv                                 OK
<skipped tables list>
mysql.user                                         OK
Running 'mysql_fix_privilege_tables'...
OK
```

Example 5.4. In-place Upgrade from MySQL 5.5

If you are upgrading from the *mysql55* Software Collection, perform the upgrade as shown in the following example:

```
~]# service mysql55-mysqld stop
Stopping mysql55-mysqld                                [  OK  ]
~]# service rh-mysql56-mysqld stop
Stopping rh-mysql56-mysqld:                            [  OK  ]
~]# rm -rf /var/opt/rh/rh-mysql56/lib/mysql/
~]# cp -r /opt/rh/mysql55/root/var/lib/mysql/ /var/opt/rh/rh-
mysql56/lib/mysql/
~]# chown -R mysql:mysql /var/opt/rh/rh-mysql56/lib/mysql/
~]# restorecon -R /var/opt/rh/rh-mysql56/lib/mysql/
~]# service rh-mysql56-mysqld start
Starting rh-mysql56-mysqld:                            [  OK  ]
~]# scl enable rh-mysql56 'mysql_upgrade -u root -p'
Enter password:
Looking for 'mysql' as: mysql
Looking for 'mysqlcheck' as: mysqlcheck
Running 'mysqlcheck with default connection arguments
Running 'mysqlcheck with default connection arguments
a.t1                                              OK
mysql.columns_priv                                OK
<skipped tables list>
mysql.user                                        OK
Running 'mysql_fix_privilege_tables'...
OK
```

For upgrading from the *mariadb55* Software Collection, use the **/opt/rh/mariadb55/root/var/lib/mysql/** as a source when copying the data.

For upgrading from **MariaDB 5.5** from base Red Hat Enterprise Linux 7, use the **/var/lib/mysql/** as a source when copying the data.

For more details about migration to **MySQL 5.6**, refer to the MySQL documentation.

5.4. Migrating to PostgreSQL 9.4

Red Hat Software Collections 2.0 is distributed with **PostgreSQL 9.4**, which can be safely installed on the same machine in parallel with **PostgreSQL 8.4** from Red Hat Enterprise Linux 6 or **PostgreSQL 9.2** from Red Hat Enterprise Linux 7 or Red Hat Software Collections 1. It is also possible to run more than one version of **PostgreSQL** on a machine at the same time, but you need to use different ports or IP addresses and adjust SELinux policy.

5.4.1. Notable Differences Between PostgreSQL 9.2 and PostgreSQL 9.4

The most notable changes between **PostgreSQL 9.2** and **PostgreSQL 9.4** are the following:

- **PostgreSQL 9.4** no longer includes native support for Kerberos authentication (for example, using the **--with-krb5** option). As consequence, the **krb_srvname** option is not available anymore. The supported way to use Kerberos authentication is with Generic Security Services API (GSSAPI).

- Since **PostgreSQL 9.4**, the **replication_timeout** configuration option has been split into the **wal_receiver_timeout** and **wal_sender_timeout** options.

- The **`scl register rh-postgresql94`** command is unsupported and the *rh-postgresql94* Software Collection is not supported to run over NFS.

The following table provides an overview of different paths in a Red Hat Enterprise Linux system version of **PostgreSQL** (*postgresql*) and in the *postgresql92* and *rh-postgresql94* Software Collections. Note that the paths of **PostgreSQL 8.4** distributed with Red Hat Enterprise Linux 6 and the system version of **PostgreSQL 9.2** shipped with Red Hat Enterprise Linux 7 are the same.

Table 5.1. Diferences in the PostgreSQL paths

Content	*postgresql*	*postgresql92*	*rh-postgresql94*
Executables	/usr/bin/	/opt/rh/postgresql92/root/usr/bin/	/opt/rh/rh-postgresql94/root/usr/bin/
Libraries	/usr/lib64/	/opt/rh/postgresql92/root/usr/lib64/	/opt/rh/rh-postgresql94/root/usr/lib64/
Documentation	/usr/share/doc/postgresql/html/	/opt/rh/postgresql92/root/usr/share/doc/postgresql/html/	/opt/rh/rh-postgresql94/root/usr/share/doc/postgresql/html/
PDF documentation	/usr/share/doc/postgresql-docs/	/opt/rh/postgresql92/root/usr/share/doc/postgresql-docs/	/opt/rh/rh-postgresql94/root/usr/share/doc/postgresql-docs/
Contrib documentation	/usr/share/doc/postgresql-contrib/	/opt/rh/postgresql92/root/usr/share/doc/postgresql-contrib/	/opt/rh/rh-postgresql94/root/usr/share/doc/postgresql-contrib/
Source	not installed	not installed	not installed
Data	/var/lib/pgsql/data/	/opt/rh/postgresql92/root/var/lib/pgsql/data/	/var/opt/rh/rh-postgresql94/lib/pgsql/data/
Backup area	/var/lib/pgsql/backups/	/opt/rh/postgresql92/root/var/lib/pgsql/backups/	/var/opt/rh/rh-postgresql94/lib/pgsql/backups/
Templates	/usr/share/pgsql/	/opt/rh/postgresql92/root/usr/share/pgsql/	/opt/rh/rh-postgresql94/root/usr/share/pgsql/
Procedural Languages	/usr/lib64/pgsql/	/opt/rh/postgresql92/root/usr/lib64/pgsql/	/opt/rh/rh-postgresql94/root/usr/lib64/pgsql/
Development Headers	/usr/include/pgsql/	/opt/rh/postgresql92/root/usr/include/pgsql/	/opt/rh/rh-postgresql94/root/usr/include/pgsql/
Other shared data	/usr/share/pgsql/	/opt/rh/postgresql92/root/usr/share/pgsql/	/opt/rh/rh-postgresql94/root/usr/share/pgsql/
Regression tests	/usr/lib64/pgsql/test/regress/ (in the -test package)	/opt/rh/postgresql92/root/usr/lib64/pgsql/test/regress/ (in the -test package)	/opt/rh/rh-postgresql94/root/usr/lib64/pgsql/test/regress/ (in the -test package)

For detailed changes, see the PostgreSQL 9.3 Release Notes and the PostgreSQL 9.4 Release Notes. For changes between **PostgreSQL 8.4** and **PostgreSQL 9.2**, refer to the Red Hat Software Collections 1.2 Release Notes.

5.4.2. Migrating from a Red Hat Enterprise Linux System Version of PostgreSQL to the PostgreSQL 9.4 Software Collection

Red Hat Enterprise Linux 6 includes **PostgreSQL 8.4**, Red Hat Enterprise Linux 7 is distributed with **PostgreSQL 9.2**. To migrate your data from a Red Hat Enterprise Linux system version of **PostgreSQL** to the *rh-postgresql94* Software Collection, you can either perform a fast upgrade using the **pg_upgrade** tool (recommended), or dump the database data into a text file with SQL commands and import it in the new database. Note that the second method is usually significantly slower and may require manual fixes; see the PostgreSQL documentation for more information about this upgrade method. The following procedures are applicable for both Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7 system versions of **PostgreSQL**.

Important

Before migrating your data from a Red Hat Enterprise Linux system version of PostgreSQL to PostgreSQL 9.4, make sure that you back up all your data, including the PostgreSQL database files, which are by default located in the **/var/lib/pgsql/data/** directory.

Procedure 5.1. Fast Upgrade Using the pg_upgrade Tool

To perform a fast upgrade of your PostgreSQL server, complete the following steps:

1. Stop the old PostgreSQL server to ensure that the data is not in an inconsistent state. To do so, type the following at a shell prompt as **root**:

   ```
   service postgresql stop
   ```

   To verify that the server is not running, type:

   ```
   service postgresql status
   ```

2. Verify that the old directory **/var/lib/pgsql/data/** exists:

   ```
   file /var/lib/pgsql/data/
   ```

   and back up your data.

3. Verify that the new data directory **/var/opt/rh/rh-postgresql94/lib/pgsql/data/** does not exist:

   ```
   file /var/opt/rh/rh-postgresql94/lib/pgsql/data/
   ```

   If you are running a fresh installation of **PostgreSQL 9.4**, this directory should not be present in your system. If it is, back it up by running the following command as **root**:

   ```
   mv /var/opt/rh/rh-postgresql94/lib/pgsql/data{,-scl-backup}
   ```

4. Upgrade the database data for the new server by running the following command as **root**:

```
scl enable rh-postgresql94 -- postgresql-setup --upgrade
```

Alternatively, you can use the **/opt/rh/rh-postgresql94/root/usr/bin/postgresql-setup --upgrade** command.

Note that you can use the **--upgrade-from** option for upgrade from different versions of **PostgreSQL**. The list of possible upgrade scenarios is available using the **--upgrade-ids** option.

It is recommended that you read the resulting **/var/lib/pgsql/upgrade_rh-postgresql94-postgresql.log** log file to find out if any problems occurred during the upgrade.

5. Start the new server as **root**:

```
service rh-postgresql94-postgresql start
```

It is also advised that you run the **analyze_new_cluster.sh** script as follows:

```
su - postgres -c 'scl enable rh-postgresql94
~/analyze_new_cluster.sh'
```

6. Optionally, you can configure the PostgreSQL 9.4 server to start automatically at boot time. To disable the old system PostgreSQL server, type the following command as **root**:

```
chkconfig postgresql off
```

To enable the PostgreSQL 9.4 server, type as **root**:

```
chkconfig rh-postgresql94-postgresql on
```

7. If your configuration differs from the default one, make sure to update configuration files, especially the **/var/opt/rh/rh-postgresql94/lib/pgsql/data/pg_hba.conf** configuration file. Otherwise only the **postgres** user will be allowed to access the database.

Procedure 5.2. Performing a Dump and Restore Upgrade

To perform a dump and restore upgrade of your PostgreSQL server, complete the following steps:

1. Ensure that the old PostgreSQL server is running by typing the following at a shell prompt as **root**:

```
service postgresql start
```

2. Dump all data in the PostgreSQL database into a script file. As **root**, type:

```
su - postgres -c 'pg_dumpall > ~/pgdump_file.sql'
```

3. Stop the old server by running the following command as **root**:

```
service postgresql stop
```

4. Initialize the data directory for the new server as **root**:

```
scl enable rh-postgresql94-postgresql -- postgresql-setup --
initdb
```

5. Start the new server as **root**:

```
service rh-postgresql94-postgresql start
```

6. Import data from the previously created SQL file:

```
su - postgres -c 'scl enable rh-postgresql94 "psql -f
~/pgdump_file.sql postgres"'
```

7. Optionally, you can configure the PostgreSQL 9.4 server to start automatically at boot time. To disable the old system PostgreSQL server, type the following command as **root**:

```
chkconfig postgresql off
```

   To enable the PostgreSQL 9.4 server, type as **root**:

```
chkconfig rh-postgresql94-postgresql on
```

8. If your configuration differs from the default one, make sure to update configuration files, especially the **/var/opt/rh/rh-postgresql94/lib/pgsql/data/pg_hba.conf** configuration file. Otherwise only the **postgres** user will be allowed to access the database.

5.4.3. Migrating from the PostgreSQL 9.2 Software Collection to the PostgreSQL 9.4 Software Collection

To migrate your data from the *postgresql92* Software Collection to the *rh-postgresql94* Collection included in Red Hat Software Collections 2.0, you can either perform a fast upgrade using the **pg_upgrade** tool (recommended), or dump the database data into a text file with SQL commands and import it in the new database. Note that the second method is usually significantly slower and may require manual fixes; see the PostgreSQL documentation for more information about this upgrade method.

Important

Before migrating your data from **PostgreSQL 9.2** to **PostgreSQL 9.4**, make sure that you back up all your data, including the PostgreSQL database files, which are by default located in the **/opt/rh/postgresql92/var/lib/pgsql/data/** directory.

Procedure 5.3. Fast Upgrade Using the pg_upgrade Tool

To perform a fast upgrade of your PostgreSQL server, complete the following steps:

1. Stop the old PostgreSQL server to ensure that the data is not in an inconsistent state. To do so, type the following at a shell prompt as **root**:

```
service postgresql92-postgresql stop
```

   To verify that the server is not running, type:

```
service postgresql92-postgresql status
```

2. Verify that the old directory **`/opt/rh/postgresql92/var/lib/pgsql/data/`** exists:

   ```
   file /opt/rh/postgresql92/var/lib/pgsql/data/
   ```

   and back up your data.

3. Verify that the new data directory **`/var/opt/rh/rh-postgresql94/lib/pgsql/data/`** does not exist:

   ```
   file /var/opt/rh/rh-postgresql94/lib/pgsql/data/
   ```

   If you are running a fresh installation of **PostgreSQL 9.4**, this directory should not be present in your system. If it is, back it up by running the following command as **`root`**:

   ```
   mv /var/opt/rh/rh-postgresql94/lib/pgsql/data{,-scl-backup}
   ```

4. Upgrade the database data for the new server by running the following command as **`root`**:

   ```
   scl enable rh-postgresql94 -- postgresql-setup --upgrade --
   upgrade-from=postgresql92-postgresql
   ```

   Alternatively, you can use the **`/opt/rh/rh-postgresql94/root/usr/bin/postgresql-setup --upgrade --upgrade-from=postgresql92-postgresql`** command.

   Note that you can use the **`--upgrade-from`** option for upgrading from different versions of **PostgreSQL**. The list of possible upgrade scenarios is available using the **`--upgrade-ids`** option.

   It is recommended that you read the resulting **`/var/lib/pgsql/upgrade_rh-postgresql94-postgresql.log`** log file to find out if any problems occurred during the upgrade.

5. Start the new server as **`root`**:

   ```
   service rh-postgresql94-postgresql start
   ```

   It is also advised that you run the **`analyze_new_cluster.sh`** script as follows:

   ```
   su - postgres -c 'scl enable rh-postgresql94
   ~/analyze_new_cluster.sh'
   ```

6. Optionally, you can configure the PostgreSQL 9.4 server to start automatically at boot time. To disable the old PostgreSQL 9.2 server, type the following command as **`root`**:

   ```
   chkconfig postgresql92-postgreqsql off
   ```

   To enable the PostgreSQL 9.4 server, type as **`root`**:

   ```
   chkconfig rh-postgresql94-postgresql on
   ```

7. If your configuration differs from the default one, make sure to update configuration files, especially the **/var/opt/rh/rh-postgresql94/lib/pgsql/data/pg_hba.conf** configuration file. Otherwise only the **postgres** user will be allowed to access the database.

Procedure 5.4. Performing a Dump and Restore Upgrade

To perform a dump and restore upgrade of your PostgreSQL server, complete the following steps:

1. Ensure that the old PostgreSQL server is running by typing the following at a shell prompt as **root**:

   ```
   service postgresql92-postgresql start
   ```

2. Dump all data in the PostgreSQL database into a script file. As **root**, type:

   ```
   su - postgres -c 'scl enable postgresql92 "pg_dumpall" >
   ~/pgdump_file.sql'
   ```

3. Stop the old server by running the following command as **root**:

   ```
   service postgresql92-postgresql stop
   ```

4. Initialize the data directory for the new server as **root**:

   ```
   scl enable rh-postgresql94-postgresql -- postgresql-setup --
   initdb
   ```

5. Start the new server as **root**:

   ```
   service rh-postgresql94-postgresql start
   ```

6. Import data from the previously created SQL file:

   ```
   su - postgres -c 'scl enable rh-postgresql94 "psql -f
   ~/pgdump_file.sql postgres"'
   ```

7. Optionally, you can configure the PostgreSQL 9.4 server to start automatically at boot time. To disable the old PostgreSQL 9.2 server, type the following command as **root**:

   ```
   chkconfig postgresql92-postgresql off
   ```

   To enable the PostgreSQL 9.4 server, type as **root**:

   ```
   chkconfig rh-postgresql94-postgresql on
   ```

8. If your configuration differs from the default one, make sure to update configuration files, especially the **/var/opt/rh/rh-postgresql94/lib/pgsql/data/pg_hba.conf** configuration file. Otherwise only the **postgres** user will be allowed to access the database.

5.5. Migrating to nginx 1.6

The *nginx16* Software Collection uses a new prefix in accordance with the name of the collection and a different path to the root directory, which is now located in **`/opt/rh/nginx16/root/`**. The error log is now stored in **`/var/log/nginx16/error.log`** by default, and the initscript is called **`nginx16-nginx`**.

Configuration files in nginx 1.6 have the same format as in the previous version and they are compatible between version 1.4 and 1.6.

Important

Before upgrading from nginx 1.4 to nginx 1.6, back up all your data, including web pages and configuration files located in the **`/opt/rh/nginx14/root/`** tree.

If you have made any specific changes, such as changing configuration files or setting up web applications, in the **`/opt/rh/nginx14/root/`** tree, replicate those changes in the new **`/opt/rh/nginx16/root/`** directory, too.

For the official **`nginx`** documentation, refer to http://nginx.org/en/docs/.

Chapter 6. Additional Resources

This chapter provides references to other relevant sources of information about Red Hat Software Collections 2.0 and Red Hat Enterprise Linux.

6.1. Red Hat Enterprise Linux Developer Program Group

Users of Red Hat Software Collections can access the Red Hat Enterprise Linux Developer Program Group in the Red Hat Customer Portal to get developer related information for the development tools available for Red Hat Enterprise Linux. In addition, users can find developer related papers and videos on topics that are of interest to developers, for example RPM building, threaded programming, performance tuning, debugging, and so on.

To visit the Red Hat Enterprise Linux Developer Program Group, log in to the Red Hat Customer Portal, click **Products and Services** at the top of the page, choose **Services**, and then **Red Hat Enterprise Linux Developer Program** from the list.

6.2. Red Hat Product Documentation

The following documents are directly or indirectly relevant to this book:

- Red Hat Software Collections 2.0 Packaging Guide — The *Packaging Guide* for Red Hat Software Collections explains the concept of Software Collections, documents the **scl** utility, and provides a detailed explanation of how to create a custom Software Collection or extend an existing one.
- Red Hat Developer Toolset 3.1 Release Notes — The *Release Notes* for Red Hat Developer Toolset document known problems, possible issues, changes, and other important information about this Software Collection.
- Red Hat Developer Toolset 3.1 User Guide — The *User Guide* for Red Hat Developer Toolset contains more information about installing and using this Software Collection.
- Using and Configuring Red Hat Subscription Manager — The *Using and Configuring Red Hat Subscription Manager* book provides detailed information on how to register Red Hat Enterprise Linux systems, manage subscriptions, and view notifications for the registered systems.
- Red Hat Enterprise Linux 6 Deployment Guide — The *Deployment Guide* for Red Hat Enterprise Linux 6 provides relevant information regarding the deployment, configuration, and administration of this system.
- Red Hat Enterprise Linux 7 System Administrator's Guide — The *System Administrator's Guide* for Red Hat Enterprise Linux 7 provides information on deployment, configuration, and administration of this system.

6.3. Red Hat Developer Blog

Red Hat Developer Blog content is directed to designers and developers of applications based on Red Hat technologies. It contains links to product team blogs and other relevant internal and external resources. Its goal is to inform and engage the developer community with up-to-date information, best practices, opinion, product and program announcements as well as pointers to sample code and other resources.

Appendix A. Revision History

Revision 2.0-15	**Thu Jun 04 2015**	**Lenka Špačková**

Release of Red Hat Software Collections 2.0 Release Notes.

Revision 2.0-5	**Thu Apr 23 2015**	**Lenka Špačková**

Release of Red Hat Software Collections 2.0 Beta Release Notes.