



Community Experience Distilled

Getting Started with Review Board

Analyze and improve your code using the collaborative code review tool, Review Board

Sandeep Rawat

[PACKT] open source*
PUBLISHING community experience distilled

www.allitebooks.com

Getting Started with Review Board

Analyze and improve your code using the collaborative code review tool, Review Board

Sandeep Rawat



BIRMINGHAM - MUMBAI

Getting Started with Review Board

Copyright © 2014 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: February 2014

Production Reference: 1140214

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78328-199-2

www.packtpub.com

Cover Image by Aniket Sawant (aniket_sawant_photography@hotmail.com)

Credits

Author

Sandeep Rawat

Project Coordinator

Ankita Goenka

Reviewers

Daniel Arbuckle

Prakash Jat

Dennis Mnuskin

Vaibhav Sehgal

Proofreader

Paul Hindle

Indexer

Rekha Nair

Acquisition Editors

Usha Iyer

Rubal Kaur

Graphics

Ronak Dhruv

Yuvraj Mannari

Content Development Editor

Shaon Basu

Production Coordinator

Aditi Gajjar Patel

Technical Editors

Ankita Jha

Dennis John

Cover Work

Aditi Gajjar Patel

Copy Editors

Insiya Morbiwala

Kirti Pai

Shambhavi Pai

About the Author

Sandeep Rawat is a passionate DevOps consultant who has extensive knowledge of build and release automation along with skills to manage a secure and scalable cloud infrastructure. He is an expert at streamlining build and release processes, and he has used them to achieve multiple, smooth build and releases per day at one of India's popular e-commerce website, Snapdeal. Currently, he is associated with Mettl, an online assessment solution for testing technical, aptitude, and psychometric skills. He works there as a DevOps consultant.

Sandeep is also an avid blogger. He blogs at <http://sandy4blogs.blogspot.in/>. In his spare time, he loves to work on various utilities; these can be found on his GitHub profile at <https://github.com/sandy724>.

His recent technology love is Puppet and Cloud Hosting.

I would like to thank my parents, my wife, and my son for always trusting me, standing by my side, and being my source of inspiration. I would also like to thank my friends Prakash and Vaibhav; it would have been difficult to finish this book without their valuable input; the book would have been an altogether different one.

I would like to thank Ashish, for giving me the privilege to write this book, and Rubal, for always giving me that special piece of advice and keeping me focused. A special note of thanks to Ankita, who kept on chasing me to make sure that I finish the book well within time; she is the reason that I was able to finish the book on time.

About the Reviewers

Daniel Arbuckle is a published researcher in the fields of robotics and nanotechnology as well as a professional Python programmer. He is the author of *Python Testing: Beginner's Guide*, Packt Publishing, and one of the authors of *Morphogenetic Engineering: Toward Programmable Complex Systems (Understanding Complex Systems)*, Springer-Verlag.

Prakash Jat is a system developer with nine years' experience in application development, mostly in Java/J2EE. Currently, he is a developer with the SoapUI team. SoapUI is a widely-used testing tool for SOAP, REST, HTTP, and JMS services/requests. It has more than one million users. His interests lie in the latest technologies and things happening around APIs and API testing.

Dennis Mnuskin is a highly driven professional with more than 15 years' software engineering experience. Over the span of his career, he has worked on a variety of products, from process control software and surveillance video recording and processing systems, to computer forensics and incident response tools. The overarching focus in all of these was always building high-performance, mission-critical data acquisition and storage server backends.

In his previous role, he was the software designer and technical project leader for an IP video recording platform. He was also a member of the Council of Architecture, where his responsibility involved setting long-term strategies for all software products developed by United Technologies Corporation's Climate, Controls, and Security Systems division. Currently, he is working at Carbon Black, a small (for now) startup company in the information security space, where he holds the position of Enterprise Server Engineering Lead.

These days, his focus is split between two major areas. The first one is purely technical: remaining hands-on with the latest tools, technologies, and programming languages; Scala being his latest one. The second area is in the realm of software team leadership and project management, where the goal of building a highly efficient, skilled, and motivated development team has its own unique set of challenges and rewards. He is a huge proponent of the Agile Software Craftsmanship and Lean Software Development movements and a strong advocate of practices such as continuous delivery, continuous process improvement, optimizing feedback loops, test-driven development, and eliminating waste.

Dennis can be contacted via e-mail at dennis.mnuskin@gmail.com.

Vaibhav Sehgal is a passionate software engineer who has contributed to the development of a wide array of software applications ranging from mobile apps to enterprise web applications.

www.PacktPub.com

Support files, eBooks, discount offers, and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
<hr/>	
Chapter 1: Introduction to Review Board	5
<hr/>	
Code review – power-charging your code	5
Code review best practices	6
Ways of performing code review	7
Pre-commit code review	8
Post-commit code review	9
Why Review Board?	10
What people are saying about Review Board	10
Features of Review Board	10
Publishing	10
Easy code review	11
Comments	11
Intuitive diff viewer	11
Great tracking	12
Integration with all major version control systems	12
Not limited to code files	12
Admin control	12
Summary	13
Chapter 2: Creating Review Requests	15
<hr/>	
Publishing a code review request	15
Generating a code diff	16
Publishing the generated code diff to Review Board	16
Publishing a review request with details	18
Tracking review requests	22
Publishing a code review request through the command line	22
Pre-commit review	23
Post-commit review	23

Review of specific files	24
Updating a code review request	24
Summary	26
Chapter 3: Reviewing Code Review Requests	27
Viewing a review request	27
Viewing the code review requests	28
Performing the code review	30
Managing issues	34
Summary	35
Chapter 4: The User Dashboard, Preferences, and Searching	37
The user dashboard	37
Incoming Reviews	38
Outgoing Reviews	38
Starred Reviews	38
All My Requests	38
Table columns	39
Preferences / My Account	40
Searching	42
Quick search	42
Full-text search	42
Summary	44
Chapter 5: Installing and Upgrading Review Board	45
Setting up Review Board	45
Review Board installation	48
Steps 1 and 2 – installing the Python setuptools	48
Step 3 – installing Patch	49
Step 4 – installing Review Board	49
Step 5 – installing MySQL database binding	49
Step 6 – the subversion source control component	49
Installing the Review Board site	50
Updating Apache config	52
Upgrading Review Board	53
Summary	54
Chapter 6: Admin Settings	55
SYSTEM SETTINGS	56
General	56
Authentication	58
E-mail	59
Diff Viewer	60
Logging	62

SSH	63
File Storage	64
SYSTEM INFORMATION	65
Summary	66
Chapter 7: Managing Users and Review Groups	67
Users	68
Review Groups	72
Summary	75
Chapter 8: Admin Dashboard	77
Repository administration	77
Activities	82
REVIEW BOARD ACTIVITY	82
USER ACTIVITY	83
RECENT ACTIONS	83
REVIEW BOARD NEWS	84
REPOSITORIES	84
REQUEST STATUSES	84
REVIEW GROUPS	85
Summary	85
Chapter 9: Advanced Tips and Tricks	87
Database	88
Extensions	90
Optimization	91
Hardware	91
Memcached	91
Database	91
Search indexing	92
Advanced commands	92
Summary	93
Index	95

Preface

Getting Started with Review Board is a book about the code review workflow in Review Board along with its administration aspect. It describes how to create a review request, perform a code review, and close the review request after incorporating the review comments in Review Board. It also covers the administration of Review Board, which includes installation, upgradation, and maintenance. It also talks about other tools such as post-review, which is also a part of the Review Board distribution.

What this book covers

Chapter 1, Introduction to Review Board, talks about Review Board components and the basic installation of Review Board on Linux.

Chapter 2, Creating Review Requests, describes the process of creating review requests in Review Board with different ways to create diffs and uploading them to Review Board.

Chapter 3, Reviewing Code Review Requests, describes the review process of a review request. It explains the concept of Ship It and various ways of providing review comments. It also explains how and when a review request can be closed.

Chapter 4, The User Dashboard, Preferences, and Searching, describes the user dashboard, which is helpful in tracking review requests and marking a review request as important for the user. It talks about user preferences and different ways of searching for users, review requests, reviewers, review groups, and so on.

Chapter 5, Installing and Upgrading Review Board, describes the detailed installation of Review Board and how to upgrade it.

Chapter 6, Admin Settings, describes the various settings that are part of the Admin Dashboard but applicable to the entire application.

Chapter 7, Managing Users and Review Groups, talks about how to manage users, set permissions, assign groups, and the side effects you might have when a user is deleted. It also talks about how to create and manage permissions for review groups.

Chapter 8, Admin Dashboard, talks about how to manage code repositories. It also describes some important aspects of the items included in the Admin Dashboard such as activities, requests, and review groups.

Chapter 9, Advanced Tips and Tricks, describes some advanced tips that are helpful for the admin to optimize the Review Board application and explains how to manage database entries directly through the Review Board site.

What you need for this book

You will need a Linux box on which you can install Review Board, preferably Ubuntu.

Who this book is for

This book is intended for people/teams who perform code reviews and who are using (or planning/evaluating to use) Review Board for code reviews in their team. This book could be helpful for someone who has never worked with Review Board but who wants to learn about it without having a running instance of Review Board. In addition, this book is also of great help to administrators, going into depth on the various aspects of administrating the Review Board server.

Conventions


In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.



Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "To enable Review Board to communicate with subversion, we need two packages to be installed on the system: `python-svn` and `subversion`."

Any command-line input or output is written as follows:

```
$ easy_install ReviewBoard
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "When the SSH key is already uploaded, the screen shows the following details: **Key type**, **Number of bits of encryption**, **Public fingerprint**, and **Public key**."

 Warnings or important notes appear in a box like this. 

 Tips and tricks appear like this. 

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the color images of this book

We also provide you a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from https://www.packtpub.com/sites/default/files/downloads/19920S_GraphicsBundle.pdf.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Introduction to Review Board

With this chapter, we will start our journey by getting acquainted with Review Board. First of all, we will look at pre-commit and post-commit code reviews, which are different ways of performing code reviews depending upon whether the review is done before or after pushing the updated code to the code repository. We will then briefly look at the various features of Review Board, such as creating and publishing review requests with code diff and integration with the **version control system (VCS)**, which will be covered in detail in the forthcoming chapters. As a code reviewer, you can view the code diff, add review comments, and create issues for a review request.



Every time you visit `search.yahoo.com`, you use code that has been reviewed on Review Board.

Code review – power-charging your code

How would you feel if the world's craftiest programmer went through every line of code that you wrote before it appeared on the user's screen (or maybe in a cold, dark machine running on a server farm on a remote island)? Good, right?

But everyone is not so lucky to have such a person go through their code! Yet, you can (and possibly do) have a good set of peers who can go through your code and help you improve it. They might not be as good as the world's craftiest programmer, but having your code pass through a different set of eyes and the feedback that it will generate will help you fine-tune your code, improving its correctness and quality significantly. This process is called **code review**.

Thus, code review helps you in identifying and resolving potential problems in your code that could otherwise be caught by a tester (bad for your ego), or even worse, appear in production (bad for your reputation).

Code review is beneficial for the following reasons:

- It is far more effective in detecting defects as compared to unit, functional, or integration testing
- It can play a significant role in helping you release your software on time (or even earlier)
- It can help you create very high-quality and reliable software
- It can help transform your team from being a mere development team to a quality development team

Guess what? The preceding points will also help you save a considerable amount of money while developing your software.

Some people think that code review is an overhead and slows down development. In reality though, many teams that perform code reviews have proven that it helps the organization to save a lot of time on bug fixing and speeds up the development, which leads to a better quality product and faster shipping.

Code review best practices

Now that you are familiar with what code review is, you will be itching to try to implement it. Hold on, aspiring code review ninja! You will be able to extract the maximum benefit out of code review only when you do it the right way. Keeping in mind the following points will be of help:

- Remember, small is beautiful. Code review works best when done in small chunks. Ideally, code review should not exceed 30 minutes, and the number of lines in a code review should not be more than 200.
- There should be a common code review checklist so that all of the code is evaluated with the same yardstick.
- It must be ensured that the outcome of each code review is sewn back into the code.
- Code review should target code, not people.
- Be egoless; *you are not your code*.
- Review findings should not be limited to a group of people. Rather, they should be visible to everyone in the organization. This would also help avoid duplication of similar coding mistakes across development teams.

Thus, code review is not a process of only identifying the coding problems; it also helps in manifesting a culture of writing high-quality code. To successfully implement the process of code review in your organization, you would need each and every programmer to be egoless.

Ways of performing code review

A few years ago in our team, the following was how we performed a code review:

1. Schedule a meeting for code review.
2. Provide the print of the code on paper in advance so that reviewers can go through the code and prepare their suggestions.
3. Discuss the review comments of each reviewer in the meeting, and record the suggested improvements as **Minutes of Meeting (MoM)**.
4. The author would incorporate the improvements and notify at least one of the reviewers to verify it.
5. The review was closed only once it was verified.

This kind of code review is known as a heavyweight code review or a formal code review. Although this process worked to get the job done, it had lots of overheads and consumed a lot of resources. It also had lengthy cycles of review feedback.

If you think that heavyweight code review is not suited for you because of its slowness, take it easy, aspiring code review ninja! Lightweight code review is here to rescue you! It typically requires lesser overhead than formal code inspections, though it can be equally (or even more) effective when done properly.

Because there is no formal defined rule of performing lightweight code review, you can devise your own methods of practicing it. The only thing you have to keep in mind is that it should not block people within your team.

A few of the common known ways of performing lightweight code review are as follows:

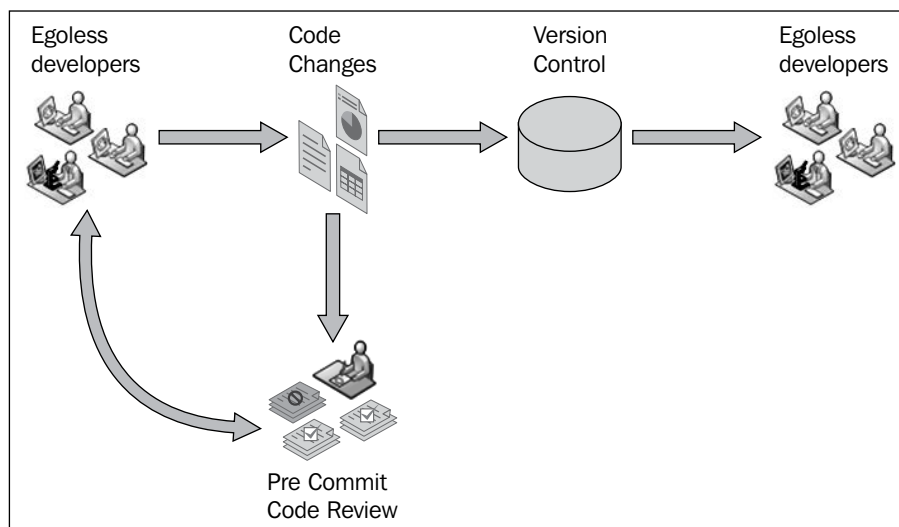
- **Over-the-shoulder:** As soon as you feel you have a piece of code ready, invite one or more peers over to your desk and walk them through the code to get it reviewed there itself.
- **E-mail pass-around:** Send your piece of code in an e-mail to your peers (two or three of them) to get their opinion.

- **Pair programming:** You and your developer friend can develop code together at the same workstation. This helps eliminate a lot of waste even before it appears. This is common practice in agile development methodologies.
- **Tool-assisted code review:** You'll use specialized tools designed for peer code review. One such tool is Review Board.

Another categorization of code review can be done on the basis of whether code review is happening before or after committing (pushing) the code to the code repository. The next section will talk about two types of code review (pre-commit and post-commit code review), the work flow that one can follow, and the pros and cons of following these code review processes.

Pre-commit code review

As the name suggests, pre-commit code review enforces that the code be committed to the code repository if and only if it has passed through the full code review cycle. The full code review cycle will ensure that the code meets the required quality standards and that all of the suggested changes have been incorporated. The pre-commit code review cycle is illustrated in the following diagram:

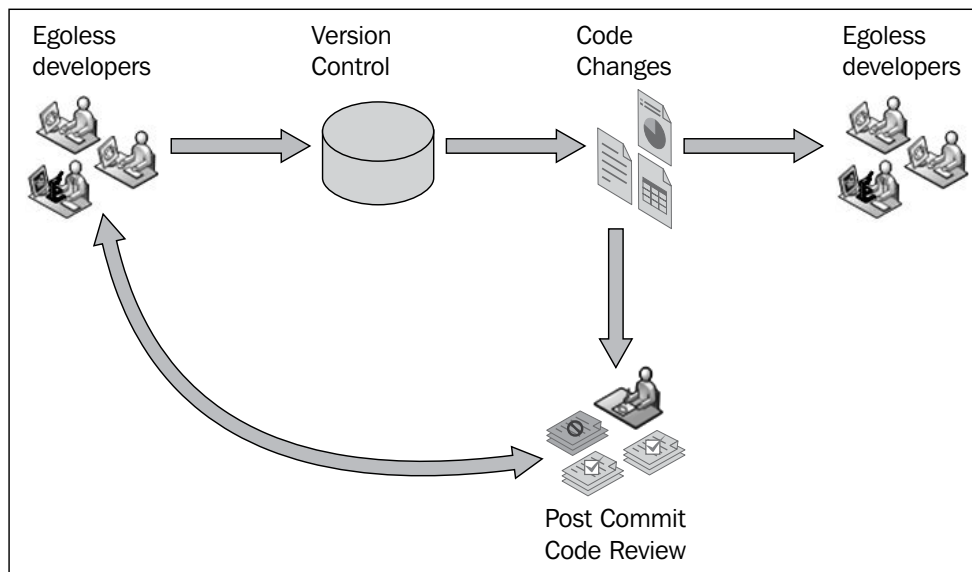


Pre-commit code review is a strict code review process as it acts as an ultimate funnel that doesn't allow a single line of code to be shared among team members unless it has been fully certified as quality code. The major challenge with this kind of review process is that it slows down the team as the multiple cycles of review become additional steps before the code reaches fellow developers.

To have a fairly successful pre-commit code review, the team members (or potential code reviewers) must always have "yes" as an answer to the question: do you have time for code review?

Post-commit code review

An alternate practice called post-commit review tries to tackle the challenge posed by pre-commit review by making the code review happen after the code becomes available to fellow developers. Thus, it doesn't prevent people from continuing with their work. The following diagram illustrates the post-commit code review cycle:



As Spiderman's uncle Ben said, "*With great power, comes great responsibility*", and the same applies to the post-commit review process. Because the developers have the power to make their code public even before it has passed the review process, the onus lies on them to ensure that they get the code reviewed later and incorporate the suggested changes.

On the cons' side, it might so happen that the other team members have already made the changes to the original code pushed by the author to the code repository by the time the author wants to incorporate the review comments. It may not be that big of a challenge though; it would depend on the team size.

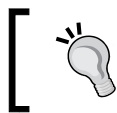
Why Review Board?

Review Board is a user-friendly code review tool. The following are some of its best characteristics:

- Open source
- Web-based
- Integrates well with most version control systems such as Subversion, Git, Mercurial, Perforce, and CVS
- Can be set up in any web server; it works very well with Apache and lighttpd
- Supports automatic posting of code reviews either through a REST Web API or Python script

What people are saying about Review Board

User feedback is always a good way to separate a great tool from the average ones. Review Board has been the most used tool for code review for many teams.



You can find a list of happy users on the Review Board site at <http://www.reviewboard.org/users/>.

Features of Review Board

As a first step towards our learning of Review Board, let's go through its features that make teams feel happy and confident about Review Board being the right tool for code review. A more detailed walkthrough of each feature will follow in the upcoming chapters.

Publishing

A basic requirement for any code review tool is that a user should be able to share the review request with the team. Review Board supports the creation of a review request through the web interface or command-line tools to push the review requests to Review Board. It requires bare minimum efforts to create and push the review request to Review Board.

Easy code review

The next most important thing after publishing the code review request is the ease with which a review can be performed. The smartest thing that can be done to make code reviewing easy is to highlight the differences introduced in the code, and that's exactly what Review Board does.

Comments

As a reviewer, you would want to add comments to the code under review to ask questions, provide suggestions, and appreciate a good implementation. Also, as an author of the code, you might want to further add comments to answer the questions asked by reviewers and explain why a particular suggestion should (or shouldn't) be incorporated. Review Board provides multiple ways to add these review comments on the code and also fulfills all of the following requirements:

- Comment on an individual line
- Comment on a group of lines or a selection of text
- Comment on an existing comment

Intuitive diff viewer

It becomes easier to review the code if the differences in the code, before and after the changes, are visible and highlighted. Review Board comes with a very intuitive diff viewer to highlight the differences with the following features:

- A side-by-side diff view is presented to see what has been changed in a particular code commit
- Code addition, modification, and deletion are highlighted in different colors for quick identification
- The changed content of a line is highlighted to show exactly what has changed in the line
- You can choose to perform a single or multiple diffs

Great tracking

One of the major parameters that teams have to weigh a code review tool is how easy it is to track the review requests. If code review is done manually, it can consume a lot of time tracking all of the review requests and their progress. Also, you might have to keep chasing people to get the review done. With Review Board, one can track the review requests just as user stories and bugs are tracked in other tools. Additionally, Review Board has the following features:

- It starts tracking the review request as soon as it is posted
- An e-mail is sent to all of the reviewers upon the publishing of a review request
- The user dashboard in Review Board presents a list of pending reviews that includes submitted and "to be done" reviews

Integration with all major version control systems

Many code review tools are tied to a specific version control system. This is not the case with Review Board. Review Board can be integrated with Git, Subversion (SVN), Perforce, and Mercurial in a seamless manner.

Not limited to code files

Another good feature of Review Board is that you are not limited to reviewing only the code files. You can upload many other types of files and file formats too.

Admin control

Review Board makes life easy not only for its users, but also for its administrators. Review Board makes it possible to manage almost everything through its web GUI. The following is the list of features that makes an administrator's life easier:

- A very intuitive web GUI to configure users and repositories
- All of the system-related statistics (for example, usage and server health parameters) are present on the dashboard
- It is possible to manage the database from the web GUI

Summary

With the setup of Review Board, we come to the end of our first chapter. In this chapter, we learned about pre-commit and post-commit code reviews, various features of Review Board, and how those features help both the author and the code reviewers to do their work efficiently and effectively.

The next chapter will talk about how you will work with Review Board as an author of code who will be publishing the code to Review Board for code review.

2

Creating Review Requests

In this chapter, we will go through the workflow for the code author—how you can work as an author of code to create and share code review requests with your team. We will discuss the following two ways of posting code review requests:

- Publishing a code review request through the web application of Review Board
- Publishing a review request through the command-line tool that comes bundled with Review Board

We will use Git and Subversion (often abbreviated as SVN) as the two version control systems in our discussion. There are no major differences between different version control systems. So, if you want to use a version control system different from these, please visit the Review Board website to find out whether or not Review Board supports it.

Publishing a code review request

Let's get down to action; publishing a code review request is a simple, two-step process that can be executed by doing the following:

1. Generating a code diff.
2. Publishing the generated code diff to Review Board.

Generating a code diff

A **code diff** is the difference between two versions of a piece of code. These versions might exist on your local machine or someplace else, such as the version control system. It's the delta of the code (the difference in the two versions of code) that you have authored and want to get reviewed. Generating a code diff is a standard procedure in the code review process, but the command for creating a code diff varies across version control systems.

Let's first see how we can generate a code diff in SVN:

```
svn diff -r54:53
```

In the preceding command, we ask SVN to generate a code diff between the SVN commit revision numbers 54 and 53. You will see the code diff as an output of this command. Because we have to attach the code diff to a code review request, let's save the generated code diff in a code diff file using the following command:

```
svn diff -r54:53 > svn_54_53.diff
```

The preceding command generates a code diff and writes the generated code diff in a file named `svn_54_53.diff`. You can have any name for your code diff file; having such a filename gives you a large advantage. As a general practice, we usually name the diff file as `svn_<parent_version>_<child_version>.diff`. The advantage of having such a filename is that you can easily figure out that it is an SVN code diff file between the SVN revision numbers 54 and 53.

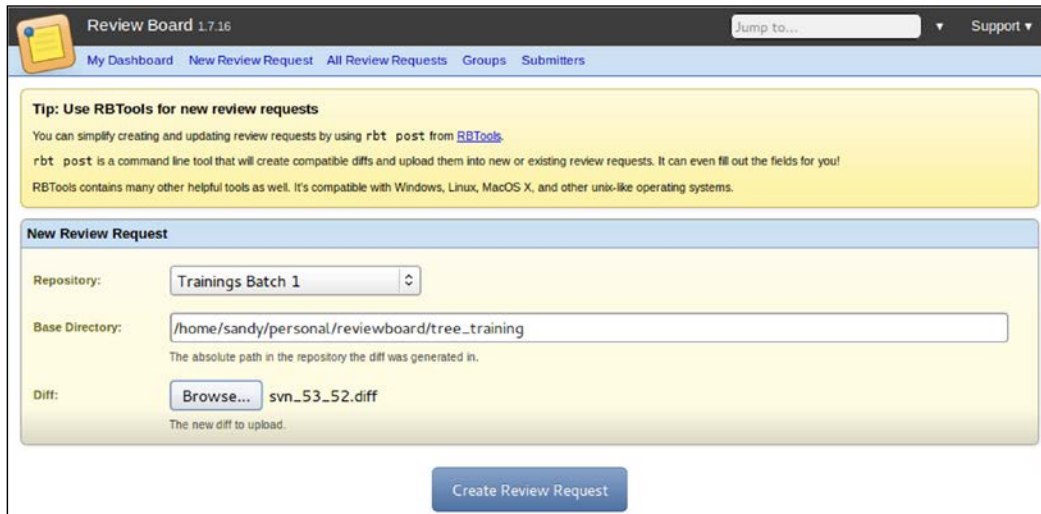
Similarly, you can generate a code diff file for other version control systems as well. The diff generation command will vary for each one of them. For example, we use the following command to generate a diff in Git:

```
git diff {commit 1 hash code} {commit 2 hash code} > git_commit1_commit2.diff
```

Publishing the generated code diff to Review Board

You have authored some code changes and prepared a code diff. Now you want to publish it to Review Board so as to share it with the team for review. Let us create and publish a code review request with the code diff that we generated.

When you log in to your Review Board site, you see different tabs at the top navigation bar. **New Review Request** is one of them. Clicking on this tab will lead you to the following screen:



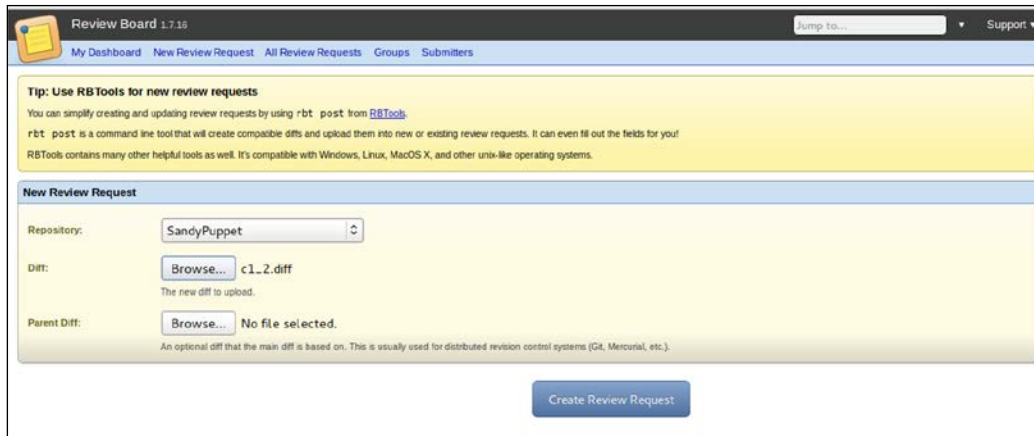
The screenshot shows the 'New Review Request' form in the Review Board 1.7.16 interface. At the top, there is a navigation bar with tabs for 'My Dashboard', 'New Review Request', 'All Review Requests', 'Groups', and 'Submitters'. A 'Jump to...' search box and a 'Support' link are also visible. Below the navigation bar is a yellow tip box that reads: 'Tip: Use RBTools for new review requests. You can simplify creating and updating review requests by using `rbt post` from [RBTools](#). `rbt post` is a command line tool that will create compatible diffs and upload them into new or existing review requests. It can even fill out the fields for you! RBTools contains many other helpful tools as well. It's compatible with Windows, Linux, MacOS X, and other unix-like operating systems.' The main form area is titled 'New Review Request' and contains three fields: 'Repository:' with a dropdown menu showing 'Trainings Batch 1'; 'Base Directory:' with a text input field containing '/home/sandy/personal/reviewboard/tree_training' and a subtext 'The absolute path in the repository the diff was generated in.'; and 'Diff:' with a 'Browse...' button and the filename 'svn_53_52.diff' and a subtext 'The new diff to upload.' At the bottom of the form is a blue 'Create Review Request' button.

You have to provide the following details in the case of an SVN repository:

- **Repository:** This dropdown lets you choose the repository on which you are working. It lists all of the repositories added by the admin. We will go through the admin section in *Chapter 7, Managing Users and Review Groups*.
- **Base Directory:** This shows the absolute path of the directory for which you have generated the SVN diff.
- **Diff:** This refers to the diff file that we generated earlier.

Creating Review Requests

As mentioned earlier, the process of uploading the code diff varies from VCS to VCS; that is, the fields required for creating the new review request will change as per the selected repository (if the repositories are of a different type of VCS). Let's see what the screen for creating a review request looks like for Git:



The screenshot shows the 'Review Board 1.7.15' interface. At the top, there is a navigation bar with 'My Dashboard', 'New Review Request', 'All Review Requests', 'Groups', and 'Submitters'. A 'Jump to...' search box and a 'Support' link are also present. Below the navigation bar is a yellow tip box that reads: 'Tip: Use RBTools for new review requests. You can simplify creating and updating review requests by using `rbt post` from [RBTools](#). `rbt post` is a command line tool that will create compatible diffs and upload them into new or existing review requests. It can even fill out the fields for you! RBTools contains many other helpful tools as well. It's compatible with Windows, Linux, MacOS X, and other unix-like operating systems.'

The main form is titled 'New Review Request' and contains the following fields:

- Repository:** A dropdown menu with 'SandyPuppet' selected.
- Diff:** A 'Browse...' button followed by the text 'c1.2.diff'. Below it, a small note says 'The new diff to upload.'
- Parent Diff:** A 'Browse...' button followed by the text 'No file selected.'. Below it, a small note says 'An optional diff that the main diff is based on. This is usually used for distributed revision control systems (Git, Mercurial, etc.)'

At the bottom of the form is a blue button labeled 'Create Review Request'.

In the case of Git, the diff option remains the same; you don't have to provide a base directory in this case (unlike SVN). There is an additional field named **Parent Diff** that is a common field for distributed version control systems such as Git or Mercurial.

As a standard practice, you should make sure that all of the branches in your Git repository are made available in the Review Board server. If that is not the case, you will not be able to post the code diff for a branch that is not available directly because Review Board will not be able to show the code diff. In this case, you have to use the **Parent Diff** feature of Review Board.

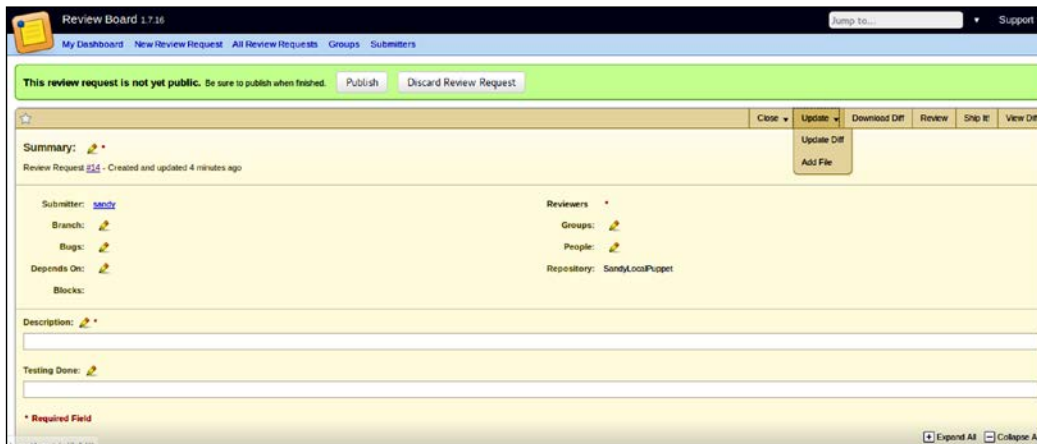
Publishing a review request with details

When you click on the **Create Review Request** button, the code review request is persisted in the Review Board system and is ready to be published as displayed in the next screenshot. Now, you can publish the request for review, make updates to it before publishing it for review, or discard it if you want to make further code changes before presenting them for review with a new diff.



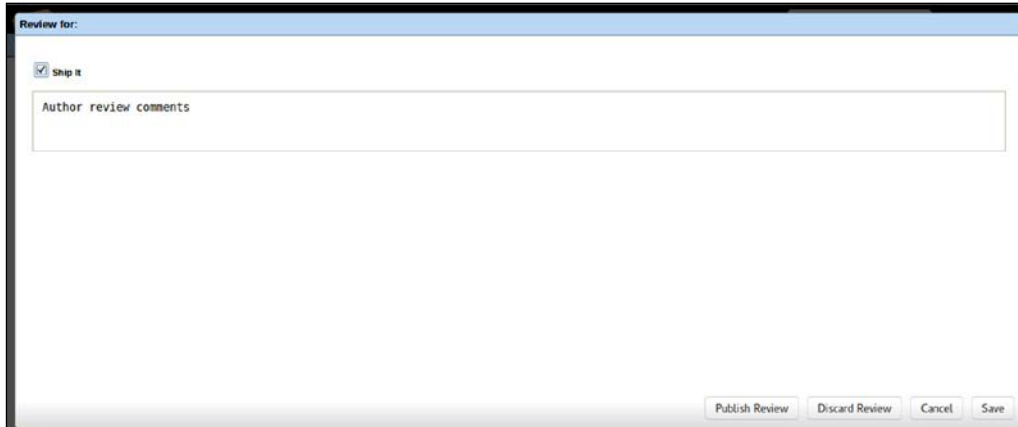
The following operations are available on the created review request:

- **Close:** In case you don't want to carry on with the code review request, you can close it by marking it as either **Submitted** or **Discarded** as shown in the following screenshot:



- **Update:** You can upload a new diff and add additional relevant files through the two links available under the **Update** menu.

- **Download Diff:** You can use the **Download Diff** option if at any point in time you want to see the contents of the diff that you uploaded in the first step of creating a review request.
- **Review:** You can review your code yourself and provide review comments as an initial starting point for fellow reviewers. Also, while posting review comments, you can mark the review operation as **Ship It!**:



The screenshot shows a dialog box titled "Review for:". Inside the dialog, there is a checkbox labeled "Ship it" which is checked. Below the checkbox is a text area labeled "Author review comments". At the bottom right of the dialog, there are four buttons: "Publish Review", "Discard Review", "Cancel", and "Save".

- **Ship It!:** This can be thought of as the approval for the code changes under review. As a code review request can be raised to more than one fellow team member, the number of **Ship it!** responses given to a review request can give an idea about the number of team members who approve of the code changes.

By now, you would have created a review request and updated it, but you haven't shared it with the team yet.

In order to share this review request with the team, click on the **Publish** button, and you will see the following screen:

The screenshot shows the Review Board interface for creating a new review request. The form is titled "New Review Request" and includes a navigation bar with "My Dashboard", "New Review Request", "All Review Requests", "Groups", and "Submitters". A green banner at the top states "This review request is not yet public. Be sure to publish when finished." with "Publish" and "Discard Review Request" buttons. The form fields are as follows:

- Summary:** Please look at the design flaws
- Submitter:** sandy
- Branch:** master
- Bugs:** DEVOPS-1
- Depends On:** DEVOPS-2
- Blocks:**
- Reviewers:**
 - Groups:** lead_reviewer,
 - People:** prakash, sandy, admin,
 - Repository:** SandyLocalPuppet
- Description:** This feature is for generating an interactive UI.

There are certain mandatory and optional fields that need to be filled before publishing the request. Let's discuss these fields one by one:

- Summary:** It is always good practice to add a summary for your code review request so that the reviewers don't have to chase you (the author). In addition to **Summary**, there are a couple of other fields as well that can be used to provide more insight into code review requests, such as **Branch**, **Bugs** (if the changes were made to fix a bug), and **Depends On** (if the request depends on other review requests).
- People:** In this field, you can provide a comma-separated list of all of the fellow developers whom you want to review your code. This option should be preferred over the **Group** option only when you have random reviewers in your team and the reviewers vary with each review request. The good thing with Review Board is that it gives you an autosuggestion textbox that has a list of all of the people who can be assigned for the code review. Pressing the *Tab* key will autocomplete the suggested name. This list is set up through the admin account. We will discuss it in *Chapter 7, Managing Users and Review Groups*.
- Group:** If the Review Board administrator has created groups, it would be a better option to choose a group instead of providing a long list of people in the **People** textbox. The **Admin** field also takes a comma-separated list of groups, and it is an autosuggestion field as well.

Once you have filled all of the details, click on the **OK** button to publish the code review request.

Tracking review requests

Review Board provides all of the required options to track a review request through the dashboard:



All of the code review requests created by you can be viewed by clicking on the **Outgoing Reviews** option in the left-hand side menu. We will talk about **Incoming Reviews** when we discuss the code reviewer's operations in the up coming chapters. The **All My Requests** link gives you a list of all of the reviews, be it the raised code review requests or the assigned code review requests.

The right-hand side box gives a summarized list of reviews. The summarized review details, by default, present you with the summary of the code review request, the author who created the review request, the date on which the review request was posted, and the date on which the last action was taken on this request. The columns on the right-hand side can be configured by clicking on the **Edit** icon and adding and removing fields as per the need. You can also sort these review request columns by clicking on the column headers.

Publishing a code review request through the command line

If you love to work with command-line tools and don't like to use the web UI of Review Board, you can use `post-review` to generate or update a code review request. `Post-review` works equally well on all three major operating systems: Windows, Linux, and Mac.

The `post-review` command can be used in a number of ways depending on your requirement or organizational setup. Installing `post-review` is pretty simple; you need to have Python and the `setuptools` already installed in your system, then you can use `easy_install` to install `post-review`:

```
easy_install -U setuptools
```



Linux comes preinstalled with Python. You just have to install `setuptools`.
Use the following command for Debian/Ubuntu or other Debian-based distributions:
apt-get install python-setuptools
For Fedora 8 and higher, use the following command:
yum install -y python-setuptools-devel.noarch
For Red Hat Enterprise, CentOS, Fedora 7 and earlier, or other distributions based on Red Hat, use the following command:
yum install python-setuptools

Pre-commit review

Some organizations enforce that only reviewed code should go into the version control system, and `post-review` helps in doing this by default, as follows:

```
$ post-review
Review request #23 posted.
http://localhost/r/23/
```

The preceding command posts all of the uncommitted code to Review Board and gives you details about the review request thus created. Then, you can open the Review Board site and publish the review request.

Post-commit review

Let's assume you want to create a review request for code that is already committed. To do this, you need to provide the revision number of your commit using the following steps:

1. Create a review request with the diff between the revisions 50 and 52:

```
$ post-review --revision-range=50:52
Review request #24 posted.
http://localhost/r/24/
```

2. Create a review request on the basis of a commit done on the revision number 52:

```
$ post-review --revision-range=52
Review request #25 posted.
http://localhost/r/25/
```

Review of specific files

You can also use `post-review` to upload a list of files. For this, you have to provide a space-separated list of the files to be reviewed:

```
$ post-review trunk/src/Calculator.java trunk/src/CalculatorTest.java
Review request #26 posted.
http://localhost/r/26/
```

Updating a code review request

Let's say you have posted a review request, and at a later point in time, you added more code. If you want to add some more code in the code review request, you can use the `-r` option in `post-review`:

```
$ post-review -r 26 trunk/src/Calculator.java
Review request #26 posted.
http://localhost/r/26/
```

```
$ post-review -r 26
Review request #26 posted.
http://localhost/r/26/
```

`Post-review` is not just limited to creating code review requests; you can do a lot of customization with it too. In fact, you can even publish review requests by providing suitable arguments to the `post-review` command. Consider the following command:

```
post-review --help
```

The preceding command will list out all of the available arguments that you can pass with your `post-commit` command. Let's look at some of the most important ones:

- `-p/--publish`: You can use `-p` if you want to publish the created code review request:

```
$ post-review -r 26 -p
```

- `-o/--open`: Once a code review request is created, this command opens a browser so that you can view the request:

```
post-review -r 26 -o
```
- `--server`: This command is used to specify the address of the Review Board server. It is usually used to override the default Review Board server configured on your system:

```
post-review -r 26 --server=http://localhost
```
- `--target-groups`: This displays a comma-separated list of groups available in the system to which you want to assign the code review request:

```
$ post-review -r 26 --target-groups=lead-reviewer
Error getting review request 26: One or more fields had errors
(HTTP 400, API Error 105)
target_groups: lead-reviewer

$ post-review -r 26 -target-groups=lead_reviewer
```

This will generate an error if you provide a group that doesn't exist in the Review Board setup. As you can see, when we gave a nonexisting group name, the request submission failed.
- `--target-people`: Similar to the `target-groups` option, you can use this option to provide the list of people whom you would like to be reviewers of your code. But, you have to ensure that these users exist in the system:

```
$ post-review -r 26 -target-people=sandy,prakash
```
- `--username`: This provides the username credential.
- `--password`: This provides the password.

You can also create a `.reviewboardrc` file that will contain the listed properties. As a recommended practice, at least add the following properties so that you don't have to provide them every time:

- `REVIEWBOARD_URL`
- `USERNAME`
- `PASSWORD`

Summary

So now you have learned how to create and publish the code review request so that a team member can review the code diff you authored. We have talked about the Web UI of Review Board and the post-review command-line tool, and how to use them to create and publish the code review request. In the next chapters, we will look at how to work with the code review request as a code reviewer.

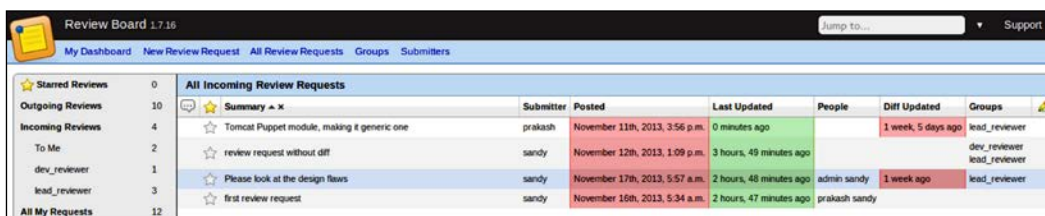
3

Reviewing Code Review Requests

So far, we have learned about Review Board and how to use it as a code author. In this chapter, we will talk about using Review Board as a code reviewer. We will look at how a reviewer comes to know about a review request that has been assigned to them and how Review Board helps a user in reviewing the assigned code review request. We will also discuss a great feature of Review Board – creating and managing issues with a review request.

Viewing a review request

As a code reviewer, you'd always like to know about a new review request assigned to you (the sooner the better). If e-mail notification is enabled, all of the proposed code reviewers will get a code review e-mail. In case of the Review Board UI, the dashboard (home page) will list out various types of review requests such as **Starred Reviews**, **Outgoing Reviews**, **Incoming Reviews**, and a list of all review requests, as shown in the following screenshot. **Incoming Reviews** contain the new review requests assigned to you. They are further subcategorized into various categories depending on the review groups of which the user is a member, as shown in the following screenshot:



The screenshot shows the Review Board 1.7.16 dashboard. The main content area is titled 'All Incoming Review Requests' and displays a table of review requests. The table has columns for Submitter, Posted, Last Updated, People, Diff Updated, and Groups. The 'Diff Updated' column uses color coding: green for '0 minutes ago', red for '1 week, 5 days ago', and blue for '1 week ago'.

Starred Reviews	0	All Incoming Review Requests						
Outgoing Reviews	10	Summary x	Submitter	Posted	Last Updated	People	Diff Updated	Groups
Incoming Reviews	4	Tomcat Puppet module, making it generic one	prakash	November 11th, 2013, 3:56 p.m.	0 minutes ago		1 week, 5 days ago	lead_reviewer
To Me	2	review request without diff	sandy	November 12th, 2013, 1:09 p.m.	3 hours, 49 minutes ago			dev_reviewer lead_reviewer
dev_reviewer	1	Please look at the design flaws	sandy	November 17th, 2013, 5:57 a.m.	2 hours, 48 minutes ago	admin sandy	1 week ago	lead_reviewer
lead_reviewer	3	first review request	sandy	November 16th, 2013, 5:34 a.m.	2 hours, 47 minutes ago	prakash sandy		
All My Requests	12							

As you can see in the preceding screenshot, the left-hand navigation tree shows the review requests grouped by the type and the corresponding number of requests assigned to the user. On this screen, we can see four incoming requests. The distribution of these review requests is shown below the **Incoming Reviews** link; **dev_reviewer** (1) and **lead_reviewer** (3) are the two review groups to which a reviewer may belong. As you can see in the **Groups** column, the first, third, and fourth review requests are assigned to the **lead_reviewer** group and one review request is assigned to the **dev_reviewer** group. Similarly, if you notice the **People** column, the third and fourth review requests are directly assigned to the user **sandy** and match with the **To Me** subsection of the **Incoming Reviews** link in the left-hand navigation tree.

Viewing the code review requests

Once you know which review request you need to check, you can start the code review. The code review can either be started by clicking on the **Summary** cell of any of the review requests, or, if you have the link of the code review request, you can directly open that link (for example, <http://localhost/r/3/>).

The reviewer will be presented with the following screen on opening a review request:



The topmost bar (**Top Most Bar**, as labeled in the preceding screenshot) presents links to various actions that a reviewer can perform on the review request. The first block of the review request (**First Block**) shows the summary, the creation, and last modification date along with the unique identifier of the review request.

The second block of the review request (**Second Block**) gives the reviewer more insights about it. The left-hand section provides information about the dependency details; for example, if this review request is associated with some bugs through the **Bugs** and **Blocks** fields. The **Depends On** field provides a link to other reviews to which the reviewer can refer to. The right-hand section has three fields: **Groups**, **People**, and **Repository**. These fields list the groups or people to whom the request is assigned for review as links.

The third block (**Third Block**) provides the descriptive details about a review request such as the description added by the author of the code and the comments about **Testing Done**. A reviewer usually refers to the **Testing Done** field to see if the author has provided the details about the testing strategy for the code under review; detailed information in the **Testing Done** field gives a lot of insights to a reviewer before starting the code review.

The last section of the review request is **Issue Summary** (as shown in the following screenshot). In this section, a reviewer can view the various issues associated to the review request. Creating and managing issues with a review request is a great feature of Review Board. We will talk more about these issues in the upcoming sections.

Issue Summary:

Total: 3 Open: 1 Resolved: 1 Dropped: 1

Status: All From: All

Description	From	Last Updated	Status
Good to see that we are moving to generic tomcat setup	sandy	Nov 24, 2013, 12:20 p.m.	Resolved
what is this change for	sandy	Nov 24, 2013, 12:20 p.m.	Dropped
We should not have ckeeditor in generic tomcat setup	sandy	Nov 11, 2013, 4 p.m.	Open

Expand All Collapse All

+ sandy Posted 1 week, 5 days ago (Nov 11, 2013, 4 p.m.)

- Review request changed Updated 6 hours, 41 minutes ago (Nov 24, 2013, 6:07 a.m.)

- status changed from pending to submitted

+ Review request changed Updated 6 hours, 41 minutes ago (Nov 24, 2013, 6:07 a.m.)

+ Review request changed Updated 17 minutes ago (Nov 24, 2013, 12:31 p.m.)

+ Review request changed Updated 12 minutes ago (Nov 24, 2013, 12:36 p.m.)

In addition to showing the review request details, the review request page also gives a chronological list of all of the actions taken on a review request, such as updating a review request or review comments by a reviewer. By default, all the actions are collapsed. You can click on the + icon to view the details of the action taken. Also, in the top-right corner, there are two buttons: **Expand All** and **Collapse All**, to expand and collapse the action details respectively. As you can see in the preceding screenshot, the first action is in the expanded state, and it tells about the change of status of the review request from **pending** to **submitted**.

Performing the code review

Now comes the most important part of the whole code review cycle: reviewing the code. There are five different actions that can be taken on a review request, and they are labeled in the following screenshot:



The following list explains each action. The number corresponds to the label given to them in the screenshot:

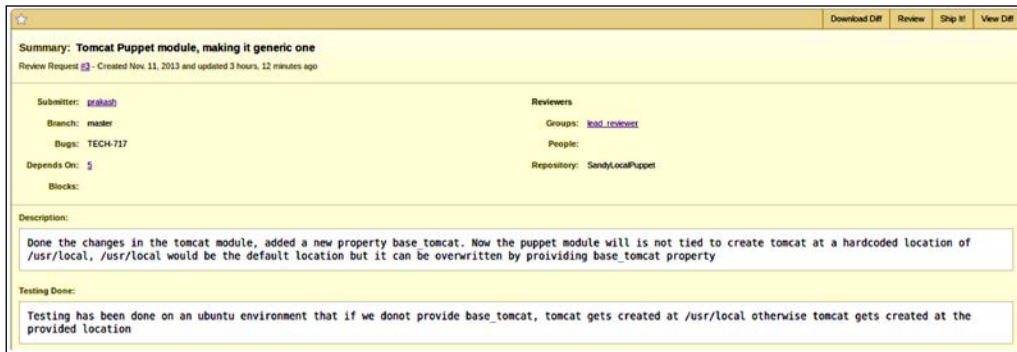
1. **The Star mark:** A reviewer can mark a review request with a star by clicking on the star icon, to mark it as an important review request. The style of the icon will be changed to highlight this. If you want to un-star the review request, just click on the star icon again.
2. **The Download Diff link:** Sometimes a reviewer prefers to review the code by downloading the code diff in their system and then running it with the code. The **Download Diff** link helps the reviewer to download the diff so that they can test it on their local system.

- The Ship It! link:** Once the reviewer is done with the code review and is satisfied with the quality and stability of the code under review, they can approve of the code to be shipped to the production system by clicking on the **Ship It!** link. On clicking on the **Ship It!** link, a confirmation dialog that says **Are you sure you want to post this review?** will be displayed to take a final confirmation from the reviewer about the go-ahead given to the code. **Ship It!** only has a logical significance in terms of the code review flow. Different organizations use it in different ways. One of the recommended practices is to use it as a voting mechanism where each click on **Ship It!** is treated as a vote. It depends on an organization to decide how strict they will be with reviewing. The most common practice is the **n-voting rule** where an organization decides upon a number; let's say the rule is 2 or 3 – so, a review request needs at least 2 or 3 Ship It votes before a review is closed. In stricter organizations, usually all of the reviewers should approve the review request (by marking it as Ship It) before it is closed. An example of this is given in the following screenshot:

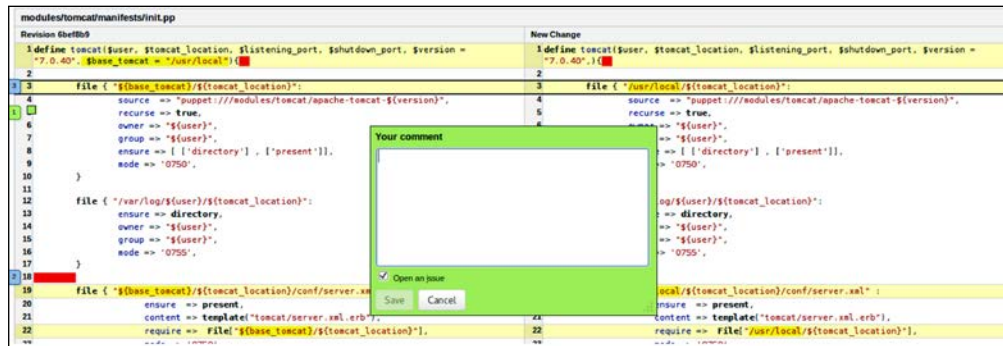


- The Review link:** A review option, as shown in the following screenshot, can be used in two scenarios. The first scenario is when a reviewer wants to give the go-ahead (Ship It) with a comment. By clicking on the **Review** link, a popup will be presented with a comment box along with a checkbox corresponding to Ship It. The second scenario is to provide the review comment to the author of the code so that they could take some action on the review. If you click on the **Publish Review** button, the review becomes directly available for others to see; you can also choose to save it and modify it later. If you choose the **Save** option, you will see a notification about your pending review at the top of the screen.

Now, you can choose to edit the saved review in case you have saved it instead of publishing it. You can publish the review request to the author or discard it altogether.

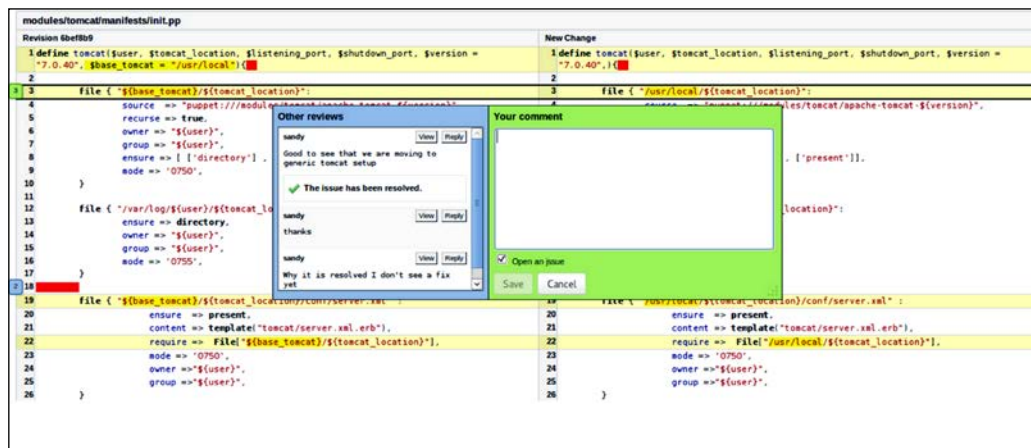


5. **The View Diff link:** **View Diff** is the most important part of code review. Through **View Diff**, you can provide review comments and view the existing ones. To provide review comments, click on the **View Diff** link. On doing so, a new page will open (that is, the Diff Viewer), which will show the code diff along with the option to view and provide review comments, as shown in the following screenshot:



With the help of Diff Viewer, you can add both single-line and multi-line comments. To provide a single-line comment (say, on line number 5), click on the line number mentioned on the left-hand side. Similarly, if you want to add a multi-line comment, click on the first line number (say 5) and drag till the last line number (say 9). This will open a popup where you can provide your review comment. You can also open an issue for tracking your review comment better.

Diff Viewer also provides a very simple view of the already provided review comments. All of the single-line and multi-line reviews are highlighted along with the total number of review comments provided with the review as shown in the following screenshot. There are three review comments on line number 3 and two review comments on line number 18:



To view the review comments, you just have to click on the number and it will open a popup for you showing all of the review comments. As shown in the preceding screenshot, by clicking on number 3 on line number 3, a popup is opened showing all three review comments along with the name of the author who added them. You also have an option to add your own comment. A user can choose to reply to existing review comments by clicking on the **Review** button placed with each review comment or add a new review comment altogether in the **Your Comment** box. The significant difference is that if a review comment is placed as a reply, it can be tracked as a conversation.

Managing issues

One of the very good features that Review Board provides along with code review is the creation of issues with each review comment. While providing a review comment, a reviewer has an option to open an issue along with the review comment. Issues can be used to track various review comments better, because issues can be managed on the basis of who raised them and also on the basis of various statuses (**Open**, **Dropped**, and **Resolved**). You can view the statuses in the following screenshot:

Issue Summary:

Total: 3 Open: 1 Resolved: 1 Dropped: 1

Status: From:

Description	From	Last Updated	Status
Good to see that we are moving to generic tomcat setup	sandy	Nov 24, 2013, 12:20 p.m.	Resolved
what is this change for	sandy	Nov 24, 2013, 12:20 p.m.	Dropped
We should not have creditor in generic tomcat setup	sandy	Nov 11, 2013, 4 p.m.	Open

+ sandy Posted 2 weeks ago (Nov 11, 2013, 4 p.m.)

+ Review request changed Updated 1 day, 12 hours ago (Nov 24, 2013, 6:07 a.m.)

+ Review request changed Updated 1 day, 12 hours ago (Nov 24, 2013, 6:07 a.m.)

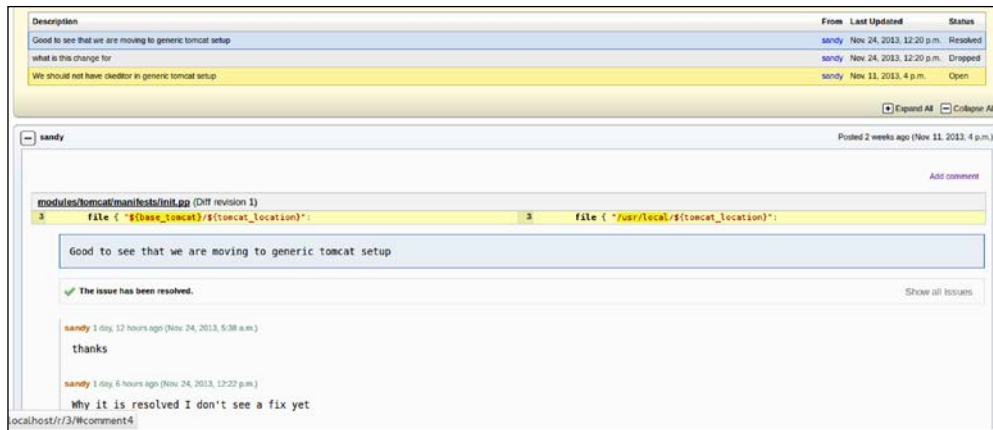
+ Review request changed Updated 1 day, 5 hours ago (Nov 24, 2013, 12:31 p.m.)

- Review request changed Updated 1 day, 5 hours ago (Nov 24, 2013, 12:36 p.m.)

- Depends On
 - added [review request without diff](#)
- Testing Done

Review Board also provides a consolidated list of all the issues associated with the review request. By default, the **Issues Summary** section shows only **Open** issues created by all of the reviewers; the **Status** dropdown has **Dropped**, **Resolved**, and **All** as the other options. Similarly, the **From** dropdown contains a list of all the reviewers who created issues.

The **Issue Summary** section contains a table of details of all the issues having the columns **Description**, **From**, **Last Updated**, and **Status**. The **Description** cell, as shown in the following screenshot, is a link to the actual issue; it will take the user to the complete displayed details of that issue:



As shown in the preceding screenshot, when a user clicks on the issue **Good to see that we are moving to generic tomcat setup**, they are presented with the details of the issue. It displays various comments that have been added to the issue along with the code section for which the issue was opened.

Only the author of the code can take the corrective action on an issue; that is, either mark an issue as **Fixed** or **Dropped**. When changing the state of the issue, the author can also provide a comment giving the details about the action. A fixed or dropped issue can be reopened by clicking on the **Re Open** button.

Summary

This chapter was very important for a code reviewer; we learned about reviewing, managing, and tracking review requests. Next, we learned how to select a code review request and interpret information from that code review request; that is, who the requester of the review request is and who else has been assigned for the code review. In addition to these details, a code review request also contains more relevant information such as the branch on which the code review request was created, if the code review request is assigned or tracked through any bug, and whether we have a testing strategy for the code under review or not. Finally, we discussed the most important part of the chapter – how Review Board facilitates a code review, and the way it tracks various reviews provided by different reviewers, be it single-line or multi-line review comments. Also, we learned about the concept of Ship It.

The next chapter will talk about how a user can use the search feature of Review Board and the various options the users have to customize the Review Board dashboard. We will also be looking at what can be managed via **User Preferences**, such as changing a password to make a profile private.

4

The User Dashboard, Preferences, and Searching

So far, we have learned how to work with a review request from end to end – both from an author's and a code reviewer's perspective. In this chapter, we will look at some important features, that is, the dashboard and search, which help the user in their day-to-day activities with Review Board. Also, we will have a brief look at user preferences.

The user dashboard

The user dashboard, as shown in the following screenshot, is the default screen that is displayed when a user logs in. It presents a summary of all of the review requests that the logged-in user might be concerned with. If the user has admin permissions, they can access the Admin Dashboard. We will discuss more about the Admin Dashboard later.

The screenshot shows the Review Board 1.7.18 user dashboard. At the top, there is a search bar and the user's name 'Sandeep' with a dropdown arrow, and a 'Support' link. Below this is a navigation bar with links: 'My Dashboard', 'New Review Request', 'All Review Requests', 'Groups', and 'Submitters'. On the left is a sidebar with a list of review categories and their counts: 'Starred Reviews' (0), 'Outgoing Reviews' (3), 'Incoming Reviews' (2), 'To Me' (1), 'code-reviewers' (1), and 'All My Requests' (3). The main content area is titled 'All Incoming Review Requests' and contains a table with columns for 'Summary', 'Posted', 'Last Updated', and 'Last Updated'. Two review requests are listed in the table.

Summary	Posted	Last Updated	Last Updated
Review request for myself	November 25th, 2013, 8:55 p.m.	2 days, 1 hour ago	November 25th, 2013, 8:55 p.m.
Review request for user authentication bug fix	November 25th, 2013, 8:53 p.m.	2 days ago	November 25th, 2013, 8:57 p.m.

There is a navigation tree on the left-hand side that has the summary of the review requests grouped by the type of request and the corresponding count of review requests. There is a table on the right-hand side that displays the list of the review requests for the selected group in the navigation tree. The columns for this table can be configured by clicking on the icon in the last column on the right-hand side.

Let's look at the different groups of review requests.

Incoming Reviews

Incoming Reviews are the review requests assigned to the current user and the groups to which the current user belongs. **Incoming Reviews** are displayed by default when the user clicks on **My Dashboard**.

Incoming Reviews are further subcategorized into **To Me** (that is, requests that are directly assigned to the user) and groups a user belongs to (for example, **code-reviewers**, as shown in the previous screenshot). A user can see the reviews for each subcategory by clicking on it in the navigation tree.

Outgoing Reviews

Outgoing Reviews are the review requests created by the current user and are either in the **Open** or the "not yet published" state, which is the **Draft** state. The states are described in the following list:

- **Open:** These are review requests that are created and shared among reviewers for reviewing the code
- **Draft:** These are review requests that are created but not yet shared among the reviewers for reviewing the code

Starred Reviews

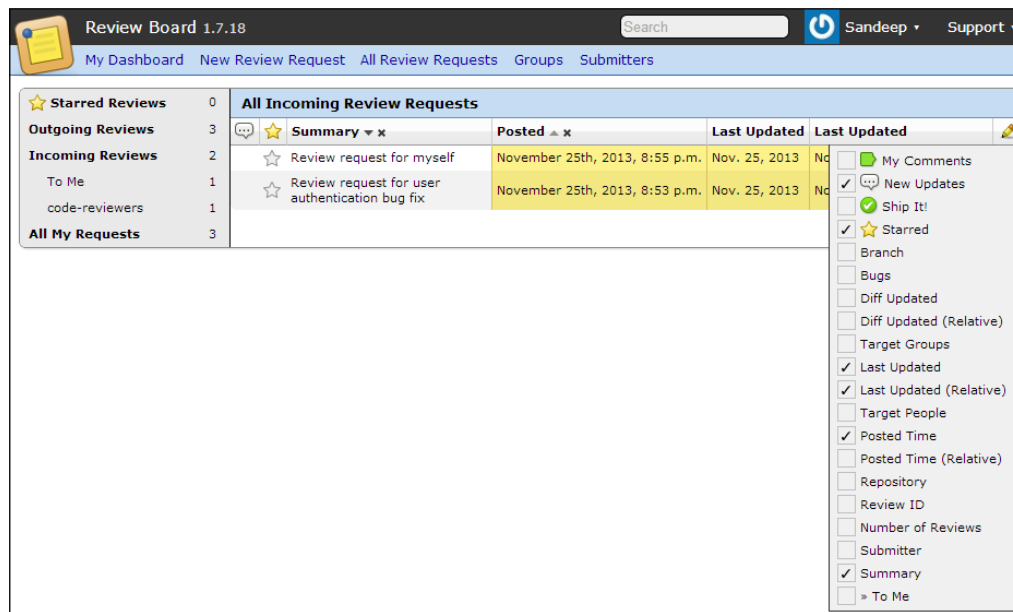
Starred Reviews are the ones on which the current user has marked a star. You can mark a review request with a star if you want to track it for a specific reason.

All My Requests

Clicking on the **All My Requests** link will open a list of all of the review requests created by the logged-in user.

Table columns

There are columns that you can configure to be displayed in the table. The order of the displayed columns can be changed by dragging-and-dropping the columns in the order you want. The following screenshot shows all of these columns:



Each column can also be sorted. You can sort a column in ascending order by clicking on the column header. The sort order is indicated by an arrow pointing in the upward or downward direction. Once sorted, it shows an "x" sign, and the sorting can be removed by clicking on this. The column that was clicked on last will be the one in which the list will be sorted primarily; all of the other columns clicked on earlier will be secondary sorting columns in reverse chronological order (in the reverse order in which they were clicked on).

The following table provides a description for each column:

Column Name	Description
My Comments	This shows a comment flag with different colors depending on whether the comment has been published or not. If a user has an unpublished comment on the review request, it shows a green flag, and if a user has a published comment, it shows a blue flag.
New Updates	If you have seen a review request and there has been any update to that review request, this column shows a bubble icon. It doesn't show this for the review requests that you haven't seen at least once.

Column Name	Description
Ship It!	If a request has been approved (someone has marked it with Ship It!), this column shows a checkmark icon.
Starred	This shows an enabled (toggled on) star indicator if a user has marked a star on the review request. The user will be copied into the discussion of the review request on which they have marked a star.
Branch	This shows the branch information present in the review request.
Bugs	This shows the list of bug IDs entered as part of the review request.
Diff Updated	This shows the timestamp of the last diff update. It is color-coded to indicate the age.
Diff Updated (Relative)	This shows the time since the last diff update. It is color-coded to indicate the age.
Last Updated	This shows the timestamp of the last update to the review request. It is color-coded to indicate the age.
Last Updated (Relative)	This shows the time since the last update to the review request. It is color-coded to indicate the age.
Posted Time	This shows the timestamp when the review request was first posted. It is color-coded to indicate the age.
Posted Time (Relative)	This shows the time since the review request was first posted. It is color-coded to indicate the age.
Repository	This shows the repository on which the review request was created.
Review ID	This shows the unique identifier of the review request.
Number of Reviews	This shows the number of reviews made on the review request.
Submitter	This shows the username of the submitter of the review request.
Summary	This shows the summary text of the review request.
To Me	This shows a chevron for review requests on which a user was added in the People field as a reviewer.

Preferences / My Account

A user can set their preferences, as shown in the following screenshot, by clicking on the **My Account** link that is displayed when you hover the mouse over the username in the top-right corner:

Review Board 1.7.18

Search Sandeep Support

My Dashboard New Review Request All Review Requests Groups Submitters

User Preferences

First Name: Sandeep

Last Name:

Email: sandeep_r80@yahoo.co.in

Change Password:

Verify New Password:

Time Zone: UTC

Enable syntax highlighting in the diff viewer

Keep your user profile private

Always open an issue when comment box opens

Groups

Which groups do you belong to or wish to watch?

Code Reviewers

Save Preferences

The **User Preferences** section lets you provide general information about the user, such as the display name and e-mail address, and lets you update the password. It is a recommended practice to update the password for the first time if the account has been created by the admin.

A user can also take any of the following steps:

- Enable syntax highlighting in the Diff Viewer (enabled by default)
- Keep your user profile private (disabled by default)
- Always open an issue when the comment box opens (enabled by default)

Choosing this issue depends on how strict you are as a reviewer. Ideally, this option should not be checked in the **User Preference** section; while doing a code review, a reviewer has the option to open an issue explicitly.

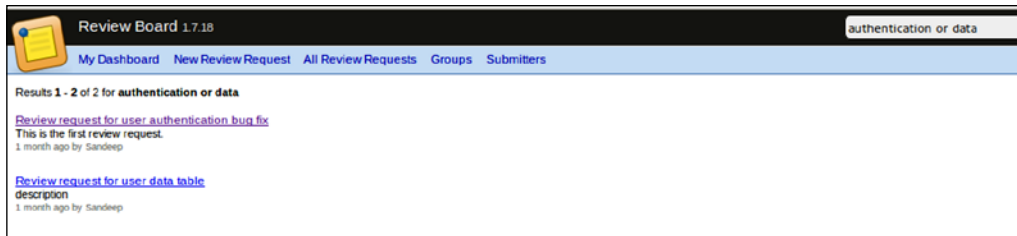
This page also lists the **Groups** that a user belongs to. A user can deselect a particular group in order to leave it.

Searching

There are various ways to search review requests in Review Board. Some of them are explained in the upcoming sections.

Quick search

There is a search box at the top of the screen with text that reads **Jump to...** This can be used to quickly search review requests, groups, and users. It starts searching and showing results as you start typing a few characters into it. Clicking on a particular result will lead to the appropriate page depending on the type of the result (review request, group, or user). An example of the same is shown in the following screenshot:



A quick search can be performed based on the following attributes:

- Review request unique identifiers (ID)
- Review request summary
- Username
- User's first/last name
- Group name
- Group's display name

Full-text search

If full-text search is enabled on the Review Board site/server, on pressing the *Enter* key in the quick search box after entering the search text, it will start a full search on the review request fields. We will learn how to enable this when we discuss the admin features.

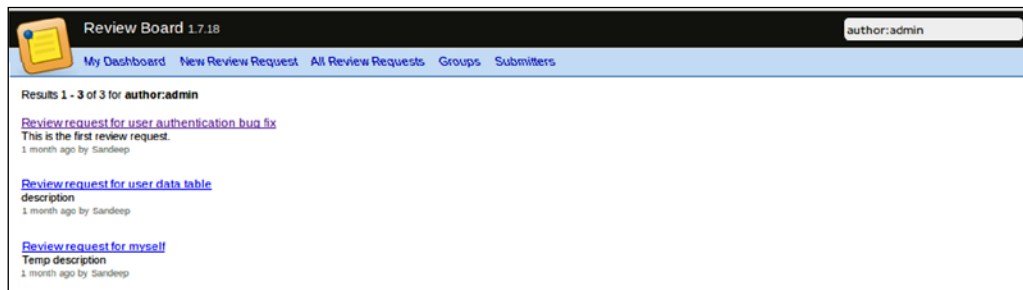
Full-text search allows you to input a query using logical operators such as AND, OR, and NOT, as well as phrases (text in double quotes).

If you enter multiple words in the textbox, the search will occur after applying the OR operator on these words. For example, if you enter `confirmation dialog`, it will give the results with any of these words. You can also use OR explicitly in the query, for example, `confirmation OR dialog`.

If you use the AND operator (for example, `confirmation AND dialog`), it will search for the presence of both the words.

If you use the NOT operator, it will exclude the words that follow NOT from the search criteria; that is, if you search for `confirmation NOT dialog`, it will produce the results that have the word `confirmation` but not the word `dialog`.

You can use double quotes to match the exact phrase. For example, the search query `"confirmation dialog"` will run a search for results containing the phrase `confirmation dialog`. An example is given in the following screenshot:



You can also use certain review request fields in the query string to narrow down the search further. You can use the `field:` prefix to a search text to search for text within that field, where `field` is one of the following:

- `summary:` The search will be within the `summary` field. For example, `summary: confirmation` will produce results with `confirmation` in the `summary` text.
- `author:` The search will be within the username and full name of the user.
- `username:` The search will be within a username.
- `bug:` The search will be for a bug ID. For example, searching for `bug:1234` will produce the results with the bug ID `1234`.
- `file:` The search will be for a diff filename.

You can search very efficiently using these fields. For example, you can search `summary: confirmation AND author:sandy AND bug:1234 AND file:foo.java` (without double quotes).

Summary

In this chapter, we learned about the user dashboard that provides an overview of the review requests a user is most concerned about. Also, we looked at the preferences that a user can set.

We looked at different searching capabilities that are useful if you have a long list of review requests, groups, or users to look at. In the next chapter, we will move into the administration territory. We will look at the detailed steps for Review Board installation along with how to upgrade Review Board.

5

Installing and Upgrading Review Board

In this chapter, we will start our journey as administrators of Review Board. As an administrator, it is really important to set up a perfectly running application with the best possible settings. We will cover the following topics in this chapter:

- Hosting the Review Board site on the RBCommons site
- Installation instructions of Review Board on the Ubuntu flavor of the Linux operating system
- How to create a working Review Board site
- How to upgrade Review Board

Setting up Review Board

You can set up Review Board in any of the following ways:

- By creating your own Review Board server
- By hosting the Review Board server at RBCommons

Hosting the Review Board server at RBCommons is pretty straightforward. You just have to go to the RBCommons site (<https://www.rbcommons.com/plans/>) and choose the plan that best suits your requirements.

RBCommons offers four different plans depending on the size and scale of the setup required, as shown in the following table:

Plan Type	Cost(\$)/Month	User's Limit	Max Repositories
Starter	29	10	3

Plan Type	Cost(\$)/Month	User's Limit	Max Repositories
Medium	99	25	20
Large	229	50	60
Enterprise	499	100	120

After a plan is chosen, a window will appear to create a team, where we have to provide personal, team, and billing details as shown in the following screenshot. With this, your Review Board server will be hosted on the RBCCommons site and can be used for performing code review requests.

Create Your Team

Create user account

Do you already have an account? [Log in](#) to skip this step.

First name: Sandeep, Last name: Rawat, Username: sandeep, E-mail address: sandeep.r80@yahoo.co.in, Password: *****, Confirm password: *****

Choose a team name

Team name: test

Your Review Board will be accessible at <https://rbcommons.com/s/test/>

Enter your billing details

Your free trial will last 14 days, after which we'll begin charging you \$29/month. You can upgrade, downgrade, or cancel at any time.

Payment options: VISA, American Express, MasterCard, Discover, JCB

Card number: 1234567890123456, Expiration date: 01-Jan, 2021, Security code (CVC): 123

How did you hear about us?

This is optional, but it's always nice to know who we can thank!

Create Team

The other option to set up Review Board installation is to create a Review Board site on your machine. Review Board has officially stopped providing support for the Windows environment after Version 1.6 (<http://www.reviewboard.org/docs/manual/dev/admin/installation/windows/>). Linux environments have become the best and the only viable options to set up Review Board. You can choose any Linux flavor: Fedora, openSUSE, Ubuntu, Debian, CentOS, and so on. The Ubuntu and Fedora Linux environments are the most preferred and recommended ones for setting up Review Board; even Review Board recommends these environments.

Before starting with Review Board installation, make sure that the system has the following prerequisites installed:

- **Database:** MySQL, PostgreSQL, or SQLite can be used as a database option. SQLite should only be used while doing a POC or getting used to Review Board. On a production system, you should use either MySQL or PostgreSQL. MySQL is the preferred option for the setup.
- **Web server:** A web server is required to host the Review Board site. Apache and lighttpd are the two available options of web servers. In addition, make sure that the web server is capable of hosting Python applications. The Apache web server provides three modules for hosting Python applications: `mod_wsgi`, `mod_astcgi`, and `mod_python`. The preferred module is `mod_wsgi`. In the case of lighttpd, the `fastcgi` module will be used to enable Python application hosting on lighttpd.
- **Version Control System client:** Depending on the version control systems to be used with Review Board, the corresponding version control system clients need to be installed.

Once you make sure that the system has all the prerequisites, you can start with installing Review Board on the system. The Review Board setup is very simple and straightforward. We will use the following tools for our Review Board setup:

- **Database:** MySQL
- **Web server:** Apache with `mod_wsgi`
- **Version Control System:** Subversion (SVN)
- **Environment:** Ubuntu system

You can choose the database, VCS, and web server to install.

Review Board installation

Setting up Review Board is a two-step process:

- Installing Review Board
- Publishing the Review Board site on the web server

Make sure that you switch to the root user before starting with the set up (else you would have to prefix `sudo` before every command). Then, run the following commands one by one:

```
1 apt-get install python-setuptools
2 easy_install --version
3 apt-get install patch
4 easy_install ReviewBoard
5 easy_install mysql-python
6 apt-get install subversion python-svn
```

The next sections will describe the commands given in the preceding image.

Steps 1 and 2 – installing the Python setuptools

You will need the Python setuptools to install Review Board; usually, Ubuntu already has this installed. If not, you can install it by executing the following command:

```
$ apt-get install python-setuptools
```

The preceding command will change from one operating system to another.

For Fedora8, the command will be as follows:

```
yum install -y python-setuptools-devel.noarch
```

For RedHat Enterprise, CentOS, and Fedora7, the command will be as follows:

```
yum install python-setuptools
```

Make sure you have a 0.6c9 or higher version of the Python setuptools; run the following command to check the version:

```
$ easy_install --version
```

Step 3 – installing Patch

As you know, the **Diff Viewer** is the most important feature of any code review tool; to add the Diff Viewer functionality in Review Board, you need to install **Patch**. The command for installing Patch is as follows:

```
$ apt-get install patch
```

For RedHat Enterprise, CentOS, and Fedora, the command will be as follows:

```
yum install patch
```

Step 4 – installing Review Board

The next step is to install Review Board and all its required dependencies. The dependencies comprise of **Djblets**, **Django-Evolution**, **Django**, **flup**, **paramiko**, and **Python Imaging Library**. The command for installing Review Board is as follows:

```
$ easy_install ReviewBoard
```

Step 5 – installing MySQL database binding

To set up the MySQL database, we need to have the Python module, using which the Review Board Python code can connect to the MySQL database. The command for installing MySQL is as follows:

```
$ apt-get install python-mysqldb
```

In the case of other operating systems, the command will be as follows:

```
easy_install mysql-python
```

If you want to use PostgreSQL instead of MySQL, the command will be as follows:

```
easy_install psycopg2
```

Step 6 – the subversion source control component

To enable Review Board to communicate with *subversion*, we need two packages to be installed on the system: *python-svn* and *subversion*, as shown in the following command. The *python-svn* package will provide the necessary support to allow the Python code to communicate with *subversion*, and the *subversion* package will provide the native support for communication with the *subversion* repository.

```
$ apt-get install subversion python-svn
```

```
yum install subversion
```

For other version control systems, you have to execute the following commands:

- **CVS:** For installing CVS, the command is as follows:

```
apt-get install cvs  
yum install cvs
```
- **Git:** For installing Git, the command is as follows:

```
apt-get install git-core  
yum install git-core
```
- **Mercurial:** For installing Mercurial, the command is as follows:

```
easy_install mercurial
```
- **Perforce:** For installing Perforce, the command is as follows:

```
easy_install P4PythonInstaller
```

Installing the Review Board site

Once Review Board is installed, the next step is to create the Review Board site. Site creation will be done using **rb-site**. The rb-site tool is bundled with Review Board and is installed along with Review Board. We will install Review Board with the hostname `reviewboard.server.com` under the directory `/var/www/reviewboard.server.com` using the following command:

```
$ rb-site install /var/www/ReviewBoard.server.com
```

The preceding command will present a set of questions for the site installation; if your server has the X11 environment, the options will be presented through a UI; otherwise, they will be presented in the form of questions in the console. We will be showing how to set up Review Board on a non-X11 environment. The following image describes the steps for installing the Review Board site:

```
1 rb-site install /var/www/reviewboards.server.com
2 Domain Name: reviewboards.server.com
3 Root Path [/]:
4 Shipped Media URL [static/]:
5 Uploaded Media URL [media/]:
  (1) mysql
  (2) sqlite3 (not supported for production use)
6 Database Type: 1
7 Database Name [reviewboard]: reviewboard
8 Database Server [localhost]:
9 Database Username: reviewboard
10 Database Password:
Confirm Database Password [reviewboard]:
* What cache mechanism should be used?
  (1) memcached (recommended)
  (2) file
11 Cache Type: 2
Cache Directory [/tmp/reviewboard_cache]:
* What web server will you be using?
  (1) apache
  (2) lighttpd
12 Web Server: 1
* What Python loader module will you be using?
  (1) wsgi (recommended)
  (2) fastcgi
  (3) modpython (no longer supported)
13 Python Loader: 1
* Create an administrator account
14 Username [admin]:
15 Password:
Confirm Password [admin]:
16 E-Mail Address: sandeep_r80@yahoo.co.in
* Installing the site...
Building site directories ... OK
Building site configuration files ... OK
Creating database ... Creating tables ...
Creating table django_admin_log
* The site has been installed
  The site has been installed in
  /var/www/reviewboards.server.com
```

The following list will describe the steps given in the preceding image:

- Step 1: The `rb-site install` command will start with the installation of Review Board and displays options to set up Review Board (as shown in the preceding image).
- Step 2: It specifies the domain name of the Review Board server.

- Steps 6, 7, 8, 9, and 10: The database details make sure that the database has already been set up with the required access. In the preceding setup, we have selected the MySQL database with the user name as `reviewboard` and the password as `reviewboard`. Set up your MySQL database accordingly as shown in the following screenshot:

```
1
2
3 $ mysql -uroot -p
4 mysql> create database reviewboard;
5 Query OK, 1 row affected (0.00 sec)
6 mysql> create user 'reviewboard'@'localhost' identified by 'reviewboard';
7 Query OK, 0 rows affected (0.00 sec)
8 mysql> grant all on reviewboard.* to 'reviewboard'@'localhost'
9 Query OK, 0 rows affected (0.00 sec)
10 mysql> exit
11 Bye
```

- Step 11: Choose the cache type; we have selected the filesystem as the cache option just to avoid more variables. On a production system, you should always use memcached as the cache type.
- Steps 14 and 15: Here, you have to provide credentials for the administrator account of your Review Board site. Provide the details cautiously as we would be providing credentials of the admin account.
- Once done, the last step of the wizard would be installing the site. This step will show the status of various steps. This step will take some time as the actual operations just configured would be performed only at this step.

Updating Apache config

The last and final step is to integrate Review Board with Apache. For this, you have to copy the Apache configuration of your Review Board site to the Apache configuration present in the directory named `sites-available`. Once done, you have to enable Review Board in Apache and finally restart Apache to play with a fully functional Review Board as shown in the following command lines:

```
$ cp /var/www/reviewboard.server.com/conf/apache-wsgi.conf
/etc/apache2/sites-available/reviewboard.server.com
$ a2ensite reviewboard.server.com
$ service apache2 restart
```

Upgrading Review Board

Upgrading Review Board is also pretty straightforward as is the case with its installation. Upgrading Review Board involves the following steps:

- Upgrade Review Board
- Upgrade all the installed sites on Review Board
- Restart the web server

A Review Board upgrade will be done with the `easy_install` command.

```
easy_install -U ReviewBoard
```

The preceding command will perform the following operations:

- Search for the best possible Review Board version to be upgraded to
- Install the corresponding tools such as `rb-site` or `rbssh`
- Update the dependencies such as `Djblets` and `Django`

The following screenshot explains the operations performed by the `easy_install` command:

```
easy_install -U ReviewBoard
Searching for ReviewBoard
.
.
Best match: ReviewBoard 1.7.19
.
.
Removing ReviewBoard 1.7.16 from easy-install.pth file
Adding ReviewBoard 1.7.19 to easy-install.pth file
Installing rb-site script to /usr/local/bin
Installing rbssh script to /usr/local/bin
.
.
Processing dependencies for ReviewBoard
Searching for Djblets>=0.7.24,<0.8
.
.
Best match: Djblets 0.7.25
.
.
Removing Djblets 0.7.21 from easy-install.pth file
Adding Djblets 0.7.25 to easy-install.pth file
.
.
Searching for Django>=1.4.10,<1.5
Reading http://pypi.python.org/simple/Django/
Best match: Django 1.4.10
.
.
Removing Django 1.4.8 from easy-install.pth file
Adding Django 1.4.10 to easy-install.pth file
Installing django-admin.py script to /usr/local/bin
Installed /usr/local/lib/python2.7/dist-packages/Django-1.4.10-py2.7.egg
Finished processing dependencies for ReviewBoard
```

Once the Review Board upgrade is complete, the corresponding sites installed on Review Board need to be upgraded, which involves the following:

- Upgrading the media tree
- Upgrading the database

We can use the following command to upgrade the corresponding sites:

```
rb-site upgrade /var/www/reviewboard.server.com
```

A database upgrade is optional, and you can choose to skip it by providing the `--no-db-upgrade` option.

Finally, you need to restart your web server using the following command:

```
service apache2 restart  
/etc/init.d/apache2 restart
```

In case you are using Memcached as the cache type, you need to restart that as well.

Summary

In this chapter, we have learned about the two ways to set up Review Board: the first one is hosting your Review Board at the RBCommons site, and the other is installing Review Board on your local infrastructure and hosting the websites on top of it. We primarily focused on Ubuntu as the operating system on which Review Board was being installed along with what needs to be done on other operating systems.

We also looked at what needs to be done to upgrade your Review Board to the next version, which is done through the `easy-install` command.

In the next chapter, we will be looking at the various system settings that an administrator can play with, such as General, Administration, and SSH settings.

6

Admin Settings

In this chapter, we will go through the following two categories of admin settings:

- System settings
- System information

A user with admin permissions has an additional link to **Admin** under the username displayed in the top-right corner of the window. Clicking on **Admin** leads to the **Admin Dashboard** screen. The **Admin Dashboard** shows activities, statuses, groups, and many other things. It has a menu on the left side to navigate to specific settings. The left side is divided mainly into three sections, which are as follows:

- SYSTEM SETTINGS
- MANAGE
- SYSTEM INFORMATION

In the SYSTEM SETTINGS section, we will discuss the following types of settings and also how we can customize them for optimal usage:

- General
- Authentication
- E-mail
- Diff Viewer
- Logging
- SSH
- File Storage
- Support

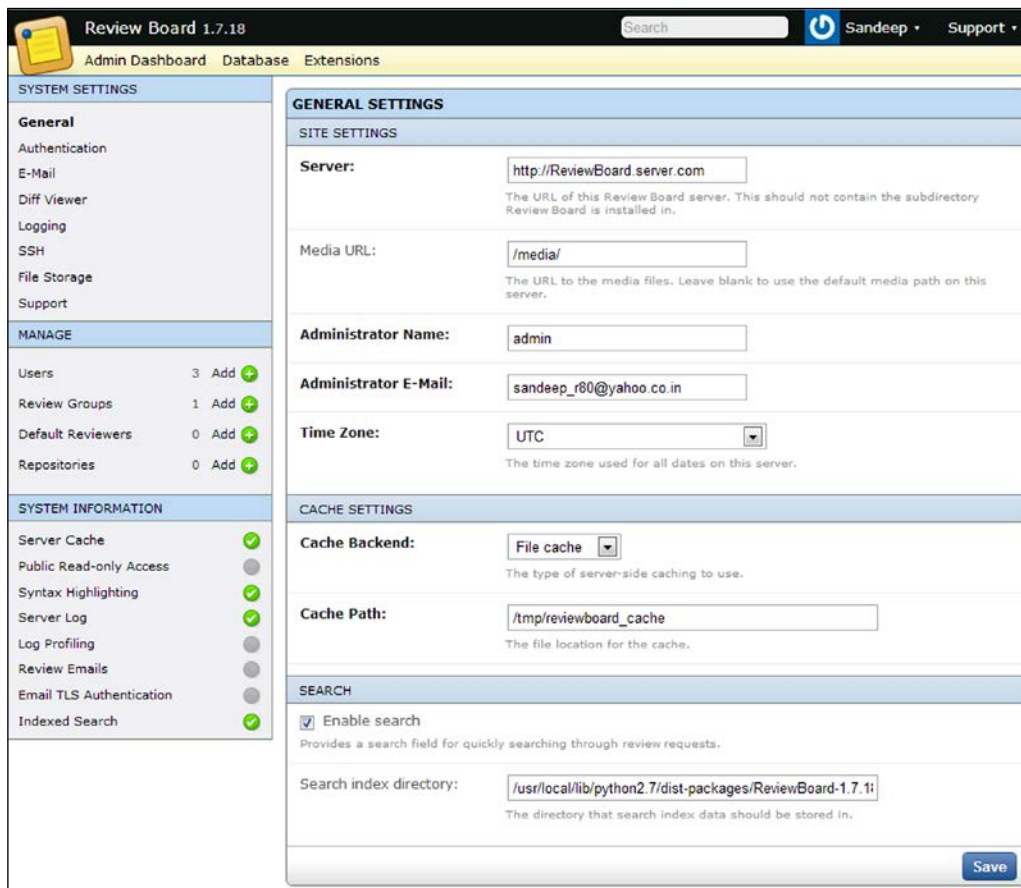
In the SYSTEM INFORMATION section, we will see an overview of some of the important settings with their statuses (enabled/disabled).

SYSTEM SETTINGS

This is the first type of admin settings that spawns settings such as General, Authentication, E-mail, Diff Viewer, Logging, SSH, File Storage, and Support. In the coming sections, we will go in details of each of these **SYSTEM SETTINGS**.

General

The following screenshot displays the settings that you first configure while performing the installation and setup of the Review Board application:



The following settings are part of **GENERAL SETTINGS** (settings marked with an asterisk (*) are mandatory and must be entered):

Settings	Description
SITE SETTINGS	
Server*	HTTP(s) + hostname
Media URL	This is the URL for the Review Board site or the external server URL where media files are hosted. The default media path is <code>/media/</code> .
Administrator Name*	This is the full name of the primary administrator.
Administrator E-Mail*	This is the primary administrator's e-mail address.
Time Zone	This is the time zone of the Review Board server. From Review Board 1.7 onwards, the user will see the correct date in their local time zone.
CACHE SETTINGS	
Cache Backend	This is the cache backend to be used. The supported options are Memcached and File cache . The Review Board team recommends Memcached over File cache.
Cache Path	This option is displayed when the File cache backend is selected. It's the absolute file path on the Review Board server where Review Board can cache the data.
SEARCH	
Enable search	If this is enabled, a full text search can be performed on the requests. It requires the installation of PyLucene and regular search indexing to function.
Search index directory	This is the relative (relative to the Review Board site directory) or absolute path on the Review Board server where the search index files will be stored. The default path (when the Search index directory field is empty) is <code>search-index</code> in the Review Board site.

Authentication

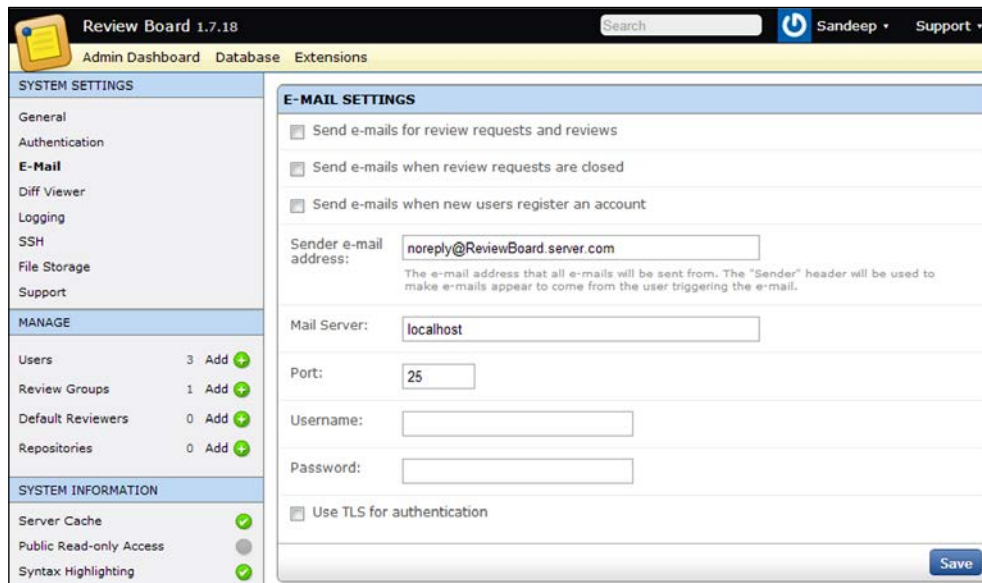
The next system setting that we will talk about is Authentication.

The following table lists the various settings that are presented in the **AUTHENTICATION SETTINGS** page along with the description of these fields:

Settings	Description
AUTHENTICATION SETTINGS	
Allow anonymous read-only access	This is enabled when anonymous users are allowed to see the review requests. Enable this if the code is not confidential or access control to the site is managed by other means; otherwise, it should be disabled.
Authentication Method	This method is used for authenticating the users. The additional settings depend on the selected method. The available methods are Standard Registration , Active Directory , LDAP , NIS , X.509 Public Key , and Legacy Authentication Module . The default method is Standard Registration .

E-mail

The next system setting, as shown in the following screenshot, is E-mail. This setting covers configurations such as the e-mail address to be used for sending messages, setting the various scenarios for which an e-mail needs to be sent, and deciding whether e-mails must be sent or not.



The following settings are a part of **E-MAIL SETTINGS**:

Settings	Description
Send e-mails for review requests and reviews	If this is enabled, it sends an e-mail to the admin e-mail address when a review request is posted and updated with reviews/replies.
Send e-mails when review requests are closed	If this is enabled, it sends an e-mail to the admin e-mail address when the review requests are closed.
Send e-mails when new users register an account	If this is enabled, it sends an e-mail when a new user registers an account. This is useful when the Standard Registration method is used for authentication.
Sender e-mail address	This is the sender's e-mail address. The default value is <code>noreply@<server.com></code> .
Mail Server	This is the SMTP server for the outgoing e-mails. The default value is <code>localhost</code> .

Settings	Description
Port	This is the SMTP server port. The default value is 25.
Username	This is the username that should be used to connect to the SMTP host. It's optional and depends on the SMTP server configuration.
Password	This is the password for the preceding username. Again, it is optional.
Use TLS for authentication	If the mail server supports TLS, then this can be enabled as it is considered to be more secure.

Diff Viewer

Through the Diff Viewer settings, you can control various diff viewer configurations of the Review Board, for example, whether you want the syntax highlighted or not. Similarly, there are a couple more settings, as shown in the following screenshot:

The screenshot displays the 'Review Board 1.7.18' Admin Dashboard. The top navigation bar includes 'Admin Dashboard', 'Database', and 'Extensions'. The user 'Sandeep' is logged in, with a 'Support' link. The left sidebar shows 'SYSTEM SETTINGS' with 'Diff Viewer' selected. The main content area is titled 'DIFF VIEWER SETTINGS' and contains the following configuration options:

- Show syntax highlighting
- Syntax highlighting threshold: (Files with lines greater than this number will not have syntax highlighting. Enter 0 for no limit.)
- Show trailing whitespace
- Show all whitespace for: (A comma-separated list of file patterns for which all whitespace changes should be shown. (e.g., '*.py, *.txt'))
- ADVANCED**
- These are advanced settings that control the behavior and display of the diff viewer. In general, these settings do not need to be changed.
- Max diff size (bytes): (The maximum size (in bytes) for any given diff. Enter 0 to disable size restrictions.)
- Lines of Context: (The number of unchanged lines shown above and below changed lines.)
- Paginate by: (The number of files to display per page in the diff viewer.)
- Paginate orphans: (The number of extra files required before adding another page to the diff viewer.)

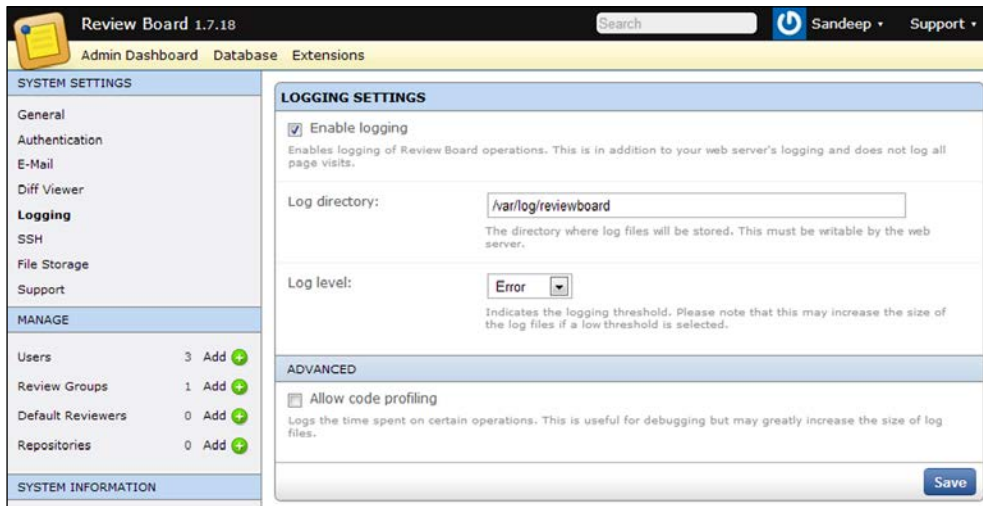
A 'Save' button is located at the bottom right of the settings panel.

The following table will give you a list with detailed descriptions of the various fields on the **DIFF VIEWER** settings page:

Settings	Description
DIFF VIEWER SETTINGS	
Show syntax highlighting	<p>This is enabled by default. When it is enabled, syntax highlighting will be used; this improves the readability of the diffs, but it takes more time to render.</p> <p>The user can override this option in their preferences.</p> <p>It may also be needed for other features, and hence disabling it might limit the functionality of the diff viewer.</p>
Syntax highlighting threshold	<p>This denotes the maximum number of lines in the file when syntax highlighting is enabled. If the file has more lines (when the threshold is not 0 or blank) than the threshold, syntax highlighting will not be enabled for that file. Zero (0) or blank means it is enabled for all the files.</p>
Show trailing whitespace	<p>This is enabled by default. It shows the excess whitespaces as red blocks.</p>
Show all whitespace for	<p>This is the list of file patterns (for example, *.py and *.txt) for which all whitespaces should be shown.</p>
ADVANCED	
Max diff size (bytes)	<p>This is the maximum allowed size of the diff (in bytes). A diff that is larger than this specified size will be rejected during the upload. The default value is 0, which means diffs of all sizes are allowed.</p>
Lines of Context	<p>This indicates the number of unchanged lines shown around (above and below) the changed lines/diff. The default value is 5.</p>
Paginate by	<p>This denotes the number of files displayed per page. The default value is 20.</p>
Paginate orphans	<p>This indicates the number of files required before adding more pages to the Diff Viewer. The default value is 10.</p> <p>For example, if a diff has 35 files and the size is set to 10, it will be displayed in three pages (not four).</p>

Logging

Through this category, you can control the logging capability of Review Board. It is recommended to turn on Logging to debug the issues that you face while using Review Board.

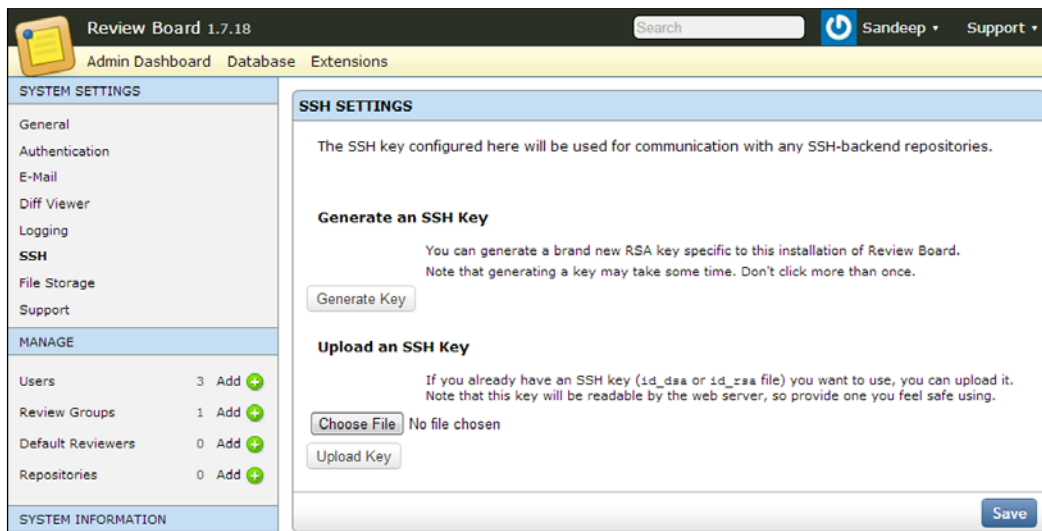


The following settings are a part of **LOGGING SETTINGS**:

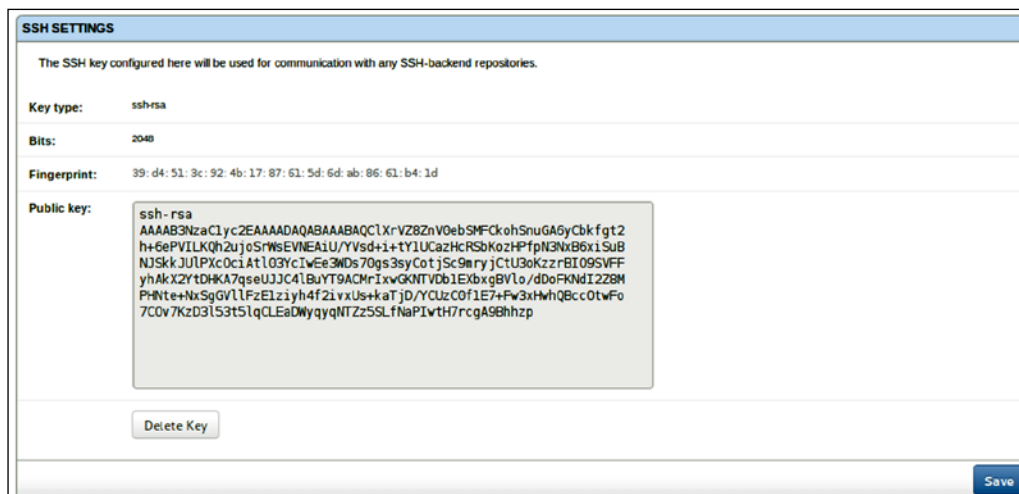
Settings	Description
LOGGING SETTINGS	
Enable logging	According to the description in the preceding screenshot, this is the additional logging of Review Board operations that could help in troubleshooting the server problems.
Log directory	This is the directory on the server where the log files will be created. The server should have the write permission on this directory. The default value is logs.
Log level	This is the level up to which you want to see the logs. The recommended option is Error . If you choose Info or Debug , it will generate a huge amount of logs, so make sure that you choose this level only as a special case.
ADVANCED	
Allow code profiling	This is disabled by default. When it is enabled, it logs the time spent on an operation. It will increase the size of a log file.

SSH

The **SSH SETTINGS** screen, as shown in the following screenshot, enables the user to set up the keys for communication with the code repositories, you can either upload an already existing key or generate a new one:



Once an SSH key is configured, the SSH screen shows the SSH key details as shown in the following screenshot:



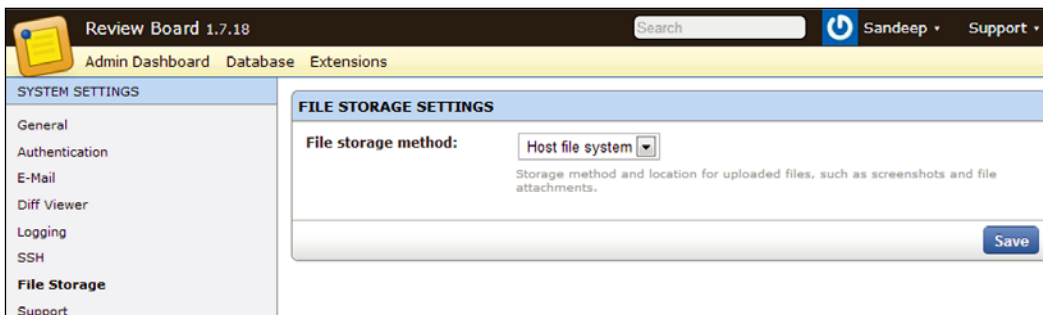
The following settings are a part of **SSH SETTINGS**:

Settings	Description
SSH SETTINGS	
Generate an SSH Key	If you don't have an SSH key already, you can generate one by clicking on the Generate Key button.
Upload an SSH Key	Click on the Upload Key button to upload an existing SSH key which you have already generated. This should be the private key (<code>id_dsa/id_rsa</code>), which can be selected by clicking on the Choose Key button. This key will be read by the server (and possibly by other applications running on it), so upload it only if you trust the server.

When the SSH key is already uploaded, the screen shows the following details: **Key type**, **Number of bits of encryption**, **Public fingerprint**, and **Public key**.

File Storage

Here you can configure the file server, where the uploaded files should be saved. There are two options: **Host file system** and **Amazon S3**.



In the case of **Host file system**, it will use the Review Board server and the files will be stored in the `media/uploaded` directory.

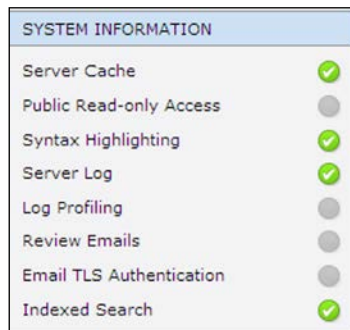
Amazon S3 requires the `django-storages` Python module to be installed on the server. In this case, Review Board will use **Amazon S3** to store files. The following settings should be configured for **Amazon S3**:

- The Amazon AWS access key
- The Amazon AWS secret access key

- The S3 bucket name
- The Amazon AWS calling format is either Path, Subdomain, or Vanity

SYSTEM INFORMATION

On the left-hand side of the menu, you can see the following SYSTEM INFORMATION items and their corresponding statuses (enabled or disabled):



SYSTEM INFORMATION	
Server Cache	<input checked="" type="checkbox"/>
Public Read-only Access	<input type="checkbox"/>
Syntax Highlighting	<input checked="" type="checkbox"/>
Server Log	<input checked="" type="checkbox"/>
Log Profiling	<input type="checkbox"/>
Review Emails	<input type="checkbox"/>
Email TLS Authentication	<input type="checkbox"/>
Indexed Search	<input checked="" type="checkbox"/>

The preceding screenshot lists a number of items and shows a checkmark against the enabled ones. Clicking on an individual item in this list shows the corresponding details on the right-hand side of the window. The following is a brief description of these items:

- **Server Cache:** This shows the cache backend and the corresponding statistics.
- **Public Read-only Access:** This option shows whether anonymous users can navigate to the Review Board site without logging in.
- **Syntax Highlighting:** Clicking on this will lead you to the Diff Viewer screen discussed earlier.
- **Server Log:** This displays the server logs in a table. The log entries are displayed in chronological order and can be filtered by date and log level.
- **Log Profiling:** Clicking on this will lead to the Logging screen discussed earlier.
- **Review Emails** and **Email TLS Authentication:** Clicking on these will lead to the E-mail settings screen.
- **Indexed Search:** Clicking on this will lead to the General settings screen discussed earlier.

Summary

In this chapter, we have looked at the configuration of the Review Board site, which is applied across the board, and some of the useful settings that can help provide better performance if used correctly. Also, we looked at logging and code profiling, which can help with troubleshooting.

In the next chapter, we will look at how to manage users, review groups, default reviewers, and repositories.

7

Managing Users and Review Groups

In this chapter, we will talk about the **MANAGE** section of the Admin Dashboard. Review Board has two main entities:

- Users
- Review Groups

In this chapter, we will see how an admin user can manage these two entities. Repositories will be covered in the next chapter.

A user with Staff status has an additional Admin link under the username displayed in the top-right corner of the Review Board window. Clicking on **Admin** leads the user to the **Admin Dashboard** screen, as shown in the following screenshot. The Admin Dashboard shows activities, statuses, groups, and many other things. It has a menu on the left-hand side with which we can navigate to specific settings. In the middle of the navigation bar is the **MANAGE** section, which is used to manage the following three prime entities of Review Board:

- Users
- Review Groups

- Repositories

The screenshot displays the review board interface with several key sections:

- SYSTEM SETTINGS:** A sidebar menu with categories like General, Authentication, E-Mail, and MANAGE. The MANAGE section includes links for Users (3 Add), Review Groups (2 Add), Default Reviewers (0 Add), and Repositories (3 Add).
- REVIEW BOARD ACTIVITY:** A line chart showing activity over time from Nov 9 to Dec 6. The legend includes Reviews (green), Comments (blue), Review Requests (red), and Change Descriptions (yellow).
- REQUEST STATUSES:** A pie chart showing 15% Pending (11) and 85% Draft (2). Submit (0) is also listed.
- RECENT ACTIONS:** A list of recent actions, including 'Trainings Batch 1' repository updates.
- REPOSITORIES:** A table listing repositories:

Name	Type	Visible
Trainings Batch 1	Subversion	✓
SandyLocalPuppet	Git	✓
SandyPuppet	Git	✓

Users

The **MANAGE** section in the left navigation menu has a Users link that leads to the user administration screen.

The screenshot shows the 'SELECT USER TO CHANGE' screen with the following details:

- SEARCH:** A search bar with a search icon and a 'Go' button. Below it, it says '0 of 3 selected'.
- USER LIST:**

Username	E-mail address	First name	Last name	Staff status
<input type="checkbox"/> admin	sandeep_100@yahoo.co.in			✓
<input type="checkbox"/> prakash				●
<input type="checkbox"/> sandy				●
- FILTER:**
 - By staff status:** All (No), No (No)
 - By superuser status:** All (No), No (No)
 - By active:** All (No), No (No)

The user administration page has a table that lists all the users available in the system. It also shows the following details about the users:

- **Username:** A clickable link to the user's edit user page
- **E-mail address:** The e-mail address of the user
- **First name:** The first name of the user
- **Last name:** The last name of the user
- **Staff status:** This signifies whether the user has admin access to Review Board or not

The user administration page also has the option to filter out users. On the right-hand side of the page, there is a **FILTER** section. Users can be filtered out depending on the following statuses:

- **Staff status:** A user who has the Staff status will have access to the administration UI, but they will have to attain the Superuser status to perform any administration operations
- **Superuser status:** A user who has the Superuser status will have full access on Review Board
- **Active:** This status signifies whether the user is currently using Review Board or not

All the mentioned filters have three options the user can choose from: **All**, **Yes**, and **No**. That is, if you wish to filter by the status of users who are currently active in the system, click on the **Yes** link under **By active** in the **FILTER** section.

Another option to filter out users is using the search box located above the table; it shows a list of users at the top of the user listbox.

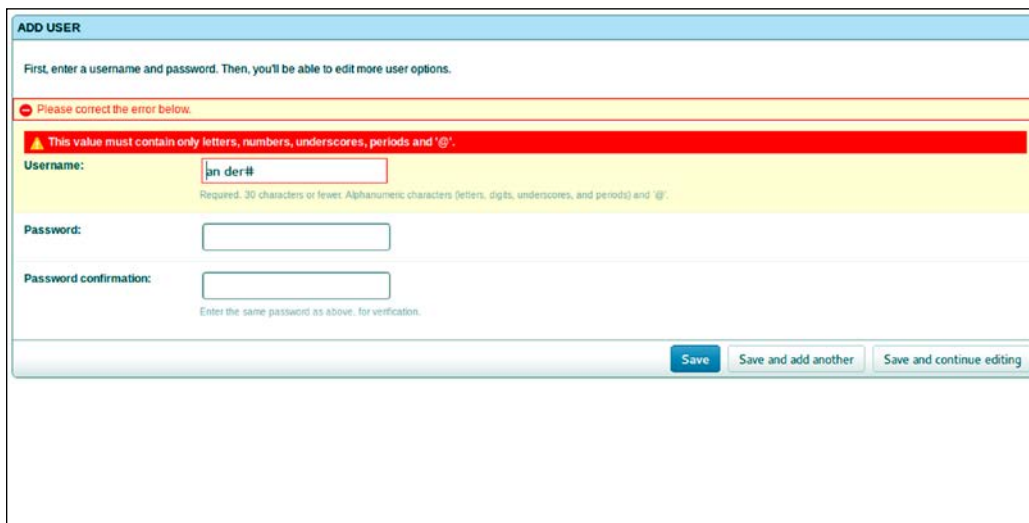
The screenshot shows a web interface titled "SELECT USER TO CHANGE" with an "Add user" link. At the top, there is a search bar containing "sa" and a "Search" button, with a result count of "2 results (13 total)". Below the search bar is an "Action:" dropdown menu and a "Go" button, with "0 of 2 selected" indicated. The main table has columns for "Username", "E-mail address", "First name", "Last name", and "Staff status". Two users are listed: "sandy" with a red status indicator and "admin" with a green status indicator and email "sandeep_r90@yahoo.co.in". Below the table, it says "2 users". On the right side, there is a "FILTER" sidebar with three sections: "By staff status" (All, Yes, No), "By superuser status" (All, Yes, No), and "By active" (All, Yes, No).

You can search for user profiles by entering a username in the search box, and clicking on the **Search** button will give a list of users matching the text entered in the search box. The result count will be displayed next to the **Search** button, and a link with the total user count will be shown adjacent to the search result so you can return to the list of all users by clicking on it.

Two links have been provided to add a new user in the system, as follows:

- The **Add user** link in the top-right corner of the user list page
- The **Add** link in the left navigation bar adjacent to the **Users** heading under the **MANAGE** section

Clicking on either of these two links will lead the user to the **ADD USER** screen.



The screenshot shows the 'ADD USER' form with the following elements:


- ADD USER** (Section Header)
- Instruction: "First, enter a username and password. Then, you'll be able to edit more user options."
- Error message: "Please correct the error below."
- Validation error: "This value must contain only letters, numbers, underscores, periods and '@'." (highlighted in red)
- Username:** Input field containing "an der#". Below it, a note states: "Required. 30 characters or fewer. Alphanumeric characters (letters, digits, underscores, and periods) and '@'."
- Password:** Empty input field.
- Password confirmation:** Empty input field. Below it, a note states: "Enter the same password as above, for verification."
- Buttons: "Save", "Save and add another", and "Save and continue editing".

The minimum input required to create a user is as follows:

- **Username:** This is a required field and is the unique identification of a user in the system. A user is referred to only by this field in the system. The maximum length of this field can be 30 characters, and only alphanumeric and @ characters are allowed for this field.
- **Password / Password confirmation:** These are the two fields where the user will provide a password. The passwords in both these fields should match.

A user can choose to perform any of the following three activities depending on the next step to be taken:

- **Save:** This option will add a new user and then take you back to the **Users** screen where all the users are listed
- **Save and add another:** Use this option to continue adding more users in the system
- **Save and continue editing:** If you'd like to save the current changes and then proceed to add more details about a user, such as the user's personal details (their first name, last name, and so on) and permission details, use this option

PERSONAL INFO	
First name:	<input type="text"/>
Last name:	<input type="text"/>
E-mail address:	<input type="text"/>
PERMISSIONS	
<input checked="" type="checkbox"/> Active	<small>Designates whether this user should be treated as active. Unselect this instead of deleting accounts.</small>
<input type="checkbox"/> Staff status	<small>Designates whether the user can log into this admin site.</small>
<input type="checkbox"/> Superuser status	<small>Designates that this user has all permissions without explicitly assigning them.</small>
Groups:	
<small>The groups this user belongs to. A user will get all permissions granted to each of his/her group. Hold down "Control", or "Command" on a Mac, to select more than one.</small>	

Review Board lets you provide fine-grained details about a user as well, through the edit-user functionality. Click on the username in the user list page to edit a particular user.

The following sections can be found on this page:

- **PERSONAL INFO:** This section of the edit user page contains the three details **First name**, **Last name**, and **E-mail address**.

- **PERMISSIONS:** This is the most important section of the edit user page. The four main permissions needed are as follows:
 - **Active:** This option is enabled if the user has been allowed to log in and work with Review Board. The user is not able to log in when this is disabled. It is suggested that you use soft deletion instead of hard deletion to delete a user; unchecking the **Active** checkbox marks the user as soft deleted.
 - **Staff status:** Check this option to give the user administrative rights access for Review Board.
 - **Superuser status:** Check this option to give all rights to a user, that is, to give full access to a user.
 - **User permissions:** A user can be given fine-grained access as well by selecting permissions in the **AVAILABLE USER PERMISSIONS** box and then moving them to the **CHOSEN USER PERMISSIONS** box by clicking on the move arrow buttons. For example, you can give the user management permission apart from other administration access to a specific user so the user can add, edit, or delete users.

Once the edit user action is complete, you can choose to perform any of the following three activities depending on the next step to be taken:

- **Save:** This option will update the user's details and take you back to the user list page
- **Save and add another:** You can use this option if you want to add another user in the system after saving the user details
- **Save and continue editing:** If you'd like to save the current changes and then proceed to add more details about a user, you can use this option

The edit user page also has a **Delete** link at the bottom to delete the user; clicking on it will present a confirmation page (**ARE YOU SURE?**). One issue with the hard deletion of a user is that if you delete a user, all the review requests that are assigned to the user will need to be updated manually. The next time you try to update a review request, Review Board will give you an error message and automatically delete the user from the list of people assigned to the review request.

Review Groups

The **MANAGE** section of the left navigation menu has a **Review Groups** link that leads to the review group administration screen.

SELECT REVIEW GROUP TO CHANGE					Add review group
Actions: <input type="text" value="-----"/> <input type="button" value="Go"/> 0 of 2 selected					
<input type="checkbox"/>	Name	Display name	Mailing list	Invite only	Visible
<input type="checkbox"/>	dev_reviewer	dev reviewer	sandeep_r100@yahoo.co.in	●	●
<input type="checkbox"/>	lead_reviewer	Lead reviewers	sandeep_r100@yahoo.co.in	●	●
2 review groups					

The review group administration page lists all the review groups available in the system. It also shows the following details about the review groups in the system:

- **Name:** A clickable link to the review group edit page
- **Display name:** A reference name for the review group
- **Mailing list:** This refers to the e-mail address corresponding to the review group; usually, it is a distribution list mapping all the reviewers belonging to the review group
- **Invite only:** This is a field signifying whether users can join this group through their preference page or whether the groups can be joined only through invitation
- **Visible:** This determines whether the review group will be visible to other users or not

The following two links are provided to add a new review group in the system:

- The **Add review group** link in the top-right corner of the review group listing page
- The **Add** link in the left navigation bar adjacent to the Review Groups heading under the **MANAGE** section

Clicking on either of these two links will lead the user to the **ADD REVIEW GROUP** screen.

The screenshot shows the 'ADD REVIEW GROUP' interface. It has a blue header bar. Below it, the 'GENERAL INFORMATION' section contains three text input fields for 'Name', 'Display name', and 'Mailing list'. A 'Visible' checkbox is checked. The 'ACCESS CONTROL' section has an 'Invite only' checkbox. Below it, the 'Users' section features two columns: 'AVAILABLE USERS' and 'CHOSEN USERS'. The 'AVAILABLE USERS' column has a search filter and a list of users: admin, prakash, sandy, user2, user3, user4, user5, user6, user7, user8, and user9. The 'CHOSEN USERS' column is empty. At the bottom, there is a 'Local site' field with a search icon.

The **ADD REVIEW GROUP** screen has two sections: **GENERAL INFORMATION** and **ACCESS CONTROL**.

The following details can be provided under the **GENERAL INFORMATION** section while creating a new review group:

- **Name:** This is a unique identifier by which a review group will be referred across the system; the name should represent the corresponding technical/business review group, such as `lead_reviewer`.
- **Display name:** This field is just for display purposes; the name provided in this field will be listed on the review group list page.
- **Mailing list:** This is the e-mail address corresponding to the review group. Usually, it is a distribution list that maps all the reviewers belonging to the review group. **Mailing list** is an optional field, but it is recommended to have the mailing list so that associated users can receive automatic notifications.

You can choose to register to the review group invite only by selecting the **Invite only** checkbox. If this option is selected, a user will not be able to register to this group via the **Groups** section of **My Account**.

In the **ACCESS CONTROL** section, you can associate users to the review group. Users can be moved between the **AVAILABLE USERS** and **CHOSEN USERS** multiselect blocks by selecting the users and clicking on the arrow keys present between these two blocks. At the bottom of these multiselect blocks, the **Choose all** and **Remove all** links are provided to move all the users between these multiselect blocks in one go.

A user can choose to perform the following three activities depending on the next step to be taken:

- **Save:** This option will add the new review group and then take you back to the review groups listing page
- **Save and add another:** Use this option to continue adding more review groups in the system
- **Save and continue editing:** If you would like to add more details about a review group, use this option

Summary

In this chapter, we went through the details of the **MANAGE** section of the Administration Dashboard and looked at how an administrator can manage users and review groups. We looked at various types of filters that can be applied to the users list and the two different staff statuses available in Review Board. We also looked at the management operations of review groups.

In the next chapter, we will talk about managing default reviewers and repositories, and finally we will be providing some advanced tips and tricks related to Review Board to utilize its maximum potential.

8

Admin Dashboard

A user with admin permissions has an additional **Admin** link under the username displayed in the top-right corner. Clicking on **Admin** leads to the Admin Dashboard screen. Admin Dashboard shows activities, statuses, groups, and many other items. It has a menu on the left-hand side to navigate to specific settings. On the right-hand side, it shows activities and request statuses. Each individual portlet (a small window that can be minimized/maximized individually) on the right-hand side can be collapsed or expanded. We have discussed some of the Admin Dashboard features in previous chapters. In this chapter, we will cover the following:

- Repository administration
- Activities
- Repositories
- Requests
- Review groups

Repository administration

The **MANAGE** section in the left navigation menu has a **Repositories** link that leads to the repository administration screen.



SELECT REPOSITORY TO CHANGE				Add repository
Repository	Path	Hosting	Show this repository	
<input type="checkbox"/> Trainings Batch 1	https://subversion.assembla.com/svn/treech/trainingsbatch1		<input checked="" type="checkbox"/>	
<input type="checkbox"/> SandyLocalPuppet	/home/sandy/personal/puppet/SandyPuppet/.git		<input checked="" type="checkbox"/>	
<input type="checkbox"/> SandyPuppet	git://github.com/sandy724/SandyPuppet.git		<input checked="" type="checkbox"/>	

3 Repositories

The repository administration page lists all the repositories available in the system. This page has a repository list table that shows the following details about repositories in the system:

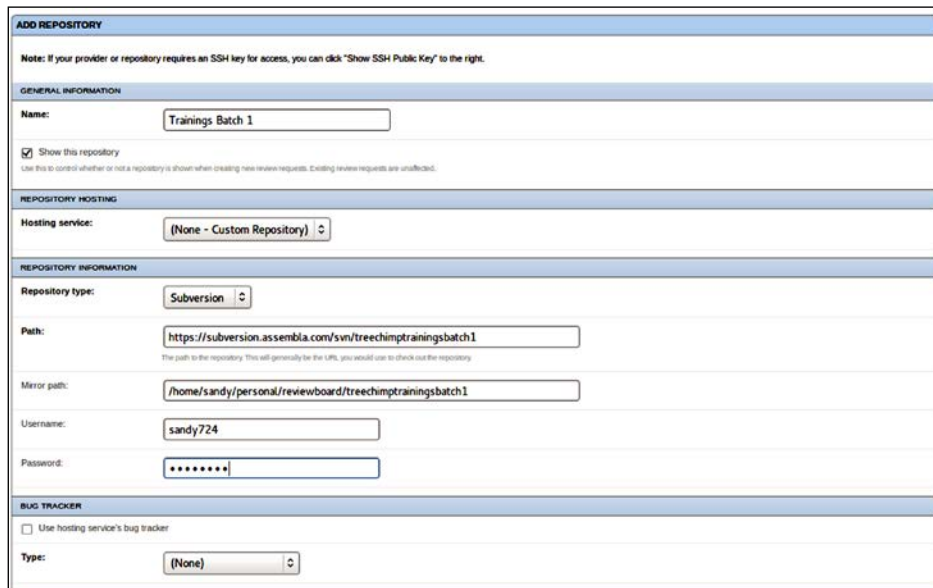
- **Repository:** This is a clickable link to the repository edit page
- **Path:** This is the path of the repository, which varies according to the type of the repository
- **Hosting:** This is the hosting service on which the repository is hosted
- **Show this repository:** This is a column that indicates whether the repository will be visible to authors and reviewers for creating new review requests

The repositories listing table page also has an option to delete groups of repositories. If you select the checkboxes for the repositories and then select **Delete selected repositories** from the **Action** dropdown, you will be presented with a confirmation page with the list of repositories that need to be deleted.

The following two links are provided for adding a new repository in the system:

- The **Add repository** link in the top-right corner of the review group listing page
- The **Add** link in the left navigation bar adjacent to the **Repositories** heading under the **MANAGE** section

Clicking on either of these two links will lead the user to the **ADD REPOSITORY** screen.



ADD REPOSITORY

Note: If your provider or repository requires an SSH key for access, you can click "Show SSH Public Key" to the right.

GENERAL INFORMATION

Name:

Show this repository
Use this to control whether or not a repository is shown when creating new review requests. Existing review requests are unaffected.

REPOSITORY HOSTING

Hosting service:

REPOSITORY INFORMATION

Repository type:

Path:
The path to the repository. This will generally be the URL you would use to check out the repository.

Mirror path:

Username:

Password:

BUG TRACKER

Use hosting service's bug tracker

Type:

The **ADD REPOSITORY** screen has multiple sections, as follows:

- **GENERAL INFORMATION:** This section contains general details about the repository, such as the name of the repository through which it would be identified and whether the repository is to be made public or not.
- **REPOSITORY HOSTING:** This hosting service lets you indicate whether you have a local repository or remote repository. In the case of a local repository, choose **Custom Repository**; otherwise, choose the corresponding hosting service where your repository is hosted.

REPOSITORY HOSTING	
Hosting service:	<input type="text" value="GitHub"/>
Account:	<input type="text" value="<Link a new account>"/>
<small>Link this repository to an account on the hosting service. This username may be used as part of the repository URL, depending on the hosting service and plan.</small>	
Account username:	<input type="text" value="sandy724"/>
Account password:	<input type="password" value="*****"/>
REPOSITORY INFORMATION	
Repository type:	<input type="text" value="Git"/>
Repository plan:	<input type="text" value="Public"/>
<small>The plan for your repository on this hosting service. This must match what is set for your repository.</small>	
Repository name:	<input type="text" value="SandyPuppet"/>
<small>The name of the repository. This is the <repo_name> in http://github.com/<username>/<repo_name>/</small>	

- **Local Repository:** The repository code is checked on the Review Board server. In this case, the administrator has to make sure that the locally configured repository always has the latest code. Once you select the **Repository type** field in the
- **REPOSITORY INFORMATION** section, the rest of the fields change accordingly. If you choose **Git**, you will have to provide the Git clone URL in the **Path** field. The **Mirror path** field should have the local path on the computer where the repository code will be checked out.

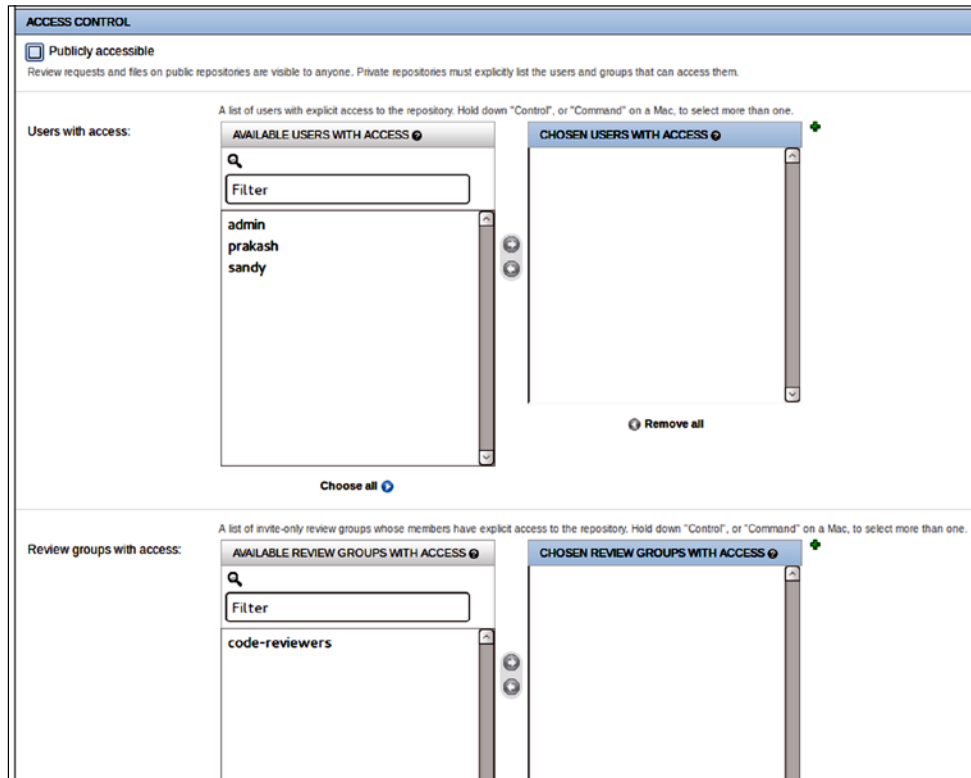
Repository credentials also need to be provided in this section. Changing the repository type will change the fields as well since each repository type has different types of settings.

REPOSITORY HOSTING	
Hosting service:	(None - Custom Repository) ▾
REPOSITORY INFORMATION	
Repository type:	Git ▾
Path:	git@github.com:sandy724/SandyPuppet.git <small>For local Git repositories, this should be the path to a .git directory that Review Board can read from. For remote Git repositories, it should be the clone URL.</small>
Mirror path:	/home/sandy/personal/SandyPuppet
Raw file URL mask:	 <small>A URL mask used to check out a particular revision of a file using HTTP. This is needed for repository types that can't access remote files natively. Use <revision> and <filename> in the URL in place of the revision and filename parts of the path.</small>
Username:	sandy724
Password:	••••••

- **Remote Repository:** Review Board talks directly with your repository server. After you select a hosting service, you have to provide the account details of that hosting service as well. The **REPOSITORY HOSTING** section now has new fields for providing the account details for the Remote Repository. Depending on the selected hosting service, the options in the **Repository type** drop-down list change accordingly, and the other fields change as well. As **GitHub** can host only Git repositories, the **Repository type** field will have only **Git** as an option. The other details to be provided in this example for **Git** repositories are **Repository plan** and **Repository name**. The following table lists all the repository types that can be hosted by a hosting service:

Hosting service	Repository type
Beanstalk	Git, Subversion
Bitbucket	Git, Mercurial
Codebase HQ	Git
Fedora Hosted	Git, Mercurial, Subversion
GitHub	Git
Gitorious	Git
Google Code	Mercurial, Subversion

- **ACCESS CONTROL:** In the **ACCESS CONTROL** section, you can either choose to make the repository available to everybody for reviewing or restrict it to a set of users.



Uncheck the **Publicly accessible** option to restrict repository access or make it private. To associate users to the repository, you have two options: to either give direct access to the users or give access to the review groups. The review groups will list only those groups that have been invited. Users and review groups can be moved between available and chosen multiselect blocks by selecting the users or review groups and clicking on the arrow keys present between these two blocks. At the bottom of these multiselect blocks, **Choose all** and **Remove all** links have been provided to move all the users or review groups between these multiselect blocks in one go.

A user can choose to perform the following three activities depending on the next step to be taken:

- **Save:** This option will add the new repository and take you back to the repositories listing page

- **Save and add another:** Use this option if you want to continue adding other repositories in the system
- **Save and continue editing:** If you would like to add more details about the repository, use this option

The edit page of a repository is similar to the repository add page. In addition, you can also delete the repository.

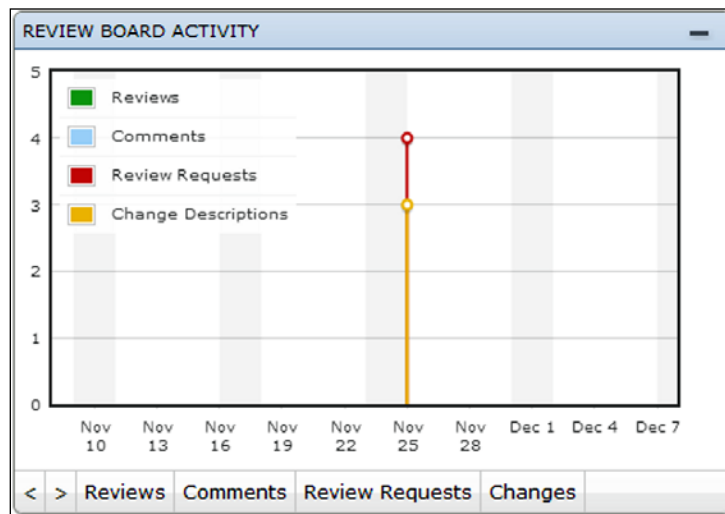
The Admin Dashboard shows various activities happening in the Review Board application. Let's have a look at some of them in the following sections.

Activities

There are several small windows (portlets) on Admin Dashboard that include specific information about a particular thing (or group of things). The admin user can monitor and manage the Review Board application using these portlets. The following sections will cover the different types of portlets.

REVIEW BOARD ACTIVITY

The **REVIEW BOARD ACTIVITY** portlet shows a color-coded graph of **Reviews**, **Comments**, **Review Requests**, and **Change Descriptions** for each day as shown in the following screenshot:

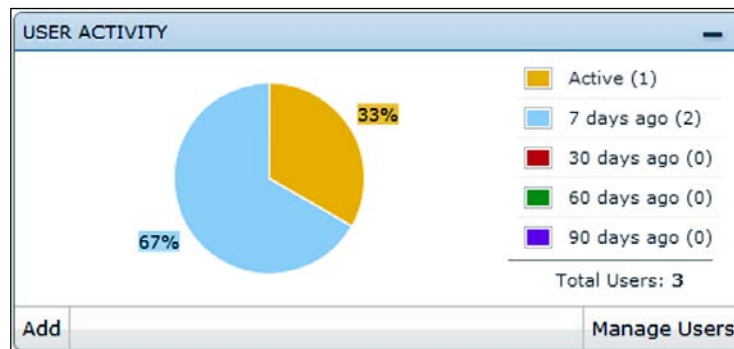


The color legends are displayed by default, but clicking on any one of them will collapse all the legends and show an expandable **Legend** link.

You can move to the previous/next month using the navigation buttons in the left-bottom corner. Also, there are links for enabling/disabling the activities that can be shown in the graph. By default, all of the previously mentioned activities are displayed. Clicking on the link at the bottom of the portlet will remove the activity from the graph. Clicking again on the same link will add it back to the graph.

USER ACTIVITY

The next portlet is **USER ACTIVITY**. This portlet shows the usage of various users as shown in the following screenshot:



The **USER ACTIVITY** portlet shows the total number of users and a pie chart for the number of active users at the moment and the number of active users 7 days ago, 30 days ago, 60 days ago, and 90 days ago.

It has links to add a user and to manage user screens.

RECENT ACTIONS

The **RECENT ACTIONS** portlet is quite similar to the **USER ACTIVITY** portlet, but it shows the chronological list of actions performed by users or review groups.

The screenshot shows the 'RECENT ACTIONS' portlet. It displays a list of actions performed by users or review groups. The list is as follows:

prakash	User
code-reviewers	Review group
admin	User
sandy	User
Reviewer	Group

REVIEW BOARD NEWS

The **REVIEW BOARD NEWS** portlet shows the latest news about new releases from www.reviewboard.org as shown in the following screenshot:

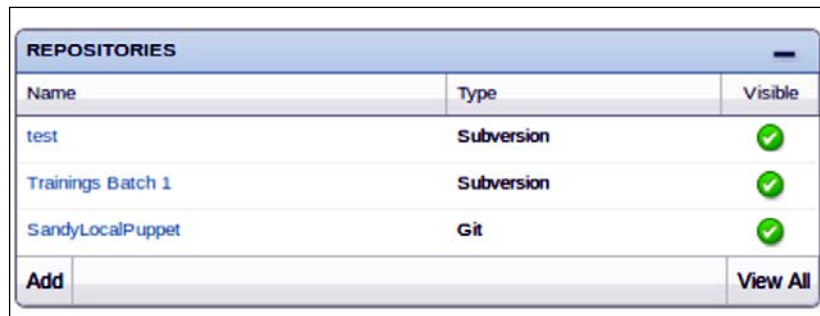


This portlet has the following two links:

- **Reload:** Click on this link to get more latest news
- **More:** Click on this link to see more news items

REPOSITORIES

The **REPOSITORIES** portlet shows the list of existing repositories in the Review Board application. It has links to add a new repository and to view all the repositories.

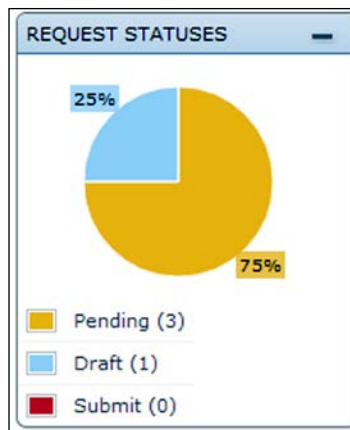


The screenshot shows a portlet titled "REPOSITORIES" with a minus sign in the top right corner. It contains a table with three columns: "Name", "Type", and "Visible". The table has three rows of data. The first row is "test" with "Subversion" type and a green checkmark. The second row is "Trainings Batch 1" with "Subversion" type and a green checkmark. The third row is "SandyLocalPuppet" with "Git" type and a green checkmark. At the bottom of the table are two buttons: "Add" and "View All".

Name	Type	Visible
test	Subversion	✓
Trainings Batch 1	Subversion	✓
SandyLocalPuppet	Git	✓

REQUEST STATUSES

A pie chart is displayed on the **REQUEST STATUSES** portlet representing the number of review requests with each status. It shows the percentage covered by the present status as shown in the following screenshot:



REVIEW GROUPS

The **REVIEW GROUPS** portlet shows the list of review groups configured / set up in the Review Board application, along with the number of users assigned to each group. Clicking on one of the review groups will lead you to the review details page.

A portlet titled "REVIEW GROUPS" with a minus sign in the top right corner. It displays a table with one row: "Code Reviewers" and "2". Below the table are two buttons: "Add" and "View All".

Group Name	Users
Code Reviewers	2

This portlet also has links to add new review groups and see the list of all the review groups set up in the application.

Summary

In this chapter, we looked at repository administration and the portlets for activities, requests, repositories, and review groups. These portlets present a quick overview of various items and can be used as quick links to access more details for each one of the items represented in them.

In the next chapter, we will look at some tips and tricks to optimize the performance of Review Board, and we will also have a look at some handy commands.

9

Advanced Tips and Tricks

We have learned enough to start working with and manage Review Board. There are many things which can be done in the Review Board application to integrate and work with various systems through the support provided by it. In this chapter, we will look at some tips and tricks to make the Review Board application perform better. For this, we will look at the following areas:

- Database
- Extensions
- Optimization
- Search indexing
- Advance commands

We will see how an admin user can manage the entities listed in the preceding list.

Database

On the Admin Dashboard, there is a **Database** link on top of the navigation bar. **Database** in the Admin Dashboard leads to the database management screen. The admin user can manage entries in the database directly from this screen. The links are divided into different sections/groups as shown in the following screenshot:

SITE ADMINISTRATION		
ACCOUNTS		
Local site profiles	Add	Change
Profiles	Add	Change
Review request visits	Add	Change
ATTACHMENTS		
File attachments	Add	Change
AUTH		
Groups	Add	Change
Users	Add	Change
CHANGEDESCS		
Change descriptions	Add	Change
DIFFVIEWER		
Diff set histories	Add	Change
Diff sets	Add	Change
File diffs	Add	Change
DJANGO_EVOLUTION		
Evolutions	Add	Change
Versions	Add	Change
EXTENSIONS		
Registered extensions	Add	Change
HOSTINGSVCS		
Hosting service accounts	Add	Change
REVIEWS		
Comments	Add	Change

The following is a list of the categories and corresponding attributes that can be managed:

Property name	Description
SITE ADMINISTRATION	
ACCOUNTS	
Local site profiles	Local site is a division of Review Board which acts as an independent Review Board app. However, all such sites share the same database. Here, the user can update user profiles and configurations for local sites.
Profiles	These are the user profiles and configurations for the current site.
Review request visits	This tracks the review request visits for the user accounts.
AUTH	
Groups	These are the user groups.
Users	These are the user accounts.
DIFFVIEWER	
Diff set histories	This is a list of diff sets with revision history owned by a review request.
Diff sets	These are file diffs with revision history owned by a diff set.
File diffs	These are diffs with the path and revision of the files.
DJANGO_EVOLUTION	
Evolutions	Database migration history. Not recommended to modify.
Versions	History of database schemas. Not recommended to modify.
REVIEWS	
Comments	Comments on diffs.
Default reviewers	Default reviewers configured for the review requests.
File attachment comments	Comments on file attachments.
Review groups	Reviewer groups.
Review request drafts	Drafts of review requests (not published).
Review requests	All review requests.
Reviews	All reviews.
Screenshot comments	Comments on screenshots.
Screenshots	All screenshots uploaded to Review Board.
Others	
File attachments	File attachments uploaded to review requests.

Property name	Description
Change descriptions	Review request change description logs.
Others	
Registered extensions	All registered extensions.
Hosting service accounts	Configured hosting service accounts.
Repositories	All configured source code repositories.
Tools	Registered tools for communication with repositories.
Local sites	Partitioned local sites.
Site configurations	Configuration for the site. Not recommended to modify.
Sites	Basic site information.

Clicking on the **Add** link leads to the screen for adding a new entry. Clicking on **Change** leads to a screen where a list of the selected types is displayed, and you can search and change a particular entry. The user can select one or more such entries by selecting the checkboxes against each entry, and delete them by choosing to delete in the drop-down list.

Extensions

Review Board is an open source tool and has vast support for extensions through its extension API. More details about extension API can be found at <http://www.reviewboard.org/docs/manual/dev/extending/extensions/>.

All of the extensions are in the form of Python packages, which can be installed using **easy_install**. If there are any specific steps to install a particular extension, they are covered by the installation guide for the extension.

There is an **Extensions** link in the top navigation bar of the Admin Dashboard.

Here you can see all the installed extensions and manage them. The admin user can perform the following operations on the installed extensions, depending on the features supported by them. A button (for these operations) will be displayed for the extension if that operation is supported by the extension:

- **Enable:** This will enable the extension.
- **Disable:** This will disable the extension.
- **Configure:** This will open the configuration screen for the extension.
- **Database:** This will open a screen where the extension's database can be edited. This should be done with care and only if necessary.

Optimization

Now we will talk about optimization and the best settings in terms of hardware, Memcached, and database.

Hardware

Review Board is a CPU-heavy, memory-heavy, network-bound application. Hence, allocating more of these resources makes the performance better. A minimum of four cores is recommended for the Review Board site. It should have at least 2 GB RAM. The disks should be faster, especially for the database server (at least 7200 RPM).

In the Review Board site, database and Memcached can be set up on different servers to have better hardware resources. Also, a load balancer can be put in place to have multiple instances of web servers hosting the same Review Board application. This should help in improving the web server's throughput.

Memcached

Memcached is a high-performing, key-value-based, in-memory caching server. It is significant for Review Board because Review Board caches a large chunk of data, which in turn is expensive to fetch all the time. Hence, providing more memory to Memcached always results in improved performance. The recommended RAM for Memcached is 2 GB. For a larger setup, the memory allocation for Memcached should be even larger. As a practice, you should always set up Review Board with Memcached.

The Memcached memory is provided through the **-m** parameter. For example, `-m 2048` will set the Memcached memory to 2 GB. Each setup has a configuration file to set this up. It can be found by navigating to `/etc/memcached.conf` on Linux.

Database

There are always some configurations, apart from hardware, that can improve the performance of the database depending on the type of database you are using.

For example, you can use query caching and configure the packet size to an optimum level to improve the performance of a MySQL database instance.

Search indexing

We have discussed how to enable the search by installing **PyLucene** in earlier chapters. We have also specified a directory for the search index in that discussion. This search index needs to be rebuilt/updated periodically. You can use any custom way to build this index, and one of the simplest possible ways is to set up a `cron` job using the `search-cron.conf` file available under the `conf` directory in the site directory as follows:

```
$ crontab /path/to/site/conf/search-cron.conf
```

The user running the `crontab` command must have the permission to write to the search index directory. The default `crontab` command will update the index every 10 minutes and will rebuild the full index every Sunday at midnight.

Full indexing is required to use the full text search for the first time; it can be done using the following management command:

```
$ rb-site manage /path/to/site index --full
```

Run the following management command to build the incremental index:

```
$ rb-site manage /path/to/site index
```

Advanced commands

There are some useful commands that you can use to perform management tasks. These syntax to execute the command is as follows:

```
$ rb-site manage /path/to/site command-name - parameters
```

You can get the list of available management commands by running the following command:

```
$ rb-site manage /path/to/site help
```

To get the details about a particular management command, use the following command:

```
$ rb-site manage /path/to/site help command-name
```

For example, you can create a superuser account using the following command without using the Review Board UI:

```
$ rb-site manage /path/to/site createsuperuser
```

It will ask for the username and password for the account. The username should be unique and should not already exist in the database. You should be able to log in to the site as the account has been created successfully.

The following command can be used to fix the review request counts on the dashboard (displayed on the left-hand side menu), which could be incorrect sometimes if the database was edited manually:

```
$ rb-site manage /path/to/site fixreviewcounts
```

The counts are rebuilt when the site is upgraded.

Summary

In this chapter, we went through various tips and tricks to manage the database and extensions and configure the building of search indices as well as some advanced commands.

Index

Symbols

- m parameter 91
- no-db-upgrade option 54
- o/--open command 25
- password command 25
- p/--publish command 24
- .reviewboardrc file 25
- server command 25
- target-groups command 25
- target-people command 25
- username command 25

A

- ACCESS CONTROL section 81
- Active status 69
- activities
 - about 82
 - RECENT ACTIONS portlet 83
 - REPOSITORIES portlet 84
 - REQUEST STATUSES portlet 84
 - REVIEW BOARD ACTIVITY 82, 83
 - REVIEW BOARD NEWS portlet 84
 - REVIEW GROUPS portlet 85
 - USER ACTIVITY 83
- ADD REPOSITORY screen
 - about 78
 - GENERAL INFORMATION section 79
 - REPOSITORY HOSTING 79
- ADD REVIEW GROUP screen
 - about 74
 - ACCESS CONTROL 74, 75
 - GENERAL INFORMATION 74
- Add user link 70
- ADD USER screen 70

- admin control
 - features 12
- Admin Dashboard screen 67
- Administrator E-Mail setting 57
- Administrator Name setting 57
- admin settings
 - category, system information 55
 - category, system settings 55
- advanced commands 92, 93
- All My Requests link 22, 38
- Allow anonymous read-only access
 - setting 58
- Allow code profiling setting 62
- Amazon S3
 - settings, configuring 64
- Authentication Method setting 58
- Authentication page settings
 - Allow anonymous read-only access 58
 - Authentication Method 58

C

- Cache Backend setting 57
- Cache Path setting 57
- CACHE SETTINGS
 - Cache Backend 57
 - Cache Path 57
- Choose Key button 64
- code
 - reviewing 30
- code diff
 - about 16
 - generating 16
- code review
 - about 5
 - benefits 6

- best practices 6
- lightweight code review, performing 7, 8
- need for 7
- post-commit code review 9
- pre-commit code review 8

code reviewing

- Download Diff link 30
- issues, managing 34, 35
- review comments, viewing 33
- Ship It! link 31
- Star mark 30
- View Diff link 32, 33

code review request

- code diff, generating 16
- generated code diff, publishing to Review Board 16, 17, 18
- publishing 15

code review request, publishing with command line

- about 23
- post-commit review 23
- pre-commit review 23
- specific files, reviewing 24
- updating process 24, 25

code review request screen

- Second Block 29
- Third Block 29
- Top Most Bar 28
- viewing 28, 29

Collapse All button 29

conf directory 92

Create Review Request button 18

crontab command 92

D

Database link

- about 88
- SITE ADMINISTRATION 89

Depends On field 29

Diff Viewer 49

Diff Viewer settings page

- about 60, 61
- ADVANCED 61
- Lines of Context 61
- Max diff size (bytes) 61
- Paginate by 61

- Paginate orphans 61
- Show all whitespace for 61
- Show syntax highlighting 61
- Syntax highlighting threshold 61

Download Diff link 30

Download Diff option 20

E

easy_install command 53

edit user page

- PERMISSIONS 72
- PERSONAL INFO 71
- Superuser status 72
- User permissions 72

E-mail 59

e-mail pass-around way 7

E-mail settings

- Mail Server 59
- Password 60
- Port 60
- Send e-mails for review requests and reviews 59
- Send e-mails when new users register an account 59
- Send e-mails when review requests are closed 59
- Sender e-mail address 59
- Username 60
- Use TLS for authentication 60

Enable logging setting 62

Enable search setting 57

Enter key 42

Expand All button 29

Extensions link

- about 90
- Configure option 90
- Database option 91
- Disable option 90
- Enable option 90

F

file server configuration

- Amazon S3 64
- Email TLS Authentication 65
- Host file system 64

- Log Profiling 65
- Public Read-only Access 65
- Review Emails 65
- Server Cache 65
- Server Log 65
- Syntax Highlighting 65

FILTER section 69

First Block 28

full-text search 42, 43

G

general settings

- about 56
- CACHE SETTINGS 57
- SEARCH 57
- SITE SETTINGS 57

Generate an SSH Key setting 64

generated code diff

- publishing, to Review Board 17, 18

I

Incoming Reviews 38

Indexed Search 65

intuitive diff viewer

- features, highlighting 11

Issue Summary section 29, 34

L

lightweight code review

- performing, e-mail pass-around way 7
- performing, over-the-shoulder way 7
- performing, pair programming way 8
- performing, tool-assisted code review way 8

Lines of Context setting 61

Local Repository 79

Log directory setting 62

Logging settings

- Allow code profiling 62
- Enable logging 62
- Log directory 62
- Log level 62

Log level setting 62

Log Profiling 65

M

Mailing list 74

Mail Server setting 59

Max diff size (bytes) setting 61

Media URL setting 57

Memcached 91

Minutes of Meeting (MoM) 7

Mirror path field 79

My Account link 40

N

New Review Request 17

n-voting rule 31

O

optimization

- database 91
- hardware 91
- Memcached 91

Outgoing Reviews

- draft 38
- open 38

over-the-shoulder way 7

P

pair programming way 8

Parent Diff feature 18

Password setting 60

patch 49

PERMISSIONS, edit user page

- Active option 72
- Staff status option 72
- Superuser status option 72
- User permissions option 72

Port setting 60

post-commit code review

- diagram 9

post-review command 23

pre-commit code review

- about 8
- challenge 8
- diagram 8

Publically accessible option 81

Public Read-only Access 65
Publish button 20

Q

quick search
about 42
attributes 42

R

RBCommon
plan type 45, 46
rb-site 50
RECENT ACTIONS portlet 83
Remote Repository 80
Re Open button 35
REPOSITORIES portlet 84
repository administration
Hosting 78
local repository 79
MANAGE section 77
new system adding, links 78
Path 78
remote repository 80, 81
Repository 78
Show this repository 78
REPOSITORY HOSTING section 80
REPOSITORY INFORMATION section 79
REQUEST STATUSES portlet 84
REST API 87
Review Board
about 10
Apache config, updating 52
characteristics 10
dependencies, Django 49
dependencies, Django-Evolution 49
dependencies, Djblets 49
dependencies, flup 49
dependencies, paramiko 49
dependencies, Python Imaging Library 49
features 10, 12
hosting, on RBCommon 45
installation setup 47
installing 48
Review Groups 67
review requests, tracking 12

setting up prerequisites, database 47
setting up prerequisites, Version Control
System client 47
setting up prerequisites, web server 47
setting up, tools 47
setting up, ways 45
site, installing 50-52
upgrading 53, 54
Users 67

REVIEW BOARD ACTIVITY portlet 82, 83

Review Board, features

admin control 12
code files limitations 12
comments 11
easy code review 11
intuitive diff viewer 11
publishing 10
tracking 12
version control systems integration 12

Review Board installation

MySQL database binding, installing 49
patch, installing 49
Python setup tools, installing 48
steps 49
subversion, communicating with 49
ways 48

REVIEW BOARD NEWS portlet 83, 84

Review Groups

Add link 73
Add review group link 73
Display name 73
Invite only 73
Mailing list 73
MANAGE section 72
Name 73
Visible 73

REVIEW GROUPS portlet 85

Review link 31

review request, operations

Close 19
Download Diff 20
Group 21
People 21
Review 20
Ship It! 20
Summary 21
Update 19

- review requests**
 - creating, operations 19-21
 - code review request, publishing through
 - command line 22
 - searching, full-text search 42
 - searching, quick search 42
 - tracking 22
 - viewing 27, 28

S

SEARCH

- Enable search 57
- Search index directory 57
- search index**
 - rebuilding 92
- Search index directory setting 57**
- Second Block 29**
- Send e-mails for review requests and re-views setting 59**
- Send e-mails when new users register an account setting 59**
- Send e-mails when review requests are closed setting 59**
- Sender e-mail address setting 59**
- Server Cache 65**
- Server Log 65**
- Server setting 57**
- Show all whitespace for setting 61**
- Show syntax highlighting setting 61**
- Show trailing whitespace setting 61**

SITE ADMINISTRATION, Database link

- ACCOUNTS 89
- AUTH 89
- DIFFVIEWER 89
- DJANGO_EVOLUTION 89
- Others 89
- REVIEWS 89

Site settings

- Administrator E-Mail 57
- Administrator Name 57
- Media URL 57
- Server 57
- Time Zone 57

SSH settings

- Generate an SSH Key 64
- Upload an SSH Key 64

- Staff status 69**
- Star mark 30**
- Starred Reviews 38**
- Superuser status 69**
- Syntax Highlighting 65**
- Syntax highlighting threshold setting 61**
- system information 65**
- system settings**
 - Authentication 58
 - Diff Viewer 60
 - E-mail 59
 - file storage 64
 - general settings 56
 - Logging 62
 - SSH 63
 - types 55

T

table columns, user dashboard

- Branch 40
- Bugs 40
- Diff Updated 40
- Diff Updated (Relative) 40
- Last Updated 40
- Last Updated (Relative) 40
- My Comments 39
- New Updates 39
- Number of Reviews 40
- Posted Time 40
- Posted Time (Relative) 40
- Repository 40
- Review ID 40
- Ship It! 40
- Starred 40
- Submitter 40
- Summary 40
- To Me 40

Testing Done field 29

Third Block 29

Time Zone setting 57

Top Most Bar 28

U

Upload an SSH Key setting 64

USER ACTIVITY portlet 83

user dashboard

about 37, 38

All My Requests 38

Incoming Reviews 38

Outgoing Reviews 38

Starred Reviews 38

table columns 39

Username setting 60**User Preferences section 41****users**

activities, Save 71

activities, Save and add another 71

activities, Save and continue editing 71

adding, links 70

creating, requirements 70

details, displaying 69

filtering 69

MANAGE section 68

profiles, searching for 70

status, active status 69

status, Staff status 69

status, superuser status 69

tasks 41

Use TLS for authentication setting 60**V****VCS 5****version control system. *See* VCS****View Diff link 32****Y****Your Comment box 33**



Thank you for buying **Getting Started with Review Board**

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.packtpub.com.

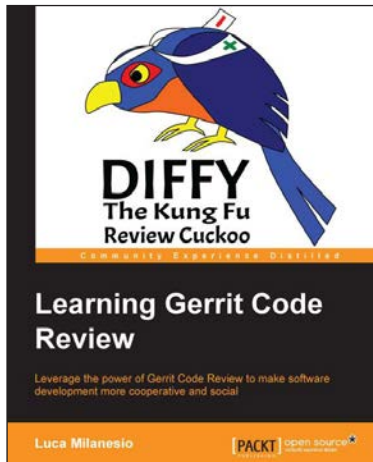
About Packt Open Source

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



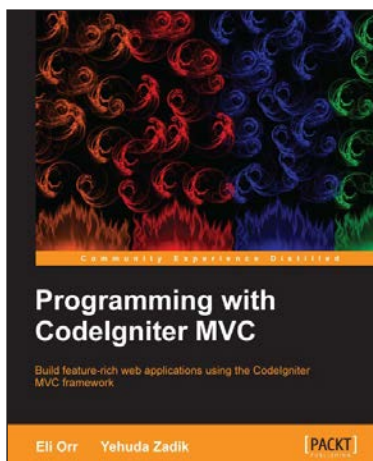
Learning Gerrit Code Review

ISBN: 978-1-78328-947-9

Paperback: 144 pages

Leverage the power of Gerrit Code Review to make software development more cooperative and social

1. Understand the concepts of collective code review using Gerrit through a set of simple examples.
2. Integrate code review functionality into Continuous Integration with Jenkins.
3. Experiment with the code review life cycle in real-life scenarios.



Programming with CodeIgniter MVC

ISBN: 978-1-84969-470-4

Paperback: 124 pages

Build feature-rich web applications using the CodeIgniter MVC framework

1. Build feature-rich web applications using the CodeIgniter MVC framework.
2. Master the concepts of maximum simplicity, separation, flexibility, reusability, and performance efficiency.
3. A quick guide to programming using the CodeIgniter MVC framework.

Please check www.PacktPub.com for information on our titles



Cinder – Begin Creative Coding

ISBN: 978-1-84951-956-4 Paperback: 146 pages

A quick introduction into the world of creative coding with Cinder through basic tutorials and a couple of advanced examples

1. More power – Cinder is one of the most powerful creative coding engines out there and it will be hard to find a better one for your professional grade project.
2. Do it fast – each section should not take longer than one hour to complete.
3. We give you the tools and it is up to you what you do with them – we won't go into complicated algorithms, but rather give you the brushes and paints so you can paint the way you already know.



Sonar Code Quality Testing Essentials

ISBN: 978-1-84951-786-7 Paperback: 318 pages

Achieve higher levels of Software Quality with Sonar

1. Take full advantage of the Sonar platform and its visual components to track code quality and defects.
2. Create step-by-step software quality profiles that match your needs.
3. Real-world examples that use Sonar efficiently to assess quality and improve Java code.

Please check www.PacktPub.com for information on our titles