Installing Redmine

This is the installation documentation for Redmine 1.4.0 and higher. You can still read the document for 1.3.x here.

Requirements

Operating system

Redmine should run on most Unix, Linux, Mac, Mac Server and Windows systems as long as Ruby is available on this platform. See specific installation HowTos here.

Ruby interpreter

The required Ruby versions for a given Redmine version is:

Redmine version	Supported Ruby versions	Rails version used
current trunk	ruby 1.9.3[3], 2.0.0[2], 2.1, 2.2[1]	Rails 4.2[0]
3.0	ruby 1.9.3[3], 2.0.0[2], 2.1, 2.2[1]	Rails 4.2[0]
2.6	ruby 1.8.7[4], 1.9.2, 1.9.3[3], 2.0.0[2], 2.1, 2.2, jruby-1.7.6	Rails 3.2

Redmine 2.6.5 does not support Ruby 2.2. Redmine 2.6.6 supports it (~~#19652~~).

Redmine 3.0 does not support JRuby because some gems do not support Rails 4.2.

- activerecord-jdbc-adapter
- loofah

[0] Rails 4.2.2 (Redmine 3.0.3) had non ASCII URL issue on MinGW Ruby (Windows-based installer) thin and puma (~~#19321~~, ~~#19374~~). Rails 4.2.3 (Redmine 3.0.4) fixed it.
[1] MinGW Ruby 2.2 had Nokogiri issue (~~#19419~~) and it was fixed by Nokogiri 1.6.7 (2015-11-30).
[2] At time of writing (3/19/2013), SQL Server support is reported broken with **ruby 2.0.0 under Windows** because of a database adapter gem incompatibility
[3] MRI 1.9.3p327 contains a bug breaking plugin loading under Windows which 1.9.3p194 or 1.9.3p392 haven't.
[4] Ruby MRI 1.8.7 support has reached its EOL and its use is discouraged. See Important: Ruby 1.8.7 out of support and ~~#14371~~ for additional information.

Supported database back-ends

- MySQL 5.0 or higher
  - make sure to install the C bindings for Ruby that dramatically improve performance. You can get them by running `gem install mysql2`.
  - Redmine 2.x is not compatible with mysql 5.7.3 (#17460). Il will be supported by Redmine 3.
- PostgreSQL 8.2 or higher
  - make sure your database datestyle is set to ISO (Postgresql default setting). You can set it using: `ALTER DATABASE "redmine_db" SET datestyle="ISO,MDY";`
  - some bugs in PostgreSQL 8.4.0 and 8.4.1 affect Redmine behavior (~~#4259~~, ~~#4314~~), they are fixed in PostgreSQL 8.4.2
- Microsoft SQL Server
  - Redmine 2.x: 2008 or higher (since Redmine 2.3.0)
  - Redmine 3.x: 2012 or higher
- SQLite 3 (not for multi-user production use!)

Optional components

- SCM binaries (eg. `svn`), for repository browsing (must be available in your PATH). See RedmineRepositories for SCM compatibility and requirements.
- ImageMagick (to enable Gantt export to PNG image and thumbnails generation).
- Ruby OpenID Library (to enable OpenID support). Version 2 or greater is required.

Redmine Version

It is recommended that the majority of users install the proper point releases of redmine. Redmine currently releases a new version every 6 months, and these releases are considered very usable and stable. It is **not** recommended to install redmine from trunk, unless you are deeply familiar with Ruby on Rails and keep up with the changes - Trunk *does* break from time-to-time.

Installation procedure

Step 1 - Redmine application

Get the Redmine source code by either downloading a packaged release or checking out the code repository.

See the download page for details.

Step 2 - Create an empty database and accompanying user

Redmine database user will be named `redmine` hereafter but it can be changed to anything else.

MySQL

```
CREATE DATABASE redmine CHARACTER SET utf8;
CREATE USER 'redmine'@'localhost' IDENTIFIED BY 'my_password';
GRANT ALL PRIVILEGES ON redmine.* TO 'redmine'@'localhost';
```

For versions of MySQL prior to 5.0.2 - skip the 'create user' step and instead:

```
GRANT ALL PRIVILEGES ON redmine.* TO 'redmine'@'localhost' IDENTIFIED BY 'my_password';
```

PostgreSQL

```
CREATE ROLE redmine LOGIN ENCRYPTED PASSWORD 'my_password' NOINHERIT VALID UNTIL 'infinity';
CREATE DATABASE redmine WITH ENCODING='UTF8' OWNER=redmine;
```

SQL Server

The database, login and user can be created within *SQL Server Management Studio* with a few clicks.

If you prefer the command line option with `SQLCMD`, here's some basic example:

Show SQL

Step 3 - Database connection configuration

Copy `config/database.yml.example` to `config/database.yml` and edit this file in order to configure your database settings for "production" environment.

Example for a MySQL database using ruby 1.8 or jruby:

```
production:
  adapter: mysql
  database: redmine
  host: localhost
  username: redmine
  password: my_password
```

Example for a MySQL database using ruby 1.9 (adapter must be set to `mysql2`):

```
production:
  adapter: mysql2
  database: redmine
  host: localhost
  username: redmine
  password: my_password
```

If your server is not running on the standard port (3306), use this configuration instead:

```
production:
  adapter: mysql
  database: redmine
  host: localhost
  port: 3307
  username: redmine
  password: my_password
```

Example for a PostgreSQL database (default port):

```
production:
  adapter: postgresql
  database: <your_database_name>
  host: <postgres_host>
  username: <postgres_user>
  password: <postgres_user_password>
  encoding: utf8
  schema_search_path: <database_schema> (default - public)
```

Example for a SQL Server database (default host `localhost`, default port `1433`):

```
production:
  adapter: sqlserver
  database: redmine
  username: redmine # should match the database user name
  password: redminepassword # should match the login password
```

Step 4 - Dependencies installation

Redmine uses Bundler to manage gems dependencies.

You need to install Bundler first:

```
gem install bundler
```

Then you can install all the gems required by Redmine using the following command:

```
bundle install --without development test
```

Optional dependencies

RMagick (allows the use of ImageMagick to manipulate images for PDF and PNG export)

If ImageMagick is not installed on your system, you should skip the installation of the rmagick gem using:

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

If you have trouble installing `rmagick` on Windows, refer to this HowTo.

Database adapters

Redmine automatically installs the adapter gems required by your database configuration by reading it from the `config/database.yml` file (eg. if you configured only a connection using the `mysql2` adapter, then only the `mysql2` gem will be installed).

Don't forget to re-run `bundle install --without development test ...` after adding or removing adapters in the `config/database.yml` file!

Additional dependencies (`Gemfile.local`)

If you need to load gems that are not required by Redmine core (eg. Puma, fcgi), create a file named `Gemfile.local` at the root of your redmine directory. It will be loaded automatically when running `bundle install`.

Example:

```
# Gemfile.local
gem 'puma'
```

Step 5 - Session store secret generation

This step generates a random key used by Rails to encode cookies storing session data thus preventing their tampering.
Generating a new secret token invalidates all existing sessions after restart.

- with Redmine 1.4.x:

```
bundle exec rake generate_session_store
```

- with Redmine 2.x:

```
bundle exec rake generate_secret_token
```

Alternatively, you can store this secret in config/secrets.yml:
http://guides.rubyonrails.org/upgrading_ruby_on_rails.html#config-secrets-yml

Step 6 - Database schema objects creation

Create the database structure, by running the following command under the application root directory:

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

Windows syntax:

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

It will create tables by running all migrations one by one then create the set of the permissions and the application administrator account, named `admin`.

Ubuntu troubleshooting:

If you get this error with Ubuntu:

```
Rake aborted!
no such file to load -- net/https
```

Then you need to install `libopenssl-ruby1.8` just like this: `apt-get install libopenssl-ruby1.8`.

Step 7 - Database default data set

Insert default configuration data in database, by running the following command:

```
RAILS_ENV=production bundle exec rake redmine:load_default_data
```

Redmine will prompt you for the data set language that should be loaded; you can also define the `REDMINE_LANG` environment variable before running the command to a value which will be automatically and silently picked up by the task.

E.g.:

Unices:

```
RAILS_ENV=production REDMINE_LANG=fr bundle exec rake redmine:load_default_data
```

Windows:

```
set RAILS_ENV=production
set REDMINE_LANG=fr
bundle exec rake redmine:load_default_data
```

Step 8 - File system permissions

NB: *Windows users can skip this section.*

The user account running the application must have write permission on the following subdirectories:

1. `files` (storage of attachments)
2. `log` (application log file `production.log`)
3. `tmp` and `tmp/pdf` (create these ones if not present, used to generate PDF documents among other things)
4. `public/plugin_assets` (assets of plugins)

E.g., assuming you run the application with a redmine user account:

```
mkdir -p tmp tmp/pdf public/plugin_assets
sudo chown -R redmine:redmine files log tmp public/plugin_assets
sudo chmod -R 755 files log tmp public/plugin_assets
```

Step 9 - Test the installation

Test the installation by running WEBrick web server:

- with Redmine 1.4.x:

```
bundle exec ruby script/server webrick -e production
```

- with Redmine 2.x:

```
bundle exec ruby script/rails server webrick -e production
```

- with Redmine 3.x:

```
bundle exec rails server webrick -e production
```

Once WEBrick has started, point your browser to http://localhost:3000/. You should now see the application welcome page.

> *Note: Webrick is **not** suitable for production use, please only use webrick for testing that the installation up to this point is functional. Use one of the many other guides in this wiki to setup redmine to use either Passenger (aka `mod_rails`), FCGI or a Rack server (Unicorn, Thin, Puma, hellip;) to serve up your redmine.*

Step 10 - Logging into the application

Use default administrator account to log in:

- login: admin
- password: admin

You can go to *Administration* menu and choose *Settings* to modify most of the application settings.

Configuration

Redmine settings are defined in a file named `config/configuration.yml`.

If you need to override default application settings, simply copy `config/configuration.yml.example` to `config/configuration.yml` and edit the new file; the file is well commented by itself, so you should have a look at it.

These settings may be defined per Rails environment (`production`/`development`/`test`).

Important : don't forget to restart the application after any change.

Email / SMTP server settings

Email configuration is described in a dedicated page.

SCM settings

This configuration section allows you to:

- override default commands names if the SCM binaries present in the `PATH` variable doesn't use the standard name (Windows .bat/.cmd names won't work)
- specify the full path to the binary

Examples (with Subversion):

Command name override:

```
scm_subversion_command: "svn_replacement.exe"
```

Absolute path:

```
scm_subversion_command: "C:\Program Files\Subversion\bin\svn.exe"
```

Attachment storage settings

You can set a path where Redmine attachments will be stored which is different from the default 'files' directory of your Redmine instance using the `attachments_storage_path` setting.

Examples:

```
attachments_storage_path: /var/redmine/files
```

```
attachments_storage_path: D:/redmine/files
```

Logging configuration

Redmine defaults to a log level of :info, writing to the `log` subdirectory. Depending on site usage, this can be a lot of data so to avoid the contents of the logfile growing without bound, consider rotating them, either through a system utility like `logrotate` or via the `config/additional_environment.rb` file.

To use the latter, copy `config/additional_environment.rb.example` to `config/additional_environment.rb` and add the following lines. Note that the new logger defaults to a high log level and hence has to be explicitly set to `info`.

```
#Logger.new(PATH,NUM_FILES_TO_ROTATE,FILE_SIZE)
config.logger = Logger.new('/path/to/logfile.log', 2, 1000000)
config.logger.level = Logger::INFO
```

Backups

Redmine backups should include:

- data (stored in your redmine database)
- attachments (stored in the `files` directory of your Redmine install)

Here is a simple shell script that can be used for daily backups (assuming you're using a mysql database):

```
# Database
/usr/bin/mysqldump -u <username> -p<password> <redmine_database> | gzip > /path/to/backup/db/redmine_`date +%y_%m_%d`.gz

# Attachments
rsync -a /path/to/redmine/files /path/to/backup/files
```

Notes on Linux/Unix installation

Be sure to disable security hardenning tools during the installation process if you run into bizarre permission problems. These problems are mostly silent and can be caused by tools like extended ACLs, SELinux, or AppArmor. There tools are mostly used in big companies with a strict security policy, default Linux/Unix distributions settings shouldn't be a problem.

Notes on Windows installation

There is an prebuilt installer of Ruby MRI available from http://rubyinstaller.org.
After installing it, select *Start Command Prompt with Ruby* in the start menu.

Specifying the `RAILS_ENV` environment variable:

When running command as described in this guide, you have to set the `RAILS_ENV` environment variable using a separate command.

I.e. commands with the following syntaxes:

```
RAILS_ENV=production <any commmand>
```

```
<any commmand> RAILS_ENV=production
```

have to be turned into 2 subsequent commands:

```
set RAILS_ENV=production
<any commmand>
```

MySQL gem installation issue:

You may need to manually install the mysql gem using the following command:

```
gem install mysql
```

And in some case it is required to copy the *libmysql.dll* file in your ruby/bin directory.
Not all libmysql.dll are ok this seem to works http://instantrails.rubyforge.org/svn/trunk/InstantRails-win/InstantRails/mysql/bin/libmySQL.dll.

Important note for Win7 and later
On Win7 and later, `localhost` is commented out in the hosts file[1] and IPV6 is the default[2]. As the mysql2 gem does no support IPV6 addresses[3], a connection can't be established and you get the error "`Can't connect to MySQL server on 'localhost' (10061)`".
You can confirm this by pinging `localhost`, if ping targets "::1:" IPV6 is being used.

Workaround:
Replace `localhost` with `127.0.0.1` in database.yml.

1 http://serverfault.com/questions/4689/windows-7-localhost-name-resolution-is-handled-within-dns-itself-why
2 http://www.victor-ratajczyk.com/post/2012/02/25/mysql-fails-to-resolve-localhost-disable-ipv6-on-windows.aspx
3 https://github.com/brianmario/mysql2/issues/279

Alternative to manual installation

Some users may prefer to skip manual installation by using one of the third-party Redmine bundles on the download page.