RED HAT®
ENTERPRISE LINUX®
DEVELOPER PROGRAM

WHITE PAPER

RED HAT ENTERPRISE LINUX DEVELOPER'S GETTING STARTED GUIDE

EXECUTIVE SUMMARY

Red Hat Enterprise Linux is an enterprise-class open-source operating system that is widely adopted world wide, scaling seamlessly from individual desktops to large servers in the datacenter. Certified by leading hardware and software vendors, Red Hat Enterprise Linux delivers high performance, reliability, and security along with flexibility, efficiency, and control. For developers, Red Hat provides an extensive set of resources, technologies, and tools that can be used to efficiently develop powerful applications for the Red Hat Enterprise Linux platform. These applications can be deployed with great flexibility, as Red Hat Enterprise Linux supports major hardware architectures, comprehensive virtualization solutions, and a range of cloud-computing options.

This document is intended for software developers who are new to Red Hat Enterprise Linux and want to understand the key touch points for any phase of application development – from planning and building, through testing and deploying. The following sections describe the resources and tools that are available on Red Hat Enterprise Linux and provide links to additional information.

TABLE OF CONTENTS

DEVELOPING SOFTWARE ON RED HAT ENTERPRISE LINUX

Development for Linux can be challenging: it is the only ecosystem where no one is telling you how to build your applications. While this freedom provides a huge amount of flexibility, it also brings risk. Primarily, that risk comes from relying on open source for not only the operating system itself, but also for enormous parts of the development stack. Many people will tout the virtues of the quality of open source software (which are true). For example, "Given enough eyeballs, all bugs are shallow,"[1] and "In the cryptography world, we consider open source necessary for good security; we have for decades."[2] However, that does not mean that every open source product is a good one, still exists, or is complete. The Internet is littered with half-complete or abandoned excellent, in concept, components. As a result, developers often spend a lot of their time researching components for viability, updates, and general status and doing their development with their "IDE in one hand and Google in the other."

Red Hat provides developers who target Red Hat Enterprise Linux platforms with a number of "risk-aversion tools". First and foremost is the promise of every open source project: every line of code that Red Hat ships with its platform is available to peruse or modify as you see fit. Furthermore, Red Hat also supports the code that it ships to an even greater degree than most competitors, often going above and beyond the letter of the contract to ensure its customers are successful.

Red Hat and its employees contribute to many open source projects, some of which are shipped with Red Hat products and some that are not. Red Hat's customer base are also strong proponents and contributors to the open source community. As a result, the Red Hat Ecosystem (implemented as the Red Hat Network and award-winning Red Hat Customer Portal) is a strong community of helpful people, articles, videos, reference architectures, and tools. The Red Hat Network also helps developers find the information they need faster by scoping the search space to just the components they want and need. Unlike developing with "Google in one hand," a developer doesn't get numerous extraneous results when searching (check out the difference in a search for "terminal": https://www.google.com/search?q=terminal versus http://red.ht/L0yIlT).

Developers who work on Red Hat platforms during development also benefit from lower-risk deployments when they ultimately move to production. Developers who work directly on Red Hat platforms have already resolved platform-related issues such as security and component versions, to name a few.

The Red Hat Enterprise Linux Developer Program brings all these components together into an easy to access environment. This program also brings tools (such as the Red Hat Enterprise Linux Developer Toolset) that are officially supported by Red Hat's team and provide for supported deployments across all active Red Hat Enterprise Linux versions.

So, let's get started! The remainder of this document is a navigation guide, from a developer's perspective, to the resources available within the Red Hat Network.

1 Raymond, Eric S. "The Cathedral and the Bazaar". http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html
2 Schneier, Bruce. Crypto-Gram Newsletter, September 15, 1999. http://www.schneier.com/crypto-gram-9909.html

Overview

The Red Hat Enterprise Linux platform provides developers with the tools and resources needed for all phases of the software development life cycle (see Figure 1). In addition to an all-inclusive set of tools, Red Hat also provides developers with the resources and support to effectively and productively use these tools for application development.

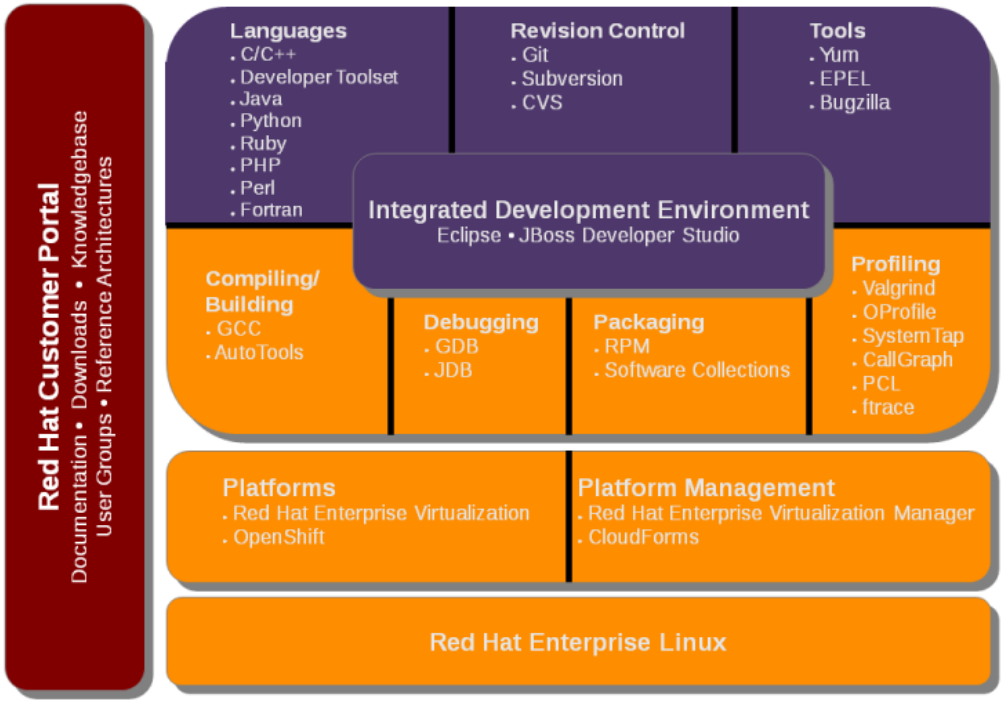

Figure 1: Developer resources and tools for Red Hat Enterprise Linux.

The tools and resources of the Red Hat Enterprise Developer Program are loosely categorized into three broad pillars: Knowledge, Access, and Deployment.

- Knowledge (shown in red in Figure 1) – Resources related to information and education about Red Hat products and development using those produces, including product documentation, downloads, and an extensive knowledgebase, as well as a collaborative developer forum for members of the development community and Red Hat staff.
- Access (shown in purple in Figure 1) – Technical tools and components needed to support development activities, such as integrated development environments, revision control systems, libraries, and other development tools.
- Deployment (shown in orange in Figure 1) – A comprehensive set of tools and resources for packaging and releasing software, on-premises or in the cloud.

KNOWLEDGE

The "Knowledge Pillar" of the Red Hat Enterprise Linux Developer Program provides developers with information on development and administration of Red Hat products. This section of the guide helps you find out where to go to get various type of information. Currently, the primary delivery system is the Red Hat Customer Portal that contains knowledgebase articles, reference architectures, and the developer forum. However, watch this space for significant advances in the types of information available over the coming months.

Red Hat Customer Portal

The Red Hat Customer Portal is available to all Red Hat subscribers and serves as a pointer to a vast number of developer resources. The best place to start is at http://start.redhat.com. This welcome page provides a link to the *Red Hat Welcome Kit*, which includes information on product registration, accessing resources, and customer support. The Welcome Guide also includes information on registering for a Red Hat Network (RHN) account and activating subscriptions at http://rhn.redhat.com. Registering with Red Hat and creating a Red Hat account provides access to an extensive library of white papers, product evaluations, and other valuable information. Purchasing a software subscription provides additional support capabilities and access to Red Hat software, upgrades, technical support, and security fixes.

After the initial visit, the Red Hat Customer Portal (see Figure 2) can be accessed at https://access.redhat.com/home. A quick and easy link to the Red Hat Network is supplied at the top of the page. Resources needed during each stage of development are organized into three main menus: Plan, Deploy, and Connect. The Portal also organizes information in tab lists along the top of the web page. Developers can easily access Knowledge, Groups, Support, Downloads, Security, and Subscriptions directly from the Portal home page.

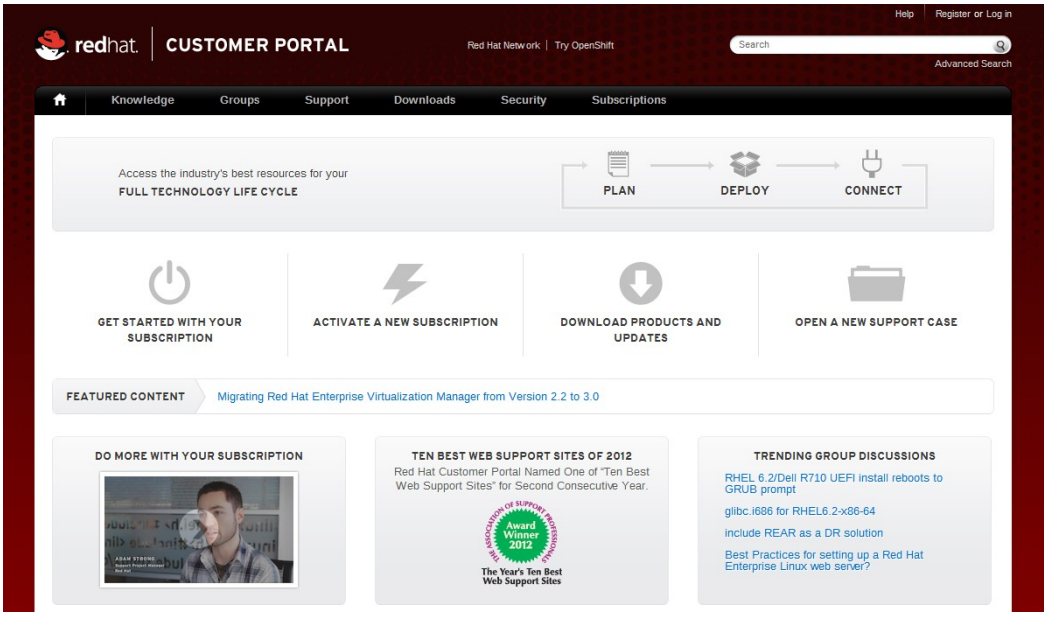

Figure 2: Red Hat Customer Portal provides easy access to developer resources and support.

Links to activate new subscriptions, download Red Hat products and updates, and open a support case are also included on the portal home page, making it easy for developers to quickly access the resources they need.

Plan Menu

The Plan menu provides links to resources needed to plan a development project, including:

- Webinars – View useful information about Red Hat products, key deployment considerations, and best practices.
- Success Stories – Learn how leading organizations are finding unbeatable value, performance, security, and reliability with Red Hat solutions.
- Product Documentation – Review high-level user guides and other product information.
- Reference Architectures – Check out detailed technical case studies of solutions that have been built, tested, and benchmarked in Red Hat labs by senior Red Hat engineers. The Reference Architectures cover a range of topics, from designing the components of a datacenter to the recommended tiers for an application architecture, and much more.
- Evaluations & Demos – Download free evaluations and development versions of various Red Hat products, including Red Hat Enterprise Linux, Red Hat Enterprise Virtualization, Red Hat Storage Appliance, and JBoss Enterprise Middleware evaluations.
- Certified Hardware – Browse hardware options that have been certified by Red Hat engineers for use with Red Hat solutions.

Deploy Menu

The Deploy menu provides links to resources needed to deploy a development project, including:

- Getting Started – Get information on how to access, download, and install Red Hat solutions, including product registration information and engaging in global support.
- Downloads – Download the latest supported and archived versions of Red Hat software.
- Activate a New Subscription – Activate your subscription to gain access to the latest updates and Red Hat support.
- Support Essentials – View brief articles covering security alerts, hardware certifications, and FAQs.
- Security – Review product life cycle documentation, update policies, errata, and other security-related information.
- Support – Access information regarding Red Hat support, including links for support cases, support programs, product life cycles, and help and assistance.

Connect Menu

The Connect menu provides links to connect to the industry's best engineers and collaborate with industry peers, including:

- Knowledgebase – Find answers to resolve support questions with direct access to the knowledge-centered support (KCS) system used by Red Hat engineers.

- **Online User Groups** – Connect and collaborate with your peers and Red Hat experts by joining a group, and creating or commenting on discussions to share ideas or solve problems.
- **Videos** – View vital solutions, useful tips, product demonstrations, and inside information in comprehensive videos featuring Red Hat engineers and experts. You can also rate, comment on, and share all videos.
- **Tech Briefs** – Receive practical advice to help solve real-world problems with Red Hat products. Each tech brief provides a detailed use case covering recommended practices, how-to instructions, or detailed discussions on how to use Red Hat technology effectively, and is reviewed and tested by Red Hat engineers.
- **Source** – Browse the kernel source for detailed information about the hundreds of discrete patches that go into the kernel.
- **Support Cases** – Gain access to Red Hat engineers by creating and managing cases through the Red Hat support ticketing and reporting system.

Note: the Knowledgebase, Groups, Source, and Support Cases sections require a valid Red Hat login and password.

Table 1: Additional resources for the Red Hat Customer Portal

TYPE	DESCRIPTION
Online Docs	*Red Hat Welcome Kit* and *Quick Guide to Red Hat Support* – https://access.redhat.com/support/start

Online Groups

Online groups provide a way for developers to collaborate with other developers and experts from Red Hat. The Groups Dashboard on the Red Hat Customer Portal (accessible from either the Connect menu or the Groups tab on the Portal home page) provides an overview of the various Red Hat discussion groups.

Of particular interest to developers is the Red Hat Enterprise Linux Developer Program group, a new place to get developer-related information for all development tools found in Red Hat Enterprise Linux. This group includes links to developer-related papers and videos on topics such as RPM building, threaded programming, performance tuning, NUMA development, debugging, and much more. Developers can join groups, follow their discussions, and create new discussions and collaborative documents.

ACCESS

The Red Hat Enterprise Linux Developer Program provides "Access" to tools for development. Many of these tools are provided by native Red Hat Enterprise Linux, and some are provided by Red Hat Enterprise Linux Developer Suite. This guide provides a centralized set of pointers to all the tools you need for development, no matter what language or stack you use.

Tools

Red Hat Enterprise Linux supports powerful and flexible developer tools and utilities for software development, including:

- Integrated Development Environments (IDEs): Eclipse and JBoss Developer Studio
- Revision Control Systems, such as G, Apache Subversion (SVN), and Concurrent Versions System (CVS)
- Yum, the Red Hat Update Manager
- Bugzilla, the bug-tracking system for Red Hat software
- Extra Packages for Enterprise Linux (EPEL)

Eclipse

Eclipse is a powerful platform- and language-neutral Integrated Development Environment (IDE) that provides tools for the entire software development process. Integrated into a single, fully configurable graphical user interface (see Figure 3), Eclipse features an extensible, pluggable architecture.

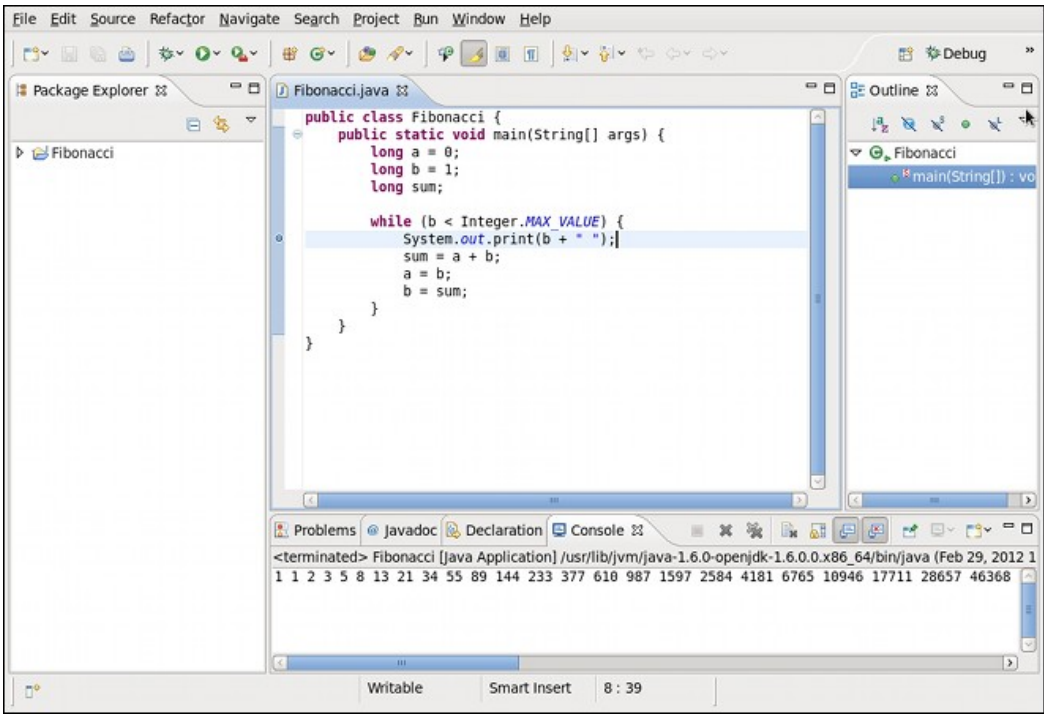

Figure 3. Eclipse user interface displaying the workbench for a Java project.

Eclipse provides editing support such as keyword and syntax coloring, code formatting, and context-specific code assistance for faster programming. Plug-ins provide additional capabilities

such as automated build assistance, revision control, integrated debugging, profiling support, and bug tracking.

Red Hat Enterprise Linux supports the primary Eclipse development toolkits for C/C++ (CDT) and Java (JDT). These toolkits provide a set of integrated tools specific to their respective languages. Both CDT and JDT also provide multiple editors for a variety of file types used in a project. For example, the CDT supplies different editors specific for C/C++ header files and source files, along with a Makefile editor. Other plug-ins support development in additional programming languages including Perl, PHP, Python, Ruby, and others.

Eclipse includes a built-in Help system that provides extensive documentation for each integrated feature and tool. The easy-to-use graphical interface, key-mapping for other common editors, and built-in Help system greatly decrease the initial time investment required for new developers to become fluent in the Eclipse environment.

Table 2: Additional resources for Eclipse IDE

TYPE	DESCRIPTION
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Chapter 1 Introduction to Eclipse and Chapter 2 The Eclipse IDE
	Eclipse Community Forums – http://www.eclipse.org/forums/

JBoss Developer Studio

JBoss Developer Studio is an open-source, integrated development environment (IDE) that provides developers with everything they need to build rich enterprise applications and web applications and services. Unlike stand-alone IDEs, JBoss Developer Studio integrates both tooling and runtime components by combining Eclipse, Eclipse Tooling, and the JBoss Enterprise Platform. JBoss Developer Studio includes the JBoss Enterprise Application Platform, JBoss SOA Platform, JBoss Enterprise Data Services Platform, and JBoss Portal Platform. It also includes JBoss Operations Network, a management and monitoring tool, and access to OpenJDK.

Table 3: Additional resources for JBoss Developer Studio

TYPE	DESCRIPTION
Online Docs	http://www.redhat.com/products/jbossenterprisemiddleware/developer-studio/

Revision Control Systems

Effective revision control is essential to software development projects of all sizes. Revision control records all changes to a set of files and enables developers to track modifications, compare changes, and revert to an earlier version if necessary. A revision control system is useful for solo development, and is critical for group projects as it allows all developers in a team to create, review, revise, and document code in a systematic and orderly manner.

Red Hat Enterprise Linux supports three of the most popular open-source revision control systems: Git, Apache Subversion (SVN), and Concurrent Versions System (CVS). These three revision control systems are all fully functional, the tools are publicly available open-source code, and all three systems integrate with the Eclipse IDE through plug-ins. The primary difference between these systems is whether they are architected with a centralized server (CVS and SVN) or as a distributed system (Git).

While the difference between centralized and distributed version control may sound subtle, it does result in a profound difference in approach, philosophy and word definition. In short, terms like "commit" mean something different when viewed from a centralized approach (where it means "code is now available to all team members") versus a decentralized approach (where it means "code will now be tracked for changes but is still not available to all team members"). It is important to remember that, while many of the "words" are the same, the words have different meanings in context.

The individual project and development team's requirements and expertise should be considered when choosing a version control system. The choice can be influenced by preference for a centralized or distributed architecture. The development team's existing knowledge of a specific system can also influence the decision, as previous experience can reduce the learning curve.

The following section provides a brief overview and references to relevant documentation for Git, Subversion, and CVS.

- **Git**

  Git is a distributed revision control system with a peer-to-peer architecture. As opposed to centralized version control systems with a client-server model, each working copy of a Git project is a full-featured repository with complete revision history. This allows developers to work on and contribute to projects without the need to have access to the official repository — or even a network connection — since the complete working copy is stored locally. This feature is a fundamental difference as compared to centralized version control systems, where network access to the server is required to check out files and commit changes. In addition, because the repository is stored locally, Git operations are typically very fast, especially when compared to remote network access of CVS or Subversion.

  A Git repository is a place to store files that are under revision control, as well as additional data such as a complete history of changes or information about who made those changes and when. A typical Git repository stores a single project that, when publicly accessible, allows anyone to create a clone (which, if unfamiliar with Git, is roughly equivalent to checkout) with a complete revision history. Git can also be used to clone Subversion and CVS repositories, with the `git-svn` and `git-cvs` packages, respectively.

  Unlike in centralized revision control systems such as CVS or Subversion, developers usually do not commit their changes to one, central repository with Git. Instead, they either create a publicly accessible clone of their own repository, or send their changes to others via email as patches (which Git can generate). This workflow model is especially well-suited to large, distributed projects such as open-source software development with a global network of collaborators.

Table 4: Additional resources for Git

TYPE	DESCRIPTION
Installed Docs	`gittutorial(7), gitturotial-2(7)` – These manual pages provide an introduction to Git
	`gitcvs-migration(7)` – This manual page provides information for users migrating an existing project from CVS to Git
	`/usr/share/doc/git-version/everyday.html` – *Everyday Git With 20 Commands or So*
	`/usr/share/doc/git-version/user-manual.html` – The official *`Git User's Manual`*
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide, Chapter 3 Collaborating*
	http://git-scm.com/ – The official Git home page
	Pro Git – The online version of the Pro Git book provides a detailed description of Git, its concepts, and its usage

- **Apache Subversion (SVN)**

  Apache Subversion, commonly abbreviated as SVN, is a centralized version control system with a client-server architecture. It is a successor of the older Concurrent Versions System (CVS), preserving the same development model while addressing problems sometimes encountered with CVS. Subversion uses a centralized repository to store all changes by all users of the system.

  Because the centralized repository contains the entire file set and all files are in a single location, backups are easy. However, centralized systems represent a single point of failure, and developers cannot check out files or commit changes when access to the server is unavailable. A centralized system also does not, as easily, allow for developers to manage their own development branches or review new or modified code before it "takes over" the code base.

  Subversion uses a copy-modify-merge model (all developers copy, modify, and merge their changes) to control simultaneous edits to files in the repository. The version control system assists with merging, but ultimately it is the developer's responsibility to reconcile any conflicts that occur. While the copy-modify-merge model may seem messy, it allows for greater development concurrency than forcing developers to wait and work sequentially. If coming from a sequential version control system, consider how rarely two developers modify the same file, much less the same method. As a result, most of the time, merges are trivial; and when they are not, the team leadership can predict the situation (based on the features and bugs to be addressed) and plan for an extended merge session.

Table 5: Additional resources for Subversion

TYPE	DESCRIPTION
Installed Docs	`svn` help — This command provides detailed information on svn usage
	`svnadmin` help — This command provides detailed information on svnadmin usage
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide, Chapter 3 Collaborating*
	Official SVN website – http://subversion.apache.org/
	Official SVN book (includes best practices) – http://svnbook.red-bean.com/

- **Concurrent Versions System (CVS)**

  Concurrent Versions System (CVS) is a centralized version control system with a client-server architecture. CVS has been in use for many years, and represents a mature and stable technology.

  The CVS repository, a complete copy of all managed files and directories, is stored on a centralized server. Developers "check out" files into a working directory, make edits, and then "check in" their changes. By default, CVS supports unreserved checkouts, which allows multiple developers to check out and edit files simultaneously. When the file is checked back in, changes are merged and reconciled.

  Developers can also use CVS to create branches. Branches are useful for developers working on isolated or unstable changes. Later, those changes can be merged back into the main working directory or discarded. Branches are also useful for supporting various releases of a software development project.

Table 6: Additional resources for CVS

TYPE	DESCRIPTION
Installed Docs	cvs(1) – the online manual page for the cvs client program provides information on its usage.
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide, Chapter 3 Collaborating*
	Version Management with CVS — The official manual for CVS
	http://www.nongnu.org/cvs/ — The official CVS home page

Yum

Yum (Yellowdog Updater, Modified), the Red Hat update manager, is used to query for information about packages, fetch packages from repositories, install and uninstall packages, and update an entire system to the latest available packages. Through RPM, Yum performs automatic dependency resolution on packages that are updated, installed, or removed, simplifying administration. Yum can also be used to set up local repositories for download and installation onto other machines.

Table 7: Additional resources for Yum

TYPE	DESCRIPTION
Online Docs	*Red Hat Enterprise Linux 6 Deployment Guide, Chapter 5 Yum*
	http://yum.baseurl.org/wiki/Guides – Various guides on using Yum

Bugzilla

Red Hat Bugzilla is the Red Hat bug-tracking system that is used to submit and review defects that have been found in the Red Hat distributions. Before submitting a defect, developers should first query the database to ensure that the defect has not yet been reported. Developers can also use the advanced search capabilities to browse the bug database and search for existing bug reports, comments, or patches.

Note: While Red Hat Bugzilla can provide useful information to developers, it is not an avenue for technical assistance or support; it is simply a bug-tracking system. If you have an active subscription, visit the Red Hat Customer Portal for support assistance.

Table 8: Additional resources for Bugzilla

TYPE	DESCRIPTION
Online Docs	http://bugzilla.redhat.com – The Red Hat Bugzilla home page

Extra Packages for Enterprise Linux (EPEL)

Fedora Extra Packages for Enterprise Linux (EPEL) is a collection of unsupported developer tools and other components that are separately packaged for Red Hat Enterprise Linux 5 and 6. Created, maintained, and managed by a Fedora Special Interest Group, EPEL consists of a high-quality set of additional packages that developers created for Fedora but want to use on Red Hat Enterprise Linux. More information on EPEL can be found at http://fedoraproject.org/wiki/EPEL.

Languages

Red Hat Enterprise Linux supports application development in a wide variety of programming languages. The following sections provide information on Red Hat Enterprise Linux support for C/C++, Java, Python, Ruby, PHP, Perl, and Fortran.

C/C++

Red Hat Enterprise Linux provides support for the C and C++ programming languages with the GNU C and GNU C++ Standard Libraries, respectively. The Red Hat Enterprise Linux Developer Toolset provides an alternate GNU toolchain with newer versions of the tools for optional use by developers.

- C/C++ Libraries

  The `glibc` package contains the GNU C Library. This library defines all functions specified by the ISO C standard, POSIX specific features, some Unix derivatives, and GNU-specific extensions. The most important set of shared libraries in the GNU C Library are the standard C and math libraries.

The `libstdc++` package contains the GNU C++ Standard Library, which is an ongoing project to implement the ISO 14882 Standard C++ library. Installing the `libstdc++` package will provide just enough to satisfy link dependencies (that is, only shared library files). To make full use of all available libraries and header files for C++ development, you must install `libstdc++-devel` as well. The `libstdc++-devel` package also contains a GNU-specific implementation of the Standard Template Library (STL).

The `boost` package contains a large number of free, peer-reviewed. portable C++ source libraries for tasks such as time/date abstraction, serialization, thread creating, regular expressions, and many others.

- **Red Hat Enterprise Linux Developer Toolset**

  The Red Hat Enterprise Linux Developer Toolset provides current versions of the GNU Compiler Collection (GCC), GNU Debugger (GDB), and GNU Binutils. This toolset does not replace the default system toolchain provided with Red Hat Enterprise Linux. Rather, this parallel set of developer tools provides an alternative, newer version of those tools for optional use by developers.

- **ABI/API Compatibility**

  One of the core goals of Red Hat Enterprise Linux is to provide a stable, consistent runtime environment for custom application development. To support this goal, Red Hat Enterprise Linux maintains a core set of libraries where the Application Programming Interfaces (APIs) and Application Binary Interfaces (ABIs) are preserved for each architecture across major releases. Refer to the *Red Hat Enterprise Linux 6 Developer Guide*, section 4.2.5 Core Libraries for more details on the set of core libraries maintained by Red Hat Enterprise Linux.

Table 9: Additional resources for C/C++

TYPE	DESCRIPTION
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Chapter 4 Libraries and Runtime Support, Section 4.2.5 Core Libraries, and Section 4.3.1 The GNU C Library
	Red Hat Enterprise Linux Developer Toolset

Java

The Java programming language is a high-level, object-oriented programming language that is typically compiled to Java bytecode representation (rather than machine code) for high portability across diverse platforms. The `java-1.6.0-openjdk` package provides support for the Java programming language to Red Hat Enterprise Linux. The `java-1.6.0-openjdk-devel` package contains the `java` interpreter and the `javac` compiler, as well as the libraries and header files needed for developing Java extensions.

Red Hat Java developers can adopt the *JBoss Way*, a development approach – including a set of frameworks, tools, and architectural patterns – that is designed to help developers build code more efficiently. Besides providing the key technology components, the JBoss Way also provides additional resources such as tutorials, quick start guides, and videos. Exceptional tooling includes an integrated development environment (JBoss Developer Studio), a testing platform (Arquillian), and a core framework for rapid-application development (JBoss Forge), to name a few.

Table 10: Additional resources for Java

TYPE	DESCRIPTION
Installed Docs	`java(1)` – online man page for Java
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Section 4.3.8. Java
	JBoss Way – http://www.jboss.org/developer

Python

Python is an interpreted, high-level, object-oriented programming language used for a wide range of applications, including web and Internet development. The `python` package adds support for the Python programming language to Red Hat Enterprise Linux. This package provides the object and cached bytecode files needed to enable runtime support for basic Python programs. It also contains the `python` interpreter and the `pydoc` documentation tool. The `python-devel` package contains the libraries and header files needed for developing Python extensions.

Red Hat Enterprise Linux also ships with numerous Python-related packages. By convention, the names of these packages have a `python` prefix or suffix. Such packages are either library extensions or Python bindings to an existing library.

The `python` package uses the reference implementation of the Python programming language, sometimes known as *CPython*, as it is implemented in C. Within Red Hat Enterprise Linux 5 Python is CPython 2.4, whereas in Red Hat Enterprise Linux 6 it is CPython 2.6. You can see a description of the differences between versions 2.4 and 2.6 of the Python language and standard library within the What's New in Python 2.4 and What's New in Python 2.6 documents maintained by the upstream Python community.

Table 11: Additional resources for Python

TYPE	DESCRIPTION
Installed Docs	`python(1)` - online man page for python
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Section 4.3.7. Python
	Official Python website - http://python.org
	Official Python documentation website - http://docs.python.org (includes *What's New* documents)

Ruby

The `ruby` package provides the Ruby interpreter and provides support for the Ruby programming language. The `ruby-devel` package contains the libraries and header files needed for developing Ruby extensions.

Red Hat Enterprise Linux also ships with numerous Ruby-related packages. By convention, the name of these packages have a `ruby` or `rubygem` prefix or suffix. Such packages are either library extensions or Ruby bindings to an existing library.

Table 12: Additional resources for Ruby

TYPE	DESCRIPTION
Installed Docs	`ruby(1)` - online man page for Ruby
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Section 4.3.9. Ruby
	Official Ruby Language website - http://www.ruby-lang.org

PHP

PHP is an open-source general-purpose scripting language that is widely used for web development, especially server-side scripting. Red Hat provides both PHP for web development and the command line interface. Common PHP extensions can also be used, some through RPM and others through PECL and PEAR, both of which may also be installed in Red Hat Enterprise Linux. The `php-devel` package is also available for developers who wish to create new extensions. All of the PHP related packages start with `php` and can be easily discovered (using `yum search php`) and installed (using `yum install php-xxxxxx`).

Table 13: Additional resources for PHP

TYPE	DESCRIPTION
Installed Docs	`php(1)` - online man page for PHP and PHP Command Line
Online Docs	PHP Community - http://php.net/
	Pear: PHP Extension and Application Repository - http://pear.php.net/
	Pecl: PHP Extension Community Library - http://pecl.php.net/

Perl

The Perl programming language is a high-level, general-purpose, interpreted scripting language with powerful text-processing capabilities. The `perl` package adds support for Perl in Red Hat Enterprise Linux. This package provides Perl core modules, the Perl Language Interpreter, and the PerlDoc tool.

Red Hat also provides various Perl modules in package form; these packages are named with the `perl-` prefix. These modules provide stand-alone applications, language extensions, Perl libraries, and external library bindings.

Table 14: Additional resources for Perl

TYPE	DESCRIPTION
Installed Docs	`perl(1)` - online man page for Perl
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide,* Section 4.3.10. Perl
	Official Perl Programming Language website - http://www.perl.org

Fortran

Fortran is a general-purpose, procedural, imperative programming language that is commonly used for scientific and numeric computing. The GNU Compiler Collection (GCC) provides a set of tools for compiling a variety of programming languages, including Fortran. Fortran 2008 is now supported, while support for Fortran 2003 is extended.

Table 15: Additional resources for Fortran

TYPE	DESCRIPTION
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide,* Section 5.1 GNU Compiler Collection (GCC)

Cross-Cutting Concerns

Cross-cutting concerns, according to Wikipedia, "are aspects of a program which affect other concerns" (en.wikipedia.org/wiki/Cross-cutting_concern). In perhaps simpler terms, these are the components of a system that are generally implemented the same way throughout many modules and can lead to code duplication and difficult to manage interdependencies. Here we discuss two cross-cutting concerns, security and logging, and how Red Hat developers may wish to approach them.

Security

Security for Red Hat Enterprise Linux begins with a core feature known as Security-Enhanced Linux (SELinux). Co-developed by Red Hat and the U.S. Government's National Security Agency, SELinux adds Mandatory Access Control (MAC) to the Linux kernel, and is enabled by default in Red Hat Enterprise Linux. This security mechanism provides more fine-grained access control than traditional file permissions and is woven into all parts of the platform, including virtualization.

SELinux includes a set of kernel modifications and user-space tools that provide support for access control security policies. Administrators can create policy rules to explicitly define which subjects (users or programs) are allowed access to which objects (system resources such as files or devices). This security mechanism provides control over which activities are allowed for each user, process, and daemon. User programs and system services can be confined to the minimum level of privilege required, reducing the ability of programs to cause harm if compromised.

Table 16: Additional resources for Red Hat Enterprise Linux and SELinux

TYPE	DESCRIPTION
Online Docs	Red Hat Enterprise Linux documentation set at docs.redhat.com
	Red Hat Enterprise Linux website – www.redhat.com/products/enterprise-linux/
	Red Hat Enterprise Linux 6: *Security-Enhanced Linux User Guide*
	Official Security-Enhanced Linux (SELinux) project page – selinuxproject.org
	Step-by-step guide to building a new SELinux policy module

Logging

Logging for an application should generally utilize the components provided by the language or stack used by the application. However, the stack or language often just provides a wrapper around an operating system level facility for logging. As a result, this section does not discuss the myriad choices for logging within an application, but rather covers the best tools provided by Red Hat Enterprise Linux, which should be easy to leverage in any language.

Generally speaking, the recommended approach for logging is to leverage Red Hat Enterprise Linux's `rsyslog` capability. `rsyslog` can be easily configured for both local logging (during development) and remote logging (likely after production deployment). However, it is often worth "pretending" to use remote logging during development (with a remote configuration and a hosts file modification) to resolve any performance or configuration issues well in advance of deployment. During development, developers may also wish to implement Reliable Event Logging Protocol (RELP) and `stunnel` with `rsyslog` to support reliability and security, and further address potential issues. (Note: RELP and `stunnel` may not be available in releases earlier than Red Hat Enterprise Linux 6, and other options should be considered to provide those features.)

Another major consideration for logging is whether to use a database or flat files to capture the logs. Solid arguments exist for both, and the choice is application dependent. However, in either scenario, a developer should strive to create an asynchronous mechanism for sending the log message versus persisting it, to avoid performance issues.

Logging can sometimes mean *auditing*, and an application may have legal responsibilities for what actions are logged, how long they are persisted, the content they contain, and the reliability of the event capture. When considering an application's logging mechanism it is important to review your compliance requirements, such as HIPPA, SarBox, and PCI.

In recent years, a number of online tools for capturing log messages in the cloud and providing significant intelligence on the collected data have become available. As these tools can provide significant reduction in development effort (particularly as a function that is rarely captured in the initial project scope), they are worth considering. At a minimum, a developer should remember "consistency is key," and if one uses a consistent layout for logging, the data will be much easier to parse. Further, if a developer can model their logging on the Apache log format, many existing tools (open and proprietary) are available to parse the logs.

Table 17: Additional resources for Logging

TYPE	DESCRIPTION
Online Docs	RELP – http://www.librelp.com/relp.html
	`rsyslog` - http://www.rsyslog.com/
	The knowledgebase contains useful articles on `syslog` and `rsyslog` - http://access.redhat.com/

DEPLOYMENT

In the "Deployment Pillar" of the Red Hat Enterprise Linux Developer Program, we cover useful information for developers on packaging and releasing their software. Obviously, we could go into significant depth on this topic, which would be more appropriate for Release Engineers and System Administrators. Instead, this section of the document is intended to answer questions and provide pointers for the typical issues developers face when determining how to get their applications out to users.

Red Hat Enterprise Linux provides resources for the following software deployment tasks:

- Compiling and Building, with the GNU Compiler Collection and Autotools
- Debugging, with the GNU Debugger and Java Debugger
- Profiling, with Valgrind, OProfile, SystemTap, Eclipse-Callgraph, Performance Counters for Linux (PCL), and ftrace
- Packaging, with Red Hat Package Manager (RPM) and Software Collections
- Releasing, with CloudForms, OpenShift, and Public Cloud Providers
- Compatibility and Self-Certification

Compiling and Building

Red Hat Enterprise Linux includes a number of tools for compiling and building source code. The following sections describe the GNU Compiler Collection (GCC) and Autotools, and provide links to more information.

GNU Compiler Collection

The GNU Compiler Collection (GCC) is a set of open source tools for compiling a variety of programming languages, including C, C++, ObjectiveC, ObjectiveC++, Fortran, and Ada. GCC is highly portable, widely used, and produces highly optimized machine code.

GCC tools include various compilers, like `gcc` and `g++`, for compiling C and C++ programs, respectively. GCC also includes run-time libraries (like `libgcc`, `libstdc++`, `libgfortran`, and `libgomp`) and miscellaneous other utilities such as a quad-precision math library.

Autotools

The GNU build system, commonly referred to as Autotools, is a suite of command line utilities for building applications on various systems. The utilities create a script named `configure`. When executed, the `configure` script performs various tests on the system and creates top-level `Makefile` files that can be used to build the application.

The following three utilities form the basis of the GNU build system:

- The `autoscan` utility generates a preliminary input file such as `configure.scan`. This file can then be edited to create a final `configure.ac` file that is used by `autoconf`.
- The `autoconf` utility generates the configure script from an input file such as configure.ac.
- The `automake` utility automatically generates the `Makefile` for a project on a specific system.

The Autotools suite is also integrated into the Eclipse IDE via the Autotools plug-in. This plug-in provides an Eclipse graphical user interface for Autotools, which is suitable for most C/C++ projects.

Table 18: Additional resources for compiling and building

TYPE	DESCRIPTION
Installed Docs	`autoconf(1)`, `autoscan(1)`, `automake(1)` and `make(1)` – The manual pages provide information on the respective commands
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide,* Chapter 5 Compiling and Building
	gcc.gnu.org – The main site for GCC development
	GNU Autoconf Documentation – The official GNU Autoconf manual
	GNU Automake Documentation – The official GNU Automake manual
	GNU Make Documentation – The official GNU Make manual

Debugging

Locating and fixing bugs in software code is a critical part of the software development life cycle. Debugging tools enable developers to run a program and manage its execution. Common capabilities include:

- Inspect and modify memory within the code being debugged (for example, read and set variable values).
- Control the execution state by setting breakpoints, stepping through code line by line, and continuing operation.
- Detect when running code reaches a particular section, or when a specified variable is accessed.
- Inspect the call stack.

The GNU Debugger (GDB) and Java Debugger (JDB) are available in Red Hat Enterprise Linux as command line tools for debugging. In addition, the Eclipse C/C++ and Java development tools have excellent integration with both GDB and JDB. These Eclipse plug-ins take advantage of the latest features available in GDB and JDB, and enable developers to set breakpoints, step over certain lines of source code, and view current values of variables in the graphical user interface.

GNU Debugger

The GNU Debugger (GDB) is a command line tool that can be used to debug software programs. Fundamentally, like most debuggers, this source-level symbolic debugging tool enables developers to set breakpoints at any specific line of code, inspect memory within the code being debugged, modify values, control the execution state of the code, detect the execution of particular sections of code, and much more. GDB also supports setting conditional breakpoints, debugging forked processes, and debugging individual threads

The GNU Debugger works with a variety of programming languages including C, C++, Objective-C, Fortran, and Ada. See *Debugging with GDB: the GNU Source-Level Debugger for GDB* for a complete list of supported languages.

Programs compiled with the debugging (`-g`) flag can then be debugged using GDB. GDB can also be integrated into the Eclipse IDE, providing a graphical user interface for debugging.

Table 19: Additional resources for the GNU Debugger (GDB)

TYPE	DESCRIPTION
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide,* Chapter 6 Debugging
	GDB Documentation – The official GDB documentation
	Debugging with GDB: the GNU Source-Level Debugger for GDB – The GNU Debugger user manual

Java Debugger

The Java Debugger (JDB) is a command line debugger for Java classes that provides inspection and debugging of local and remote Java Virtual Machines. Like other symbolic debuggers, it enables developers to inspect memory within the code being debugged, control the execution state of the code, detect the execution of particular sections of code, and much more.

Programs compiled with the debugging (`-g`) flag can then be debugged using JDB. JDB can also be integrated into the Eclipse IDE, providing a graphical user interface for debugging.

Table 20: Additional resources for the Java Debugger (JDB)

TYPE	DESCRIPTION
Installed Docs	`jdb(1)` – The `jdb` manual page provides detailed information on its usage

Profiling

Throughout an application's development cycle, it is useful to collect information about how the application interacts with its intended environment. Software profiling, a form of dynamic program analysis, involves tracking details such as where memory is allocated and which part of the program consumes the most processor time. Developers often profile programs to help determine the source of errors, bugs, and performance bottlenecks in their applications.

Profiling collects data from the actual program execution. Thus, the quality of the data collected is influenced by the actual tasks being performed by the program. The tasks performed during profiling should therefore be representative of actual use.

This section describes some of the profiling tools available in Red Hat Enterprise Linux, and outlines when each should be used.

Valgrind

Valgrind is an instrumentation framework distributed in the `valgrind` package. This framework ships with a number of tools that can be used to profile applications in detail, aiding in the detection of memory errors and memory management problems, such as the use of uninitialized memory, improper allocation and freeing of memory, and the use of improper arguments in system calls.

Valgrind profiles an application by rewriting it and instrumenting the rewritten binary, so it is not necessary to recompile your application in order to profile it. However, this process also means that Valgrind can be 4 to 50 times slower than other profilers, especially when performing extremely detailed runs. It is therefore not suited to debugging time-specific issues, or kernel-space debugging.

Table 21: Additional resources for Valgrind

TYPE	DESCRIPTION
Installed Docs	`valgrind(1)` – The valgrind manual page named provides an overview of command line options supported by the Valgrind suite of tools
	`/usr/share/doc/valgrind-version/html/index.html` – Valgrind Documentation provides an in-depth description of Valgrind and its usage
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Chapter 7 Profiling

OProfile

OProfile is a low-overhead, system-wide profiler distributed in the `oprofile` package. It consists of a kernel driver and a daemon for collecting data, and a number of tools that turn that data into meaningful output. Its primary use is to determine which sections of code consume the greatest amount of CPU time, and why. OProfile uses the processor's performance monitoring hardware to retrieve information about the kernel and system executables, such as when memory is referenced, the number of last-level cache requests, and the number of hardware interrupts received. It can also be used to determine processor usage.

OProfile profiles an application "as is" (without adding instrumentation) by recording the details of every Nth event (where an event may be a cache miss or a branch instruction). This allows OProfile to consume fewer resources than Valgrind; however, it also means that its samples are not as precise. OProfile also requires root privileges to run.

Table 22: Additional resources for OProfile

TYPE	DESCRIPTION
Installed Docs	`oprofile(1)` – The `oprofile` manual pages provides an overview of OProfile and available tools
	`opcontrol(1)`, `opreport(1)`, `opannotate(1)`, `oparchive(1)`, and `opgprof(1)` – The various manual pages provide information on the respective tools
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Chapter 7 Profiling
	Red Hat Enterprise Linux 6 Deployment Guide, Chapter 22 OProfile

SystemTap

SystemTap is a tracing and probing tool that enables users to monitor the activities of a running computer system in detail, simplifying information gathering from a running Linux system. Developers can use SystemTap to monitor how their application behaves within the Linux system, and to help identify underlying causes of bug or performance issues.

SystemTap is designed to eliminate the need for the developer to go through the tedious instrument, recompile, install, and restart sequence required to collect data on the operation of the code. SystemTap provides a simple command line interface and scripting language for writing instrumentation. Developers write simple scripts to monitor kernel functions, system calls, and other events that occur in kernel-space. For example, they specify which system events (e.g., virtual file system reads or packet transmissions) should trigger a specified action (e.g., print or otherwise manipulate the data). SystemTap translates the script into a C program, which it compiles into a kernel module and then loads to perform the actual probe. With SystemTap, developers can examine the activities of a live Linux system without needing to recompile code to use the instrumentation, re-install things, or restart the code to get the data out.

The *SystemTap Beginners Guide* contains a good introduction to the SystemTap architecture and provides setup instructions for all kernel types. It also provides pre-written SystemTap scripts for monitoring activity, along with instructions on how to run them and analyze their output. The SystemTap Tapset Reference documents SystemTap's most useful and common tapset scripts, and provides guidelines on proper tapset development.

Table 23: Additional resources for SystemTap

TYPE	DESCRIPTION
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Chapter 7 Profiling.
	SystemTap Beginners Guide – An introduction to SystemTap
	SystemTap Tapset Reference – Describes SystemTap's most useful and common tapset scripts
	SystemTap documentation, including a Tutorial, Beginner's Guide, SystemTap Language Reference, and man pages
	Video: SystemTap Introduction – A brief demonstration of using the SystemTap tool

Eclipse-Callgraph

Red Hat Enterprise Linux includes the Eclipse-Callgraph plug-in, which provides the capability to graphically display a function trace of a program. This plug-in adds support for tracking selected functions or all functions in a program. Options include displaying function parent/child relationships, the number or times a function is called, the time spent in an instance of a function, or the time spent in all instances of a function. This information is useful to developers, enabling them to see the hierarchy of functions belonging to the executable as well as the time spent by various functions.

Callgraph can display the function information in a radial view or a tree view. An aggregate view displays all functions as boxes, with the size of the box relative to the function's execution time. The graph views in the Eclipse call graph are linked to the corresponding source code. This makes it easy for developers to easily navigate to the relevant portion of the code. Developers can also step through function calls in chronological order.

The Eclipse Callgraph plug-in uses SystemTap to perform the function tracing. As such, SystemTap must be installed along with the required kernel information packages.

Table 24: Additional resources for Eclipse Callgraph

TYPE	DESCRIPTION
Online Docs	http://wiki.eclipse.org/Linux_Tools_Project/Callgraph/User_Guide
	http://fedoraproject.org/wiki/Eclipse/CallGraph

Performance Counters for Linux (PCL)

Performance Counters for Linux (PCL) is a kernel-based subsystem that provides a framework for collecting and analyzing performance data. This subsystem counts hardware events such as instructions executed, cache-misses suffered, or branches mispredicted, as well as software events like page faults and timers in order to create a body of data that can be used by profiling applications.

Red Hat Enterprise Linux 6 uses the PCL kernel subsystem to collect data. The user-space `perf` profiling tool is then used to analyze the collected performance data.

Table 25: Additional resources for Performance Counters for Linux

TYPE	DESCRIPTION
Installed Docs	`perf-list(1)`, `perf-stat(1)`, `perf-record(1)`, `perf-report(1)` – The manual pages provide information on the respective commands
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Chapter 7 Profiling

ftrace

The `ftrace` framework is a built-in feature of the Red Hat Enterprise Linux kernel that provides tracing of the internal operations of the Linux kernel. The `ftrace` framework can be used by developers to see what's happening inside kernel space, and to debug performance issues that occur outside of user space.

A set of virtual files in the `debugfs` file system enable specific tracers. The `ftrace` function tracer outputs each function called in the kernel in real time. Other tracers are available to report and analyze kernel events, task switches, wake up latency, and other events. In addition, developers can add new tracers to analyze kernel events.

Table 26: Additional resources for ftrace

TYPE	DESCRIPTION
Installed Docs	`Ftrace` – Function Tracer: `/usr/share/doc/kernel-doc-version/Documentation/trace/ftrace.txt`
Online Docs	*Red Hat Enterprise Linux 6 Developer Guide*, Chapter 7 Profiling

Packaging

Red Hat Package Manager (RPM) is the standard utility for software package management on Red Hat Enterprise Linux. Software Collections, a newer utility that supports the concurrent installation of multiple versions of the same RPM packages on a system, is also supported.

Red Hat Package Manager (RPM)

The Red Hat Package Manager, RPM, is a package management system for installing, verifying, querying, updating, and uninstalling software packages. RPM makes it easier to distribute, manage, and update software created for Red Hat Enterprise Linux. Although many software vendors distribute their software via a conventional archive file (such as a tarball), there are several advantages in packaging software into an RPM. These advantages include:

- Each RPM package includes metadata that describes the package's components, version, release, size, project URL, installation instructions, and so on.
- Users can use a set of standard package management tools (for example Yum or PackageKit) to install, remove, and manage your software.
- With the standard package management tools, you can directly distribute your software, including any software updates, to your users.
- During upgrades, RPM handles configuration files carefully so that customizations are retained – something that can't be accomplished with regular `.tar.gz` files.

RPM enables developers to take software source code and package it into source and binary packages for end users. This process is quite simple and is driven from a single file and optional

patches. This clear delineation between pristine sources and patches along with build instructions eases the maintenance of the package as new versions of the software are released.

Table 27: Additional resources for RPM

TYPE	DESCRIPTION
Installed Docs	`rpm(1)` – The manual page provides detailed information about RPM parameters.
Online Docs	Red Hat Enterprise Linux 6 Deployment Guide, Appendix B: Package Management with RPM
	www.rpm.org – The official RPM website
	Maximum RPM – An online book that describes general RPM usage plus how to build your own RPMs

Software Collections

Software Collections enable the concurrent installation of multiple versions of the same RPM packages on a system. Distributed as a set of several components, Software Collections provide full functionality without overwriting system files. Furthermore, Software Collections require no changes to the existing RPM package manager.

Table 28: Additional resources for Software Collections

TYPE	DESCRIPTION
Online Docs	*Software Collections Guide*

Releasing

Red Hat Enterprise Linux provides deployment flexibility with support for major hardware architectures, virtualization solutions, and cloud computing. Releasing software as a traditional Red Hat Enterprise Linux application that runs locally on a native system remains an option. Red Hat also provides complete virtualization solutions that can provide operational flexibility and efficiency for deployed solutions. In addition, Red Hat also delivers leading open cloud solutions for deployment on private, hybrid, and public clouds. Businesses can deploy physical, virtual, and cloud computing within their datacenters, and access a consistent application environment across the different physical and virtual environments.

Virtualization is an integral part of the Red Hat Enterprise Linux platform. Virtualization capabilities are integrated into Red Hat Enterprise Linux and leverage the latest hardware virtualization features. Red Hat offers a complete virtualization solution with Red Hat Enterprise Virtualization for Servers and Red Hat Enterprise Virtualization Manager.

- **Red Hat Enterprise Virtualization for Servers**

  Red Hat Enterprise Virtualization provides a complete enterprise virtualization solution for server workloads with industry-leading scalability and performance. Easy manageability is provided with Red Hat Enterprise Virtualization Manager, a feature-rich management system for server virtualization that provides advanced capabilities including live migration, high availability, and storage management. Use of sVirt (a technology that integrates SELinux and virtualization) and the hardened Red Hat Enterprise Linux kernel provide a secure virtualization infrastructure.

- **Red Hat Enterprise Virtualization Manager**

  Red Hat Enterprise Virtualization Manager provides a centralized enterprise-grade virtualization management system with both a graphical administration console and programming interfaces. The graphical administration portal provides capabilities to easily manage virtual machines, desktops, storage, and clusters throughout the datacenter. Support for live migration allows for running virtual machines to be moved seamlessly from one host to another, for greater flexibility and availability. A system scheduler handles policies for load balancing, and supports a power saver mode which consolidates virtual machine loads onto fewer hosts during non-peak hours.

Table 29: Additional resources for Virtualization

TYPE	DESCRIPTION
Online Docs	Red Hat Enterprise Virtualization for Servers – http://www.redhat.com/products/virtualization/server/
	Red Hat Enterprise Virtualization Manager – http://www.redhat.com/virtualization/rhev/desktop/rhevm/

Red Hat offers CloudForms, OpenShift, and Certified Public Cloud Providers as options for open cloud computing solutions:

- **CloudForms**

  Red Hat CloudForms provides a complete framework for open hybrid cloud management. Organizations can use multiple virtual platforms and public cloud resources together in a private cloud. Infrastructure resources can be grouped into clouds and used for different functions, such as development, test, and production, and policies can be defined to control resource consumption. In addition to providing Infrastructure as a Service (IaaS), Red Hat CloudForms also provides robust application life cycle management tools to ease management.

- **OpenShift**

  OpenShift is Red Hat's free, auto-scaling Platform as a Service (PaaS) for applications. As an application platform in the cloud, OpenShift manages the stack so developers can focus on their code. OpenShift handles the infrastructure, middleware, and management so developers can focus on designing and coding in their favorite language. There's built-in support for Java, Ruby, Node.js, Python, PHP, and Perl, plus OpenShift is extensible and permits adding other languages. Built on open technologies, there's no lock-in. And the components that power OpenShift are all open source and available on GitHub as OpenShift Origin.

- **Public Cloud Providers**

  Red Hat offers the Certified Cloud Provider Program, a program that certifies that public cloud providers have tested Red Hat Enterprise Linux in their cloud and have support processes in place to quickly resolve problems should they occur. Rigorous testing and certification requirements help ensure the delivery of a safe, scalable, supported, and consistent environment for cloud deployments.

Table 30: Additional resources for Cloud Computing

TYPE	DESCRIPTION
Online Docs	Red Hat CloudForms – http://www.redhat.com/products/cloud-computing/cloudforms/
	Red Hat OpenShift – http://openshift.redhat.com
	Red Hat Certified Cloud Providers – http://www.redhat.com/solutions/cloud-computing/red-hat-cloud/find-public-cloud.html

Compatibility and Self-Certification

Developing applications for Red Hat Enterprise Linux provides developers with access to new software markets and a wide range of customers. To help developers bring products to these markets faster, Red Hat provides a series of recommended best practices for creating compatible applications. By following these practices, developers can reduce development and maintenance costs and ensure their applications are ready to deploy on Red Hat Enterprise Linux. The suggested practices help developers efficiently build competitive and capable applications, and help to ensure compatibility with subsequent Red Hat Enterprise Linux versions.

For more information please see the *Best Practices for Building Applications for Red Hat Enterprise Linux* document.

NEXT STEPS

Get started by acquiring a Red Hat Enterprise Linux subscription by contacting your local Red Hat representative, buying online, or joining a Red Hat Partner Program.

Your feedback is important to us. Send your questions and comments to RHELdevelop@redhat.com. For more information on Red Hat Development, visit http://www.redhat.com/developers/RHEL or post comments to the Red Hat Enterprise Developer Program online group

GLOSSARY

ABI

Application Binary Interface; describes the low-level interface between an application and the operating system or other services.

API

Application Programming Interface; a set of programming interfaces and data structures used by software components to communicate with each other.

Autotools

The GNU build system; a suite of command line utilities for building applications on various systems.

Bugzilla

A web-based bug tracking tool.

CDT

C/C++ Development Toolkit; a set of integrated tools for the Eclipse IDE.

CloudForms

A Red Hat offering that provides the hybrid cloud management, Infrastructure-as-a-Service (IaaS).

CVS

Concurrent Versions System, a centralized version control system with a client-server architecture.

DSO

Dynamic Shared Objects; object files that are used simultaneously, or shared, between multiple executing applications.

Eclipse

An Integrated Development Environment with an extensible plug-in architecture that supports multiple programming languages.

EPEL

Extra Packages for Enterprise Linux; a collection of developer tools that are created, maintained, and managed by a Fedora Special Interest Group.

ftrace

A built-in framework in the Red Hat Enterprise Linux kernel that provides tracing of the internal kernel operations.

GCC

The GNU Compiler Collection (GCC); a set of tools for compiling a variety of programming languages, including C, C++, ObjectiveC++, Fortran, and Ada.

GDB

The GNU Debugger; a symbolic debugger for C, C++, Objective-C and several other programming languages.

Git
A distributed revision control system with a peer-to-peer architecture.

IDE
Integrated Development Environment; an integrated software environment that provides tools for software development, such as a source code editor, built automation tools, and a debugger.

IaaS
Infrastructure as a Service; a type of cloud computing that provides the "machines" as a service (the "machines" may be physical or virtual), usually allowing support for various operating systems.

JDB
The Java Debugger; a symbolic debugger for programs written in the Java programming language.

JDT
Java Development Toolkit; a set of integrated tools for the Eclipse IDE.

MAC
Mandatory Access Control; restrictions by the operating system on the ability of users or programs to access system resources.

OpenShift
Red Hat's free, auto-scaling Platform as a Service (PaaS) for applications.

OProfile
A low-overhead, system-wide profiler that includes a kernel driver and daemon for collecting data and tools for analyzing and outputting the data.

PaaS
Platform as a Service; a type of cloud computing that provides the computing platform and software stack as a service (in other words, provides the whole development stack to deploy an application – think a full JBoss instance or Ruby + Rails + httpd + MySQL).

PCL
Performance Counters for Linux; a kernel-based subsystem that provides a framework for collecting and analyzing performance data.

PEAR
A framework and distribution system for reusable PHP components.

PECL
A repository for PHP Extensions.

PHP
An open-source general-purpose scripting language that is widely used for web development, especially server-side scripting.

Profiling
A form of dynamic program analysis that tracks information about a program during execution.

RELP
Reliable Event Logging Protocol; a client-server based logging protocol that provides reliable event logging over the network.

Revision Control System
See Version Control System.

RPM
Red Hat Package Manager; a package management system for installing, verifying, querying, updating, and uninstalling software packages.

SELinux
Security Enhanced Linux, an enhancement that provides greater access control of system resources.

stunnel
An open-source utility that provides secure TLS/SSL tunneling.

Subversion
Apache Subversion version control system, often abbreviated SVN/svn (because the command line tool is `svn`); a version control system with a client-server architecture.

sVirt
A technology included in Red Hat Enterprise Linux that integrates SELinux and virtualization to improve securing when using virtual machines.

SVN
See Subversion.

SystemTap
A tracing and probing tool that enables users to monitor the activities of a running computer system in detail.

Valgrind
An instrumentation framework that can be used to profile applications and help detect memory errors and memory management problems.

Version Control System
A set of tools for managing changes to a set of files and directories. Also called Revision Control System.

Yum
Yellowdog Updater, Modified; the Red Hat update manager.

ABOUT RED HAT

Red Hat was founded in 1993 and is headquartered in Raleigh, NC. Today, with more than 60 offices around the world, Red Hat is the largest publicly traded technology company fully committed to open source. That commitment has paid off over time, for us and our customers, proving the value of open source software and establishing a viable business model built around the open source way

SALES AND INQUIRIES

NORTH AMERICA
1-888-REDHAT1
www.redhat.com

EUROPE, MIDDLE EAST AND AFRICA
00800 7334 2835
www.europe.redhat.com
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
www.apac.redhat.com
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
www.latam.redhat.com
info-latam@redhat.com

Copyright © 2012 Red Hat, Inc. Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, and RHCE are trademarks of Red Hat, Inc., registered in the U.S. and other countries. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.