Redmine Cookbook

In a variety of online project management tools, Redmine markets itself as offering flexibility. Choosing the right management tool can mean the difference between the success and failure of a project.

Whether you are a project manager or system administrator, this book provides valuable recipes to get the best possible performance out of your team, organization, infrastructure, and Redmine itself. Through a series of carefully crafted recipes covering the nitty-gritty of Redmine, you'll be guided through the installation of Redmine, as well as how to fine-tune and customize your Redmine installation. Finally, we walk you through integrating Redmine with other software and databases, such as Tortoise SVN and Visual Studio, and troubleshooting Redmine.

What this book will do for you...

- Make Redmine run on Linux or Microsoft Servers
- Customize Redmine to fit your project management needs and business processes
- Manage multiple projects and teams simultaneously
- Leverage Redmine's features to enhance your team's performance
- Use Redmine for SCRUM and Agile methodologies
- Deploy Redmine for the Service Desk
- Optimize Server performance, configure backups and migrate or upgrade Redmine safely
- Extend Redmine through various plugins

Inside the Cookbook...

- A straightforward and easy-to-follow format
- A selection of the most important tasks and problems
- Carefully organized instructions to solve problems efficiently
- Clear explanations of what you did
- Solutions that can be applied to solve real-world problems

Quick answers to common problems

Redmine Cookbook

Over 80 hands-on recipes to improve your skills in project management, team management, process improvement, and Redmine administration

Aleksandar Pavić


$ 44.99 US
£ 28.99 UK

Prices do not include local sales tax or VAT where applicable

Free Sample
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Installing and Running Redmine'
- A synopsis of the book’s content
- More information on Redmine Cookbook
About the Author

Aleksandar Pavić began his first project as a student at the Faculty of Technical Sciences, Novi Sad, Serbia where he developed Technical Faculty's first website. He later assembled a web team and graduated with BScs in computers sciences and information technology, and an MSc in product lifecycle management at the same university.

Acting mainly as a project manager and sometimes as a developer or team leader, Aleksandar made the following notable projects possible: student nourishment IS and ERP with smart cards at the University of Novi Sad, Novi Sad Farmer Market IS and ERP, E-government system of Novi Sad, including various sub-services, a web portal of the City of Novi Sad, Los Angeles-based Usamailagent.com packet-forwarding web-application, Unival-logistics.com online business system.

Currently, Aleksandar is employed as head of the IT services department at PUC Informatika, Novi Sad. He is involved in the following EU-funded Projects here: Sociotal, Weelive, and CLIPS.

Acting as an entrepreneur, he stands behind www.redminegit.com, cloud hosting, and Redmine implementations.


While participating in projects, he acts as an open source evangelist, and he advocates usage of lean methodologies backed up by Redmine as the management software of choice.

Aleksandar uses various management skills, such as, project management, product and service management, ISO 27000 and ITIL, Scrum and Agile methodologies, on a daily basis. He is also profi cient in PHP, CakePHP, Ruby, Ruby on Rails, JavaScript, Bootstrap, C#, jQuery, Apache, Linux, Ubuntu, CentOs, Nginx, Phusion Passenger, Node.js, HTML5, Canvas, CentOS, Windows Servers, IIS, MySQL, PostgreSQL, and Microsoft SQL Server.
Preface

Redmine is probably the most underused collaboration and management tool existing on the open source market for 10 years. Decision makers within corporations are often unaware of Redmine's full potential and ability to expand in all areas of business just by performing several actions inside Redmine and agreeing to several conventions within the company.

Almost all departments of a company can store its documents, collaborate, coordinate, and benefit from its usage over an extended period of time because in every business it's crucial to know who did (or who is going to do) what, when, and why. So for example, in software development, one Redmine ticket within feature tracker can tell us who is going to do what—fulfill the client's request, such as export report to Excel; why—because it is the customer's request from within project A; when—the date when tasks were scheduled or done. Additionally, we can take a look at how it's done using issue-code relation. However, this should not just be limited to software if it's a design project, for example. Designers can also attach their design to a particular Redmine ticket/task if they use repository to store files. If it's a janitor company, then they can store photos of before and after repair, and so on.

Instead of just installing Redmine and using its features out of the box, this book tries to teach readers to think outside the box, and customize Redmine to improve user experience, customize workflows, and harness the power of its flexible design.

Choosing the right management tool can mean the difference between success and failure of a project. Flexible project management tools bend themselves to fit your needs. Whether it's a simple project communication, collaboration, or more complex project methodology, such as SCRUM, or issue-code relationship, or different methodology for each project, this book will teach you how to quickly customize Redmine for maximal business benefits and user satisfaction. It goes even further than project management and collaboration, illustrating how Redmine's flexible trackers could be used for automated recurring tasks and processes. Additionally, readers are advised to visit the book's website at http://www.redminecookbook.com and take a look at the blog and frequently asked questions section.
Redmine is open source and donation supported, driven mostly by
volunteers who donate their skills and time toward the project. They also
need to pay for servers, computers they develop on, and so on. Consider
donating to the project through the following link:
http://www.redmine.org/projects/redmine/wiki/Donors

What this book covers

Chapter 1, *Installing and Running Redmine*, provides several different ways to obtain and
install Redmine. It then goes on to elaborate on how to run it with different servers, including
Nginx, Puma, Apache, Thin, IIS, and Phusion Passenger with MySQL or PostgreSQL on Linux
servers or in a complete Microsoft environment with Microsoft SQL Server and IIS on Windows
Server 2012.

Chapter 2, *Customizing Redmine*, covers some basic Redmine customizations, such as
customizing roles, homepage, extending projects with custom fields, and so on.

Chapter 3, *Project Management with Redmine*, shifts more toward management. This chapter
teaches readers how to achieve some basic project management routines, such as splitting
projects into phases or subprojects, using template projects, tasks and sub-tasks, creating
reports. It then moves toward a bit more complex stuff, such as using and forcing issue-code
relationship, and defining a roadmap to the release plan.

Chapter 4, *Improving Team Performance*, also focuses on management and team-oriented
scenarios, such as organizing and keeping project documentation inside Redmine, putting the
timeline to good use, making sure that everyone is informed, improving team performance,
managing teams in different time zones, and one scenario of SCRUM and Redmine usage.

Chapter 5, *Regular and Planned Maintenance*, is written for system administrators, and
deals with recipes, such as migration, upgrade, backup, and recovery that are required for
Redmine's usage in production over an extended period of time.

Chapter 6, *Performance and System Tuning*, mixes recipes from system administration,
Redmine tuning, and management to improve Redmine's performance and user experience by
providing how-tos for new project creation, workflows, auto-login features, server tuning, and
integrating Redmine with Active Directory.

Chapter 7, *Integrating Redmine with Other Software*, teaches readers how to interact with
Redmine directly from TortoiseSVN, TortoiseGIT, or Visual Studio; or to integrate your own
software or website with Redmine through API; use mobile applications, and so on.

Chapter 8, *Getting the Most Out of Scripts and Plugins*, deals with some Redmine plugins and
scripts, such as pasting images from the clipboard, implementing and using recurring tasks,
practicing Kanban, being Agile, using Redmine with Jenkins, and using CKEditor.
Chapter 9, Troubleshooting, provides solutions to common Redmine issues, such as installations, slow responses from server, how to get help faster, troubleshooting plugin installation, and so on.

Chapter 10, Making the Most of Redmine, is mostly business- and management-oriented, while also keeping some interesting content for Redmine administrators, such as improving Redmine security, using and configuring service desk plugins and custom queries, and wiki security.
1

Installing and Running Redmine

In this chapter, you will learn how to install and run Redmine in different ways and on different platforms. This chapter covers the following recipes:

- Default installation on an Ubuntu server
- Installing from a source on Ubuntu
- Installation on Windows servers
- Using PostgreSQL as a database
- Using Puma and IIS on Windows
- Running with Apache as mod_fcgI
- Running Redmine with Phusion Passenger
- Running Redmine with Nginx and Thin
- Installing optional requirements
- Using custom Ruby for Redmine
Introduction

Redmine is a project-management web application that is flexible in many ways, including its installation and running it. There are multiple configurations of web server software on which Redmine can run. These also include different operating systems, databases, and Ruby versions. Information on supported operating systems, web server software, or Ruby programming language interpreter and supported rails framework versions can be found at the Redmine website on the installation page:

http://www.redmine.org/projects/redmine/wiki/redmineinstall

This chapter presents recipes that are based on several common choices to install and run Redmine in production environments.

This chapter's recipes are deliberately split between installing and running Redmine because it can be installed and run in multiple ways.

One of the most common configurations to run Redmine on is based on a Linux operating system, MySQL database, Phusion Passenger, and an Apache web server. This is the most common commercial offered by various hosting companies as a shared hosting at affordable prices. Also, there are already prebuilt virtual machine images for popular hypervisors, such as VMware or VirtualBox, that are offered by Bitnami and Turnkey Linux, which let you run Redmine simply by downloading the virtual machine image and turning it on; no configuration is required. However, such an approach may be an issue when you decide to go further with Redmine, install plugins, troubleshoot, upgrade, and so on.

If you are planning to use Redmine in your company or team, this chapter provides recipes that make sure that you have everything under control, and you will be able to migrate/upgrade, backup, fine-tune Redmine, and so on, because these may be required as time goes by.

As Redmine is flexible and can be used for different kinds of projects, you need to plan your installation carefully. If, for example, you are installing Redmine for a team of designers, they may want to attach large image files to tasks, and you need to plan the server's storage accordingly. Another scenario may be that if your company is going to use Redmine’s flexible trackers as a service desk and there are going to be multiple-concurrent users, you are going to need more RAM memory and CPU power. The same goes for database configuration; Redmine will work on SQLite, but in larger production environments where performance is an issue, SQLite will present a problem.

While the recipes in this chapter provide step-by-step installation and configuration instructions, readers are encouraged to adopt them or use different operating systems and servers that may better fit their needs.
Default installation on an Ubuntu server

At the time of writing this book, the actual Ubuntu server version is 14.04 Long Term Support (LTS). So, this recipe covers installation on 14.04.3; however, things are probably not going to change much in the next few releases, and this is applicable to older versions of Ubuntu as well.

Getting ready

Make sure that you have sudo or root access to the server. If you don't have them, you may want to jump to Installing from a source on Ubuntu recipe, which explains how to install and run Redmine from the user's home directory.

1. First of all, make sure that your system is up-to-date by running the following commands:
   
   `sudo apt-get update && sudo apt-get upgrade`

2. Then, we need some prerequisites, such as MySQL or PostgreSQL, for the database. This recipe uses MySQL:
   
   `sudo apt-get install mysql-server mysql-client`

3. During installation of MySQL you will be asked to enter a password for the MySQL root user. Write down this password, you are going to use it later.

How to do it...

After updating your system, go ahead and try a simple install:

```
sudo apt-get install redmine-mysql redmine
```

After running this command on a blank Linux server box, you may get a large number of dependencies to install. Just click <Yes> or press ENTER.
This process is going to take some time depending on your server and network capacity. The next screen that you get asks you to configure Redmine automatically; choose <Yes>.

If prompted for database configuration, choose MySQL. On the next several screens, provide the administration password that you wrote down and the database name for your Redmine installation; then, your username and password follows, which are used by Redmine.

After providing these credentials (which you should write down somewhere safe) and waiting for the apt-get script to finish, you'll find your Redmine installed in the following folder:

/usr/share/redmine

To find out which version you installed, run the following:

more /usr/share/redmine/lib/redmine/version.rb
You will get something like this:

![Image of code](image)

Your exact version is either Major, Minor, or Tiny, and in this case, it is Redmine 2.4.2 (shipped with Ubuntu 14.04.3). You will notice that the same version is used for Ubuntu versions. This rule applies for most open-software projects nowadays.

**Accessing your Redmine**

Installing Redmine on an Ubuntu system was easy and straightforward; however, accessing it via HTTP is not that easy and straightforward a task, and it gives you many options, as provided later in other recipes that deal with running Redmine on dedicated web server software.

To test the installation, perform the following:

1. Navigate to the Redmine installation directory:
   ```bash
   cd /usr/share/redmine
   ```
2. Run the **WEBrick** server to test the installation:
   ```bash
   sudo ruby script/rails server webrick -e production
   ```
WEBrick can be used in production, but it is highly recommended that you read other recipes in this chapter and choose a more advanced and reliable solution, such as Apache or Nginx, to run Redmine.

**How it works...**

Apt-get installations are supposed to help administrators save time on common administration tasks. Aptitude is a Linux package manager that comes shipped with Ubuntu and Ubuntu-like systems; it downloads precompiled binaries and runs configuration scripts. In this case, first we update and upgrade our Ubuntu system, then we install MySQL and Redmine. Aptitude automatically installs the required dependencies and configures the system for us.

Please keep in mind that while this may be Ubuntu's default and easy installation way, it installs Redmine to be run as a root user, and running web-applications exposed to the Internet as the root user is not a good idea. This is because as time goes by, hackers might become aware of security holes and use them to hack your entire server.

**There's more...**

If you want to install a newer version of Redmine than the default one that is provided in the official Ubuntu channels or you want to update an existing one, you will want to add the following repository to your system. To add it, run the following:

```
sudo add-apt-repository ppa:ondrej/redmine
sudo apt-get update && sudo apt-get upgrade
```

You must run this step first. However, the ondrej/redmine repository does not always keep track of the latest Redmine releases. If you want to have the latest release and easily update it, read the next recipe.

**See also**

For more information on WEBrick server take a look at its documentation at [http://ruby-doc.org/stdlib-2.0.0/libdoc/webrick/rdoc/WEBrick.html](http://ruby-doc.org/stdlib-2.0.0/libdoc/webrick/rdoc/WEBrick.html).
Installing from a source on Ubuntu

There are two common ways to obtain a Redmine source: through Subversion Client (SVN) or by downloading the compressed source code from a website.

Also, there are two common ways to install Redmine: under a Linux user account, or system-wide. The previous recipe installed Redmine system-wide. This recipe covers the installation of Redmine under an ordinary user account in the user's home directory, which is the recommended way to install Redmine.

Getting ready

When downloading and installing a custom version of Redmine, make sure that you have the required prerequisites installed on your Ubuntu server. At the time of writing this recipe, the current version of Redmine is 3.2.0. You will find the list of supported prerequisites on the Redmine homepage:

http://www.redmine.org/projects/redmine/wiki/RedmineInstall

If you are using Ubuntu 14.04.03, then you are ready to go with Redmine 3.2.x; if you install Ruby and Rails, use the following command:

```
sudo apt-get install ruby ruby-railties-4.0 ruby-dev build-essential zlib1g-dev libmysqlclient-dev
```

Use the following command to check your Ruby and Rails version type:

```
ruby -v
ruby 1.9.3p484 (2013-11-22 revision 43786) [x86_64-linux]
rails -v
Rails 4.0.2
```

On the console output, you can read your versions and compare them to the supported ones listed on the Redmine website. Currently, we can confirm that we are ready to go with Redmine 3.2.0, as follows:
Installing and Running Redmine

Database configuration

You also need to have a MySQL, PostgreSQL, or SQLite database that is going to be used with Redmine. If you are creating a MySQL, or PostgreSQL database manually, make sure that you create a UTF-8 database with a UTF-8 general_ci collation. To create a MySQL database, perform the following:

1. Login to MySQL with a user that has sufficient privileges to create databases:
   \texttt{mysql -u root -pmy\_password}
2. Create a UTF-8 database as follows; please keep in mind that you can choose any database name:
   \texttt{CREATE DATABASE redmine CHARACTER SET utf8 COLLATE utf8\_general\_ci;}
3. Create a user with a password (write it down in a safe place):
   \texttt{CREATE USER 'redmine'@'localhost' IDENTIFIED BY 'my\_password';}
4. Give your user privileges, as follows:
   \texttt{GRANT ALL PRIVILEGES ON redmine.* TO 'redmine'@'localhost';}

How to do it...

First, let's confirm that we are not using the system as a root user by opening the console and typing the following: \texttt{whoami}

The output should be some other username than root.

Obtaining the Redmine source files

Firstly, obtain the Redmine source, either by downloading it to your computer, unpacking it, and uploading it to \texttt{/home/your\_user/redmine}, or by following the methods that are illustrated next.

Downloading and extracting the Redmine source files

The \texttt{wget} tool is installed by default on the Ubuntu server; so, in order to use it to get the Redmine source, we perform the following tasks:

1. Navigate to your user's home directory by typing the following:
   \texttt{cd ~}
2. Utilize the \texttt{wget} tool to obtain a compressed Redmine source, as follows:
   \texttt{wget http://www.redmine.org/releases/redmine-3.2.0.tar.gz}
3. Unpack the archive:
   `tar xvf redmine-3.2.0.tar.gz && mv redmine-3.2.0 redmine`

4. Remove the downloaded archive file:
   `rm redmine-3.2.0.tar.gz`

**The SVN checkout method**

Many administrators who plan to upgrade Redmine often prefer to grab the code via SVN because it allows code to be automatically updated while preserving local changes with a simple SVN update command. To use this method, perform the following steps:

1. Navigate to your user's home directory by typing the following:
   `cd ~`

2. Create a directory for Redmine, as follows:
   `mkdir redmine`

3. Check out Redmine from its official repository, as follows:
   `svn co https://svn.redmine.org/redmine/branches/3.2-stable redmine`

**Redmine installation**

After any of the previous methods, you should end up with the latest Redmine source in your home directory:

1. To proceed with the installation, use the following command:
   `sudo gem update && sudo gem install bundler`

   If you are behind a proxy, the command to run `gem` commands behind a proxy is as follows:
   `gem command_name --http-proxy http://your_proxy:port`

2. Copy the default database settings to a file without the `.example` extension:
   `cp redmine/config/database.yml.example redmine/config/database.yml`

3. Edit the database settings using your favorite editor, in this case, `nano`:
   `nano redmine/config/database.yml`
4. You can edit production settings, or those for any other environment type, 
   (for production, replace the values with your credentials):
   
   ```
   production:
   adapter: mysql2
   database: redmine
   host: localhost
   username: redmine
   password: my_password
   encoding: utf8
   ```

5. Now, let's run bundler from the directory from which we extracted Redmine:
   ```
   bundle install --without development test postgresql sqlite rmagick
   ```

6. After completing the bundler installation, you need to run several commands that are 
   required by Redmine, as follows:
   ```
   bundle exec rake generate_secret_token
   ```

7. Proceed by populating the Redmine database as follows:
   ```
   bundle exec rake db:migrate RAILS_ENV=production
   ```

8. Load the default data, and choose a proper locale for default data:
   ```
   bundle exec rake redmine:load_default_data RAILS_ENV=production
   ```

9. Now, create the required directories:
   ```
   mkdir -p tmp tmp/pdf public/plugin_assets
   ```

10. Change permissions accordingly:
    ```
        chmod -R 755 files log tmp public/plugin_assets
    ```

11. Now, you can test the installation by running Redmine through a rails server:
    ```
        bundle exec rails server webrick -e production -b your_server_ip
    ```

12. Test whether Redmine is properly installed by opening the browser and typing
    ```
        http://your_server_ip:3000/. The Redmine home page should be loaded.
    ```
How it works...

This method assumes that Redmine will be installed in the user's /home/username/redmine directory. It can be used on shared hosting accounts if the Redmine prerequisites are already installed. At first, we made sure that the prerequisites were installed, then we grabbed the Redmine source either by downloading and extracting it manually or by SVN checkout. Then, we updated Ruby gems and installed bundler. Gem is a package manager for Ruby, and it is used to download and install the required Ruby libraries from the Internet. Bundler makes sure that all the gems required by an application are downloaded and installed. Then, we configured credentials for database access (in this case MySQL) and proceeded with bundling Redmine. The bundle command in Step 5 fetches gems that are specified in Gemfile, omitting the gems that are required for development, test, postgresql, sqlite, and rmagick. Then, we called bundle exec rake generate_secret_token, which generated the hashing code that was required for cookie authentication. After this, we created database tables, and we populated them with the default data that is language/locale-dependent. The last step was to create the necessary folders and set permissions. At the end, we tested the Redmine installation with WEBrick.

See also

This recipe taught you how to install Redmine in the home directory for a user using system-wide Ruby. Take a look at the final recipe Using custom Ruby for Redmine if you need custom Ruby and can't tamper with the system at all (usually on shared hosting). Also, you now need a server in front of your Redmine installation. Later recipes will teach you how to use Apache, Nginx, or Puma as web-servers.

You can find alternative methods that are customized for different operating systems on the Redmine website:

http://www.redmine.org/projects/redmine/wiki/HowTos

Installation on Windows servers

This recipe teaches you how to install Redmine on Windows servers. It covers the Windows 2012 R2 Standard version and Microsoft SQL Server versions 2008 or later. However, this recipe can most certainly be applied to other Windows server versions. Also, PostgreSQL and MySQL can be used instead of Microsoft SQL Server (MSSQL).
Getting ready

Make sure that the Windows server is properly installed with all the default libraries and required drivers. This recipe is based on Microsoft SQL server, and it assumes that you already have it installed. If you need to install it, make sure that you add the proper roles to the server first and include .NET 3.5 (required for SQL server 2014). Any type of MSSQL can be used (Express, Standard, Enterprise, and so on). Prepare a database named Redmine, and create a user for it. Prior to your Redmine installation, make sure that you have enabled SQL Server's TCP IP connectivity options by following the instructions that are provided here:

http://dba.stackexchange.com/questions/62165/i-cant-connect-to-my-servers-sql-database-via-an-ip-address

This will ensure that you have prepared your SQL server successfully.
How to do it...

1. First we need to install Ruby:
2. Open your browser and navigate to http://rubyinstaller.org/.
3. Download **Ruby 2.0.0-p645 (x64)**.
4. Right-click the Ruby installer that you downloaded and click **Run as Administrator**.
5. Proceed with the installation and make sure that you don't have any spaces or special characters in the installation path due to some potential issues with some Ruby third-party libraries. So, let's choose `C:\ruby\ruby2` as a path:

![Screenshot of Ruby installer setup](image)

After a successful installation, in your Start menu, a new shortcut named **Start Command Prompt with Ruby** should be visible. Once clicked, you can type `ruby -v` and confirm that Ruby is successfully installed.
Next, we need to install the DevKit 4.7.2 minigw 64-bit version by performing the following steps:


   Make sure you are downloading the proper DevKit, it needs to match your Ruby version and system architecture: 64-bit or 32-bit.

2. Extract it to C:\ruby\devkit.

3. Run Command Prompt and navigate to C:\ruby\devkit by typing the following command:
   ```
   cd C:\ruby\devkit
   ```

4. Type the following command:
   ```
   ruby dk.rb init
   ```

5. Review what's initialized by typing:
   ```
   ruby dk.rb review
   ```

6. If you stumble upon a problem and get a message such as **Invalid configuration. Please fix 'config.yml'.**, then you need to edit C:\ruby\devkit\config.yml with Notepad and enter a line such as C:/ruby/ruby2 so that when running ruby dk.review, you will get a screen like this:

   ![Command Prompt output](image.png)

7. Type the following command:
   ```
   ruby dk.rb install
   ```

   When this is done, you will be informed that DevKit is installed.

8. Run the following:
   ```
   devkitvars.bat
   ```

9. Update gem, as follows:
   ```
   gem update
   ```
10. Install `bundler` by typing the following:

   `gem install bundler`

11. Download and extract Redmine to `C:\ruby\redmine`

   This recipe uses Redmine 3.1.0, which is tested with this recipe, but probably newer versions will also work.

12. Rename `C:\ruby\redmine\config\database.yml.example` to `database.yml`.

13. Edit `C:\ruby\redmine\config\database.yml`. You will find the MSSQL sample configuration at the bottom of the file, but it's enough just to edit the first production configuration in `database.yml` so that it looks like this:

   ```
   production:
     adapter: sqlserver
     database: redmine_db
     host: 127.0.0.1
     username: redmine_ms_sql_user
     password: "redmine_db_password"
   ```

   Replace the values in this code with configuration values that fit your server's IP address and database credentials.

14. Start the installation with `bundler` by typing the following command:

   `bundle install --without development test rmagick mysql postgresql`

15. Generate session storage encryption, as follows:

   `bundle exec rake generate_secret_token`

16. To create a database tables—objects, use the following code:

   ```
   set RAILS_ENV=production
   bundle exec rake db:migrate
   ```

17. Load the default data to the database. You can choose your language during or prior to this command. If you know that there is an existing translation of Redmine for the default language that you are choosing, you can set a two-letter country code by the Windows `set` command (https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2):

   ```
   set REDMINE_LANG=rs
   bundle exec rake redmine:load_default_data
   ```
Installing and Running Redmine

18. Test the installation by typing the following:

   `bundle exec rails server webrick -e production`

If everything is okay, you should get a screen that looks like this:

![Screen capture of a command prompt showing the output of `bundle exec rails server webrick -e production`]

After this, Redmine should be accessible via `http://localhost:3000`.

How it works...

For a Windows installation to work, we need a proper Ruby version and a Microsoft SQL server. It is not obligatory to use the Microsoft SQL Server, but this recipe uses it just from the perspective of trying to stick with Microsoft technologies. First, we set up a blank database and user for Redmine on the MSSQL server, and we make sure that it is working and able to connect via TCP/IP correctly. After this, we download and install a precompiled binary Ruby and DevKit that fits our chosen Ruby version. Then, we update Ruby gems and install bundler. After downloading and configuring database parameters for Redmine, we proceed by bundling Redmine, generating a session store token, and populating the database. After this, our installation is done, and we can test it with WEBrick.

See also

Now that you have Redmine successfully installed, running it requires a different recipe. To run Redmine on Windows, there are a few options that you can use, such as Apache + Fcgi, Nginx, or Puma.

The next recipe Using Puma and IIS on Windows is a very nice and flexible way to run Redmine.

Using Puma and IIS on Windows

Puma is advertised as a small and fast server. It is derived from Mongrel, which is an open source web server written in Ruby by Zed Shaw. It can serve Redmine on its own, or behind Nginx, Apache, or IIS. Puma can be run and installed in a user's home directory or system-wide.
Chapter 1

Getting ready

First, install Redmine as explained in the previous recipe.

Then, we need the OpenSSL Developer Package (this contains header files and binaries), which can be downloaded from http://packages.openknapsack.org/openssl/openssl-1.0.0k-x64-windows.tar.lzma. Considering that we followed the previous recipe precisely, if you need a different SSL version, you can obtain it from https://www.openssl.org. Now, perform the following steps:

1. Download http://packages.openknapsack.org/openssl/openssl-1.0.0k-x64-windows.tar.lzma and copy it to C:\ruby.
2. Create a folder, openssl, in C:\ruby and copy the downloaded OpenSSL lzma archive here.
3. Run cmd and navigate to C:\ruby\openssl by typing the following: cd C:\ruby\openssl. Extract the content of the archive by typing:
   bsdtar --lzma -xf openssl-1.0.0k-x64-windows.tar.lzma

   If mingw bin from devkit is not added to path, you must specify full folder in order to execute bsdtar, and it would look somewhat like this:
   c:\ruby\devkit\mingw\bin\bsdtar.exe --lzma -xf c:\ruby\openssl\openssl-1.0.0k-x64-windows.tar.lzma

4. You should end up with OpenSSL files extracted in C:\ruby\openssl.

How to do it...

Once you have installed Redmine and its prerequisites, as explained in this recipe, proceed by installing the Puma server by typing the following:

gem install puma -- --with-opt-dir=c:\ruby\openssl

Testing Puma

This recipe assumes that your Redmine is installed, as explained in the, Installation on Windows servers recipe.

Run the following command from Redmine's directory in the command prompt:

puma -e production -p 3000
You should get a screen that looks like the following:

Navigating to http://127.0.0.1:3000 on your browser should open Redmine screen.

**Configuring Puma to start with Windows**

To have your Puma server started automatically with Windows, perform the following steps:

1. Create a file, pumastart.bat, in C:\ruby with the following contents:
   ```
   cd C:\redmine
   start /min puma -e production -p 3000 -t 8:32
   ```
2. Then go to Server Manager | Tools | Task Scheduler | Create a Task.
3. Check the Run whether user is logged or not, Run with highest privileges, and Hidden checkboxes.
4. Then in Actions, go to New | Start a program and find pumastart.bat.
5. On the Triggers tab, click New and choose Begin the task: At startup (located in the top dropdown).

**Configuring IIS**

You should add an IIS role to your server and install the following two add-ons to IIS. You can install them directly from the Microsoft website.

You can get the URL Rewrite from http://www.iis.net/download/URLRewrite and the reverse proxy from http://www.iis.net/download/ApplicationRequestRouting.

1. Open IIS Manager.
2. Navigate to the website that you want to use as a Redmine proxy.
3. Click the URL Rewrite icon.
4. Right-click inbound rules list.
5. Select Add Rule and choose Reverse proxy.
6. Add an Inbound rule with the following: 127.0.0.1:3000.
If your Puma server runs after clicking **OK**, you should be able to type `http://localhost` or whatever your server name is and you will get the Redmine welcome screen.

**How it works...**

At first, we needed some prerequisites to install Puma because the `gem` to install Puma compiles it on your machine, and it requires proper OpenSSL library headers and binaries to compile. This is the reason why OpenSSL and dev tools are required. Then, the Puma installation is tested just by typing `puma -e production -p 3000`. To ensure that Puma starts after the Windows server restarts, Task Scheduler is used, and it schedules Puma to start on boot through a BAT file. In a bat file, `command /t 8,32` tells Puma to start with a minimum of 8, and a maximum of 32 threads. You can adjust these values to fit your configuration. After this, we installed two Microsoft original modules to the IIS server and added a reverse proxy rule to forward all requests to `127.0.0.1:3000` where the Puma server is listening. We used the default IIS site, but this rule works with any or multiple IIS sites.

**There's more...**

This recipe can be easily adopted to use Puma behind Nginx or Apache on both Windows and Linux systems. Also Thin or Unicorn can be used instead of Puma.

**See also**

Check out the Puma website for updates, additional configurations and fine-tuning:

http://puma.io/

**Running with Apache as mod_fcgid**

The `mod_fcgid` module is a high-performance alternative to `mod_cgi` and handles all requests for programs (such as Ruby) that need to be executed on the server. This recipe may not be a very popular choice due to the fact that many standalone Ruby servers such as Thin, Puma, and Phusion Passenger now exist. However, it may be required, and it's very reliable and relies only on Apache and Ruby executables.

**Getting ready**

- Install Redmine as explained in *Installing Redmine from a source on Ubuntu* to the home directory.
- Install Apache and `mod_fastcgi` by typing the following:

  ```bash
  sudo apt-get install apache2 libapache2-mod-fcgid
  ```
Installing and Running Redmine

How to do it...

Once the prerequisites are installed, and Redmine is confirmed as working with WEBrick, then navigate to /home/youruser/redmine/public and type the following commands:

```
    cd /home/youruser/redmine/public
    mv dispatch.fcgi.example dispatch.fcgi
    chmod 755 dispatch.fcgi
    mv htaccess.fcgi.example .htaccess
```

Create a new virtual host for Apache, or edit the default host. We will create a new host called redmine.yoursite.com:

```
    sudo nano /etc/apache2/sites-available/redmine.yoursite.com
```

Enter the following contents:

```
<VirtualHost *:80>
    ServerName redmine.yoursite.com
    DocumentRoot /home/youruser/redmine/public
    <Directory /home/youruser/redmine/public >
        Options Indexes ExecCGI FollowSymLinks
        Order allow,deny
        Allow from all
        AllowOverride all
    </Directory>
    ErrorLog /var/log/apache2/yoursite_error.log
    CustomLog /var/log/apache2/yoursite_access.log combined
</VirtualHost>
```

Enable your new website and restart Apache:

```
    sudo a2ensite redmine.yoursite.com
    sudo service apache2 restart
```

After restarting, your Redmine should be ready and accessible through http://redmine.yoursite.com provided DNS is set properly or at least hosts file for testing purposes.
How it works...

At first, we installed mod_fcgid for Apache, then we prepared dispatch.fcgi; .htaccess, dispatch.fcgi is used by mod_fcgid through Apache and Ruby to run Redmine. htaccess tells Apache what to do with dispatch.fcgi. After this, we created a new virtual server for Apache. This may be done through an Apache management tool, such as Webmin for example, but then the value needs to be entered through an entry form or edited like we did from the command line in this recipe. After creating a virtual server, we enabled it and tested Redmine.

There's more...

If you get an error such as, Rails application failed to start properly, try some of the troubleshooting recipes from Chapter 9, Troubleshooting (the Troubleshooting Apache installations section.)

See also

The official mod_fcgid website is http://httpd.apache.org/mod_fcgid/.

More on Apache administration can be found at https://httpd.apache.org/docs/2.2/vhosts/name-based.html.

Running Redmine with Phusion Passenger

Phusion Passenger is a Ruby application server that was originally designed to run web applications (such as Redmine) that were built on the Ruby on Rails framework. Nowadays it has evolved, and besides Ruby applications, it supports Python and Node.js, making it a good candidate for various use cases.

This recipe is written for the Ubuntu 14.04 server.

Getting ready

Make sure that you have Apache and passenger installed:

    sudo apt-get install apache2 libapache2-mod-passenger
Installing and Running Redmine

How to do it...

To configure Passenger, perform the following steps:

1. Open /etc/apache2/mods-available/passenger.conf with your favourite editor, or use nano:
   
nano /etc/apache2/mods-available/passenger.conf

2. Add the following line: PassengerDefaultUser www-data.

3. So, your passenger.com will look something like this:
   
   ```
   <IfModule mod_passenger.c>
     PassengerRoot /usr/lib/ruby/vendor_ruby/phusion_passenger/
layers.ini
     PassengerDefaultRuby /usr/bin/ruby
     PassengerDefaultUser www-data
   </IfModule>
   ```

   If you are using a different Ruby version installed via RVM, update the paths accordingly. Also replace www-data with your user.

   Please keep in mind that PassengerDefaultUser can be entered in your virtual server’s config file, which you may need to edit manually for installations on shared hosting.

4. Create a symbolic link to the Redmine public folder from a web server’s root folder like this:
   
   ```
   sudo ln -s /usr/share/redmine/public /var/www/html/redmine
   ```

5. This link assumes that you installed Redmine through apt-get. If this is not the case and you installed it somewhere else such as /home/user/redmine, then you must adjust your links according to your Redmine installation and www root, which may look something like this (for example):
   
   ```
   ln -s /home/user/redmine/public /home/user/public_html/redmine
   ```

6. Make sure that passenger is enabled in Apache:
   
   ```
   sudo a2enmod passenger
   ```

7. Modify /etc/apache2/sites-available/000-default.conf by typing the following:
   
   ```
   sudo nano /etc/apache2/sites-available/000-default.conf
   ```
8. Add the following content near other Directory sections; if there are none, just make sure it's added before closing the </VirtualHost> line:

```<Directory /var/www/html/redmine>
  RailsBaseURI /redmine
  PassengerResolveSymlinksInDocumentRoot on
</Directory>```

9. Install bundler by typing the following:

```sudo gem update && sudo gem install bundler```

10. Finish the installation and restart Apache:

```sudo touch /usr/share/redmine/Gemfile.lock
sudo chown www-data:www-data /usr/share/redmine/Gemfile.lock
sudo service apache2 restart```

After restarting Apache, your Redmine installation should be ready to access by going to http://your_server/redmine.

How it works…

First, you installed Apache web server and Phusion Passenger as a Ruby application server, which runs as an Apache module. Then you edited Apache's configuration file for Passenger by adding the default user to be www-data. After this, we created a symbolic link from Redmine's public directory to the web server's root directory. Then, we edited the virtual server's config to serve this particular directory with two Redmine-related configs – RailsBaseURI/redmine, which tells Ruby on Rails to access Redmine via the /redmine subfolder, and PassengerResolveSymlinksInDocumentRoot tells Passenger to follow symlink to find a Rails application through the symlink path. Instead of this, you could have written the following:

```PassengerAppRoot /usr/share/redmine/```

Redmine would work the same way.

There's more…

The best way to run Redmine or any other web-exposed software is to run it as a restricted user from its home directory and adjust Apache or any other web server to serve data from this directory. Running Redmine as a subdomain such as http://redmine.yoursite.com, is done by creating an ordinary subdomain, and an Apache virtual host file for this subdomain with the settings that are provided in this recipe.
See also

If you stumble upon an error such as 403 (forbidden), error 500, or if you see Ruby code on the screen or the Passenger error screen, but you have followed the preceding steps exactly, refer to Troubleshooting Apache installations section in Chapter 9, Troubleshooting.

If you need to run Redmine as sub-uri, then take a look at http://www.redmine.org/projects/redmine/wiki/HowTo.Install.Redmine.in.a.sub-URI.

Running Redmine with Nginx and Thin

Nginx (pronounced Engine X) is an HTTP and reverse-proxy server with generic TCP proxying features. It is known for its stability, speed, and security. As such, it's often chosen by many web server and Redmine administrators. Thin is a fast and very simple Ruby web server. Combining these two servers provides additional flexibility and scalability in configuration. However, you are free to test/adjust this recipe and use any other server such as Puma or Unicorn instead of Thin.

Getting ready

Install Redmine from a repository, as explained in the Installing Redmine on Ubuntu recipe. If you are using Windows, then install Redmine by following the Windows installation recipe.

How to do it...

First, install Nginx and Thin by typing the following:

```bash
sudo apt-get install nginx thin
```

As we are using Ubuntu, the Aptitude installer will install Nginx and its required libraries for us, perform basic configuration, and start it. To test that it is installed and working correctly, navigate to your server's IP and you should see Nginx's welcome screen:

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.
Configuring thin

To configure thin, it is better to perform the following commands as root, so we type the following:

```
sudo su
```

Then, we type one big line (explained later):

```
thin config --config /etc/thin1.9.1/redmine.yml --chdir /usr/share/redmine --environment production --socket /var/run/redmine/sockets/thin.sock --daemonize --log /var/log/thin/redmine.log --pid /var/run/thin/redmine.pid --user www-data --group www-data --servers 1 --prefix /redmine
```

Create the required directories, as follows:

```
mkdir /var/run/redmine /var/run/redmine/sockets/ /var/run/thin/ /var/log/thin/
chown www-data:www-data /var/run/redmine/sockets/ /var/run/thin/
```

```
nano /etc/logrotate.d/thin
```

Enter the following content to `nano`:

```
/var/log/thin/*.log {
  daily
  missingok
  rotate 52
  compress
  delaycompress
 notifempty
  create 640 root adm
  sharedscripts
  postrotate
    /etc/init.d/thin restart >/dev/null
  endscript
}
```

We need to make sure that thin will work properly after restart. To ensure this, we edit `/etc/init.d/thin` by typing `nano /etc/init.d/thin`.

We then add the following just before `DAEMON=/usr/bin/thin`:

```
pre-start script
  mkdir -p -m0755 /var/run/redmine
  mkdir -p -m0755 /var/run/redmine/sockets
  mkdir -p -m0755 /var/run/thin
```
Installing and Running Redmine

chown www-data:www-data /var/run/redmine/sockets
chown www-data:www-data /var/run/thin

end script

Configuring Nginx

Add a new server to Nginx by typing the following:

nano /etc/nginx/sites-available/redmine

Add the following content that is updated to fit your server name needs:

upstream redmine_thin_servers {
    server unix:/var/run/redmine/sockets/thin.0.sock;
    # Add additional copies if using multiple Thin servers
    #server unix:/var/run/redmine/sockets/thin.1.sock;
}

server {

    listen  80; ## listen for ipv4
    listen  [::]:80 default ipv6only=on; ## listen for ipv6

    # Set appropriately for virtual hosting and to use server_name_in_redirect
    server_name  localhost;
    server_name_in_redirect off;

    access_log  /var/log/nginx/localhost.access.log;
    error_log  /var/log/nginx/localhost.error.log;

    # Note: Documentation says proxy_set_header should work in location
    # block, but testing did not support this statement so it has
    # been placed here in server block
    include /etc/nginx/proxy_params;
    proxy_redirect off;
    # Note: Must match the prefix used in Thin configuration for Redmine
    # or / if no prefix configured
    location /redmine {
        root   /usr/share/redmine/public;

        error_page 404  404.html;
        error_page 500 502 503 504  500.html;

        # Uncomment below lines if using HTTPS
# Note1: Change $host to SSL CN if multiple host names used
# Note2: Adjust prefix, if different in Thin Redmine config
# rewrite ^/redmine/login(.*) https://$host$request_uri permanent;
# rewrite ^/redmine/my/account(.*) https://$host$request_uri permanent;
# rewrite ^/redmine/my/password(.*) https://$host$request_uri permanent;
# rewrite ^/redmine/admin(.*) https://$host$request_uri permanent;

    try_files $uri/index.html $uri.html $uri @redmine_thin_servers;

[location @redmine_thin_servers {
    proxy_pass http://redmine_thin_servers;
}
]

Enable the Redmine site under Nginx:

    ln -s /etc/nginx/sites-available/redmine /etc/nginx/sites-enabled/redmine

### Configuring Redmine for sub-uri

Using nano, open routes.rb, as follows:

    nano /usr/share/redmine/config/routes.rb

Add the following line to Redmine's config file:

    Redmine::Utils::relative_url_root = "/redmine"

Add this line just above the line that looks like this:

    RedmineApp::Application.routes.draw do

### Testing the installation

Restart both servers and test the installation:

    service thin restart
    service nginx restart

Now, navigate to your server's IP or domain/Redmine, and Redmine's initial screen should await you.
Installing and Running Redmine

How it works...

First, we performed a Redmine installation on an Ubuntu system, as explained in the recipe Installing Redmine on an Ubuntu server. Then, we installed the required servers: Thin and Nginx. Thin is a dedicated Ruby apps server, and Nginx is a dedicated web and reverse-proxy server. This way, we have multilayer architecture, allowing us to have, for example, one Nginx and multiple thin instances. In this case, it is connected through a Unix socket but more advanced versions of TCP upstream can be used to run multiple servers on multiple machines and load-balance this way.

Line

```
thin config --config /etc/thin1.9.1/redmine.yml --chdir /usr/share/redmine --environment production --socket /var/run/redmine/sockets/thin.sock --daemonize --log /var/log/thin/redmine.log --pid /var/run/thin/redmine.pid --user www-data --group www-data --servers 1 --prefix /redmine
```

uses the thin server to create config file.

- **--config** tells us where to put the generated file and under which name
- **--chdir** tells us which dir to use to start the thin server
- **--environment** tells us about the Rails environment
- **--socket** tells us the unix socket file path in /var/run
- **--daemonize** to run server as Unix daemon
- **--log** tells us location where to dump log file
- **--pid** is the unix pid file
- **--user** and **--group** under which Unix user and group to run the server
- **--servers** (how many servers to run)
- **--prefix** tells us under which prefix to run it
- **--servers** (tells how many servers to run) for managing concurrent requests (if you put more than one, then the Nginx configuration needs to follow number of the servers specified in this command)
- **--prefix** can be omitted if you want Redmine not to run as sub-uri (Nginx configuration also needs to be updated if using SSL)

After this, we create the necessary folders, set permissions, and add thin to logrotate. Nginx is already there, so we only add thin.

There's more...

You can use this recipe to set up Nginx and Thin on Windows; the only difference is Windows can’t be daemonized. You need to start thin manually; set up a batch or registry file to do it. Also, the TCP port should be used instead of a Unix port.
Redmine can be forced to use SSL all the time through Nginx configuration. To do so, uncomment the SSL lines in Nginx config and copy and paste the server section, set the listen ports to 443, and add the path to your SSL key and certificate:

```
ssl on;
ssl_certificate  /etc/ssl/certs/ssl-cert-snakeoil.pem;
ssl_certificate_key  /etc/ssl/private/ssl-cert-snakeoil.key;
```

**See also**

A comparison of Ruby web servers can be found at:

http://www.madebymarket.com/blog/dev/ruby-web-benchmark-report.html

### Installing optional requirements

Redmine uses several optional components that may be required in some use-case scenarios. This recipe will download multiple libraries from the Internet. Do not use/download unnecessary software if you don't know what it does or just to have green lights in Redmine if you are not planning to use it. Every piece of software on the Internet-connected server/software is a potential security hole, especially if exposed through a web app such as Redmine.

**How to do it...**

Installing optional components is operating system-dependent. This recipe covers the installation of requirements on Ubuntu.

**ImageMagick and rmagick**

ImageMagick is a software suite to create, edit, compose, or convert bitmap images. It can read and write a variety of formats and be used from within third-party applications, such as Redmine, to perform various tasks on the images uploaded or generated by users.

To install ImageMagick on Ubuntu, simply type the following:

```
sudo apt-get install ImageMagick libmagickwand-dev
```

It will download and install a significant number of libraries that are required by ImageMagick.

If you have already installed Redmine and want to add the rmagick gem after installation, use the following command. It can be used for a fresh install as well:

```
bundle install --with rmagick --without development test postgresql
```
Installing and Running Redmine

Installing SCM binaries
Redmine uses several SCM binaries so that it can track code-issue relation in most of the popular Source Control Management (SCM) scenarios. Don't confuse this with Software Configuration Management.

On Ubuntu, installing all supported SCM binaries is extremely simple. Just open up a terminal and type the following:

```
sudo apt-get install subversion darcs mercurial cvs bzr git
```

After installation, restart your Redmine and navigate to Administration | Settings | Repositories.

You should get a screen that looks similar to this:

<table>
<thead>
<tr>
<th>Enabled SCM</th>
<th>Command</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subversion</td>
<td>svn</td>
<td>1.8.8</td>
</tr>
<tr>
<td>Darcs</td>
<td>darcs</td>
<td>2.8.4</td>
</tr>
<tr>
<td>Mercurial</td>
<td>hg</td>
<td>2.8.2</td>
</tr>
<tr>
<td>CVS</td>
<td>cvs</td>
<td>1.12.13</td>
</tr>
<tr>
<td>Bazaar</td>
<td>bzr</td>
<td>2.7.0</td>
</tr>
<tr>
<td>Git</td>
<td>git</td>
<td>1.9.1</td>
</tr>
</tbody>
</table>

You can configure your SCM commands in config/configuration.yml. Please restart the application after editing it.

The green check sign and version number indicates that the repository client binary is installed and usable.

How it works...
Redmine is a complex Ruby on Rails (ROR) application that utilizes a wide variety of Ruby libraries that are required for Redmine to perform various tasks, such as exporting to PDF, connecting to a repository or manipulating images. In this recipe, we covered the installation of ImageMagick and its development libraries, which are optional but required. For example, SCM binaries are used and can be configured on a per-project basis, so you can track multiple repositories on different servers and on different SCM platforms.
There's more...

Subversion along with Git is really popular and widely used by many developers and teams. Redmine can create repositories for you automatically. Additional recipes and how tos can be found at the bottom of Redmine wiki HowTos:


Using custom Ruby for Redmine

While Windows is without a Ruby interpreter and it needs to be manually installed, Linux distributions usually have Ruby in a base package or a web server package. This recipe teaches the user how to install and run Redmine with a custom Ruby interpreter, which can be used if Redmine needs a different Ruby version or if you can't install Ruby system-wide.

Getting ready

First, you need to install Ruby Version Manager (RVM). To use it, you need to have some basic build tools installed on your Linux box. If it's a shared hosting server, most likely these tools are already installed; if they are not, then ask root administrators to install basic build tools or look for another server where you can get sudo or root access.

Preparing for Ubuntu servers

Log in to your server and run the following command:

```
curl -SSL https://rvm.io/mpapis.asc | gpg --import -
```

Open your .gemrc file and add the following:

```
gem: --no-rdoc --no-ri
```

You can do this with nano:

```
nano ~/.gemrc
```

If curl is not installed or available for some reason, you can download a curl binary from http://curl.haxx.se/download.html and execute the same command just by adding ./ in front of it.

If the Ubuntu server is freshly installed, you are probably going to need some basic build tools such as make. To install them, type: `sudo apt-get install build-essential`.

```
Preparation for Cent OS servers
For CentOS, you don't need to import repository keys, but you probably need to install development tools that are required to compile some gems. To install prerequisites, run the following command:

```
yum groupinstall -y development
```

RVM installation
Once we have performed the necessary preparation steps, we start by downloading and installing RVM with this command:

```
curl -L get.rvm.io | bash -s stable
```

This will install the latest stable version of RVM; to start using it, type:

```
source ~/.rvm/scripts/rvm
```

After this command you can check which version of RVM is installed by typing the following:

```
rvm -v
```

How to do it...
Once we have installed RVM, we use it to install the required Ruby version.

First, we ensure RVM is working properly after installation by typing the following:

```
rvm reload
```

Then, we install a custom Ruby version; at the time of writing this book it is 2.2.1, which is supported by Redmine 3.X versions:

```
rvm install ruby-head
```

Then, we proceed with the Redmine installation, as explained in the recipe, Installing from a source on Ubuntu.

After this, running Redmine with Passenger, you need to add the following line to your virtual server's configuration:

```
PassengerRuby /home/your_user/.rvm/gems/ruby-2.1.0/wrappers/ruby
```

Replace your_user and the Ruby version with your actual username; text after PassengerRuby is actually an output of the following command:

```
which ruby
```
After this, reload Apache. If you are on a shared hosting, Apache probably reloads automatically once your server's virtual configuration is edited; if you have root access, you need to reload it manually.

Redmine should now work, if it's not working or throwing some errors, try looking at Chapter 9, Troubleshooting.

**How it works...**

First, we installed **Ruby Version Manager (RVM)** and its required prerequisites, which is usually a set of basic build tools that are required to compile some binary packages on Linux systems. Then, we used RVM to install a custom version of Ruby. RVM lets users have multiple Ruby versions on the system and switch to the default Ruby among them. However, most users are probably going to need only one custom Ruby version available in their home directory, which is going to be used by Redmine. Once Ruby was installed, we proceeded with the Redmine installation from the source by following the recipe *Installing from a source on Ubuntu*. Then, we configured Phusion Passenger and Apache by following the recipe *Running Redmine with Phusion Passenger* but with one variation: we needed to tell Phusion Passenger which Ruby version to use to run Redmine installed in our custom home directory.

**There's more...**

You can experiment with different RVM and Ruby versions and have multiple Redmines installed on the same server by following this recipe. This can also be used to upgrade Redmine without ruining your server if you need the system Ruby version to be used by some other applications. In this case, create a new user, install Redmine under this user's home directory, and migrate data from your old installation. Or just copy the whole directory and perform an upgrade also by upgrading the old database.

**See also**

If you need to use a different version of RVM for some reason, follow the tutorials available on RVM's website: https://rvm.io/rvm/install.
Where to buy this book
You can buy Redmine Cookbook from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.