Instalação Redmine Docker Sameersbn

Link: https://github.com/sameersbn/docker-redmine

git clone https://github.com/sameersbn/docker-redmine.git
Agosto/2025

Table of Contents

- Table of Contents
- Introduction
  - Version
- Contributing
- Issues
- Installation
- Quick Start
- Configuration
  - Data Store
  - Database
    - MySQL
      - Internal MySQL Server
      - External MySQL Server
      - Linking to MySQL Container
    - PostgreSQL
      - External PostgreSQL Server
      - Linking to PostgreSQL Container
    - AWS RDS Integration
  - Memcached (Optional)
    - External Memcached Server

        - Linking to Memcached Container
        - Mail
        - SSL
            - Generation of Self Signed Certificates
            - Strengthening the server security
            - Installation of the SSL Certificates
            - Enabling HTTPS support
            - Configuring HSTS
            - Using HTTPS with a load balancer
        - Deploy to a subdirectory (relative url root)
        - Apache Proxy as frontend
        - Mapping host user and group
        - Available Configuration Parameters
- Plugins
    - Installing Plugins
        - Reloading plugins for development
    - Uninstalling Plugins
- Themes
    - Installing Themes
        - Reloading themes for development
    - Uninstalling Themes
- Maintenance
    - Creating backups
    - Restoring Backups
    - Automated backups
    - Rake Tasks
    - Upgrading
    - Shell Access
- Development
    - Upgrading to next redmine release
- References

Introduction

Dockerfile to build a Redmine container image.

Version

Current Version: **sameersbn/redmine:6.0.6**

P.S.: If your installation depends on various third party plugins, please stick with 2.6.xx series to avoid breakage.

Contributing

If you find this image useful here's how you can help:

- Send a Pull Request with your awesome new features and bug fixes
- Help new users with Issues they may encounter
- Support the development of this image with a donation

Issues

Docker is a relatively new project and is active being developed and tested by a thriving community of developers and testers and every release of docker features many enhancements and bugfixes.

Given the nature of the development and release cycle it is very important that you have the latest version of docker installed because any issue that you encounter might have already been fixed with a newer docker release.

Install the most recent version of the Docker Engine for your platform using the official Docker releases, which can also be installed using:

```
wget -qO- https://get.docker.com/ | sh
```

Fedora and RHEL/CentOS users should try disabling selinux with `setenforce 0` and check if resolves the issue. If it does than there is not much that I can help you with. You can either stick with selinux disabled (not recommended by redhat) or switch to using ubuntu.

If using the latest docker version and/or disabling selinux does not fix the issue then please file a issue request on the issues page.

In your issue report please make sure you provide the following information:

- The host distribution and release version.
- Output of the `docker version` command.
- Output of the `docker info` command.
- The `docker run` command you used to run the image (mask out the sensitive bits).

Installation

Automated builds of the image are available on Dockerhub and is the recommended method of installation.

> **Note**: Builds are also available on Quay.io

```
docker pull sameersbn/redmine:latest
```

Since version `2.4.2`, the image builds are being tagged. You can now pull a particular version of redmine by specifying the version number. For example,

```
docker pull sameersbn/redmine:6.0.6
```

Alternately you can build the image yourself.

```
docker build -t sameersbn/redmine github.com/sameersbn/docker-redmine
```

Quick Start

The quickest way to get started is using docker compose.

```
wget https://raw.githubusercontent.com/sameersbn/docker-redmine/master/docker-compose.yml
docker compose up
```

Alternately, you can manually launch the `redmine` container and the supporting `postgresql` container by following this two step guide.

Step 1. Launch a postgresql container

```
docker run --name=postgresql-redmine -d \
  --env='DB_NAME=redmine_production' \
  --env='DB_USER=redmine' --env='DB_PASS=password' \
  --volume=/srv/docker/redmine/postgresql:/var/lib/postgresql \
  sameersbn/postgresql:14-20230628
```

Step 2. Launch the redmine container

```
docker run --name=redmine -d \
  --link=postgresql-redmine:postgresql --publish=10083:80 \
  --env='REDMINE_PORT=10083' \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

NOTE: Please allow a minute or two for the Redmine application to start.

Point your browser to `http://localhost:10083` and login using the default username and password:

- username: **admin**
- password: **admin**

Make sure you visit the `Administration` link and `Load the default configuration` before creating any projects.

You now have the Redmine application up and ready for testing. If you want to use this image in production the please read on.

The rest of the document will use the docker command line. You can quite simply adapt your configuration into a `docker-compose.yml` file if you wish to do so.

Configuration

Data Store

For the file storage we need to mount a volume at the following location.

- `/home/redmine/data`
- `/var/log/redmine` for server logs

> **NOTE**
>
> Existing users **need to move** the existing files directory inside `/srv/docker/redmine/redmine/`.
>
> ```
> mkdir -p /srv/docker/redmine/redmine
> mv /opt/redmine/files /srv/docker/redmine/redmine
> ```

SELinux users are also required to change the security context of the mount point so that it plays nicely with selinux.

```
mkdir -p /srv/docker/redmine/redmine
sudo chcon -Rt svirt_sandbox_file_t /srv/docker/redmine/redmine
```

Volumes can be mounted in docker by specifying the **'-v'** option in the docker run command.

```
docker run --name=redmine -it --rm \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

Database

Redmine uses a database backend to store its data.

MySQL

Internal MySQL Server

The internal mysql server has been removed from the image. Please use a linked mysql or postgresql container instead or connect with an external mysql or postgresql server.

Refer to Linking to MySQL Container for more information.

External MySQL Server

The image can be configured to use an external MySQL database instead of starting a MySQL server internally. The database configuration should be specified using environment variables while starting the Redmine image.

Before you start the Redmine image create user and database for redmine.

```
mysql -uroot -p
CREATE USER 'redmine'@'%.%.%.%' IDENTIFIED BY 'password';
CREATE DATABASE IF NOT EXISTS `redmine_production` DEFAULT CHARACTER SET `utf8` COLLATE `utf8_ur
GRANT SELECT, LOCK TABLES, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER ON `redmine_produc
```

We are now ready to start the redmine application.

```
docker run --name=redmine -it --rm \
  --env='DB_ADAPTER=mysql2' \
  --env='DB_HOST=192.168.1.100' --env='DB_NAME=redmine_production' \
  --env='DB_USER=redmine' --env='DB_PASS=password' \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

This will initialize the redmine database and after a couple of minutes your redmine instance should be ready to use.

Linking to MySQL Container

See example docker-compose-mysql.yml.

PostgreSQL

External PostgreSQL Server

The image also supports using an external PostgreSQL Server. This is also controlled via environment variables.

```
CREATE ROLE redmine with LOGIN CREATEDB PASSWORD 'password';
CREATE DATABASE redmine_production;
GRANT ALL PRIVILEGES ON DATABASE redmine_production to redmine;
```

We are now ready to start the redmine application.

```
docker run --name=redmine -it --rm \
  --env='DB_ADAPTER=postgresql' \
  --env='DB_HOST=192.168.1.100' --env='DB_NAME=redmine_production' \
  --env='DB_USER=redmine' --env='DB_PASS=password' \
```

```
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

This will initialize the redmine database and after a couple of minutes your redmine instance should be ready to use.

Linking to PostgreSQL Container

You can link this image with a postgresql container for the database requirements. The alias of the postgresql server container should be set to **postgresql** while linking with the redmine image.

If a postgresql container is linked, only the `DB_ADAPTER`, `DB_HOST` and `DB_PORT` settings are automatically retrieved using the linkage. You may still need to set other database connection parameters such as the `DB_NAME`, `DB_USER`, `DB_PASS` and so on.

To illustrate linking with a postgresql container, we will use the sameersbn/postgresql image. When using postgresql image in production you should mount a volume for the postgresql data store.

Please refer the README of docker-postgresql for details.

First, lets pull the postgresql image from the docker index.

```
docker pull sameersbn/postgresql:14-20230628
```

For data persistence lets create a store for the postgresql and start the container.

SELinux users are also required to change the security context of the mount point so that it plays nicely with selinux.

```
mkdir -p /srv/docker/redmine/postgresql
sudo chcon -Rt svirt_sandbox_file_t /srv/docker/redmine/postgresql
```

The run command looks like this.

```
docker run --name=postgresql-redmine -d \
  --env='DB_NAME=redmine_production' \
  --env='DB_USER=redmine' --env='DB_PASS=password' \
  --volume=/srv/docker/redmine/postgresql:/var/lib/postgresql \
  sameersbn/postgresql:14-20230628
```

The above command will create a database named `redmine_production` and also create a user named `redmine` with the password `password` with access to the `redmine_production` database.

We are now ready to start the redmine application.

```
docker run --name=redmine -it --rm --link=postgresql-redmine:postgresql \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

Here the image will also automatically fetch the `DB_NAME`, `DB_USER` and `DB_PASS` variables from the postgresql container as they are specified in the `docker run` command for the postgresql container. This is made possible using the magic of docker links and works with the following images:

- postgres
- sameersbn/postgresql
- orchardup/postgresql
- paintedfox/postgresql

AWS RDS Integration

docker-redmine has support for fetching secrets from AWS Secrets Manager at runtime. Read docs/aws.md for detailed instructions.

Memcached (Optional)

This image can (optionally) be configured to use a memcached server to speed up Redmine. This is particularly useful when you have a large number users.

External Memcached Server

The image can be configured to use an external memcached server. The memcached server host and port configuration should be specified using environment variables `MEMCACHE_HOST` and `MEMCACHE_PORT` like so:

Assuming that the memcached server host is 192.168.1.100

```
docker run --name=redmine -it --rm \
  --env='MEMCACHE_HOST=192.168.1.100' --env='MEMCACHE_PORT=11211' \
  sameersbn/redmine:6.0.6
```

Linking to Memcached Container

Alternately you can link this image with a memcached container. The alias of the memcached server container should be set to **memcached** while linking with the redmine image.

To illustrate linking with a memcached container, we will use the sameersbn/memcached image.

Please refer the README of docker-memcached for details.

First, lets pull and launch the memcached image from the docker index.

```
docker run --name=memcached-redmine -d sameersbn/memcached:1.5.6
```

Now you can link memcached to the redmine image:

```
docker run --name=redmine -it --rm --link=memcached-redmine:memcached \
  sameersbn/redmine:6.0.6
```

Mail

The mail configuration should be specified using environment variables while starting the redmine image. The configuration defaults to using gmail to send emails and requires the specification of a valid username and password to login to the gmail servers.

Please refer the Available Configuration Parameters section for the list of SMTP parameters that can be specified.

```
docker run --name=redmine -it --rm \
  --env='SMTP_USER=USER@gmail.com' --env='SMTP_PASS=PASSWORD' \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

If you are not using google mail, then please configure the SMTP host and port using the `SMTP_HOST` and `SMTP_PORT` configuration parameters.

If you are using a google apps account with a custom domain (other than google.com), you need to set the `SMTP_DOMAIN` parameters or else you will get internal server error when doing an action that would normally send a mail.

Please see redmine email config for examples of different email configurations:

https://www.redmine.org/projects/redmine/wiki/emailconfiguration

Similary you can configure receiving emails using the `IMAP_` configuration options. Please refer Available Configuration Parameters for details. When receiving emails is enabled users can comment on issues by replying to emails.

P.S. The receiving emails feature is only available since versions `2.6.6-2`, `3.0.4-2` and `3.1.0-2`. Refer the Changelog for details.

SSL

Access to the redmine application can be secured using SSL so as to prevent unauthorized access. While a CA certified SSL certificate allows for verification of trust via the CA, a self signed certificates can also provide an equal level of trust verification as long as each client takes some additional steps to verify the identity of your website. I will provide instructions on achieving this towards the end of this section.

To secure your application via SSL you basically need two things:

- **Private key (.key)**
- **SSL certificate (.crt)**

When using CA certified certificates, these files are provided to you by the CA. When using self-signed certificates you need to generate these files yourself. Skip the following section if you are armed with CA certified SSL certificates.

Jump to the Using HTTPS with a load balancer section if you are using a load balancer such as hipache, haproxy or nginx.

Generation of Self Signed Certificates

Generation of self-signed SSL certificates involves a simple 3 step procedure.

STEP 1: Create the server private key

```
openssl genrsa -out redmine.key 2048
```

STEP 2: Create the certificate signing request (CSR)

```
openssl req -new -key redmine.key -out redmine.csr
```

STEP 3: Sign the certificate using the private key and CSR

```
openssl x509 -req -days 365 -in redmine.csr -signkey redmine.key -out redmine.crt
```

Congratulations! you have now generated an SSL certificate thats valid for 365 days.

Strengthening the server security

This section provides you with instructions to strengthen your server security. To achieve this we need to generate stronger DHE parameters.

```
openssl dhparam -out dhparam.pem 2048
```

Installation of the SSL Certificates

Out of the four files generated above, we need to install the `redmine.key`, `redmine.crt` and `dhparam.pem` files at the redmine server. The CSR file is not needed, but do make sure you safely backup the file (in case you ever need it again).

The default path that the redmine application is configured to look for the SSL certificates is at `/home/redmine/data/certs`, this can however be changed using the `SSL_KEY_PATH`, `SSL_CERTIFICATE_PATH` and `SSL_DHPARAM_PATH` configuration options.

If you remember from above, the `/home/redmine/data` path is the path of the data store, which means that we have to create a folder named certs inside `/srv/docker/redmine/redmine/` and copy the files into it and as a measure of security we will update the permission on the `redmine.key` file to only be readable by the owner.

```
mkdir -p /srv/docker/redmine/redmine/certs
cp redmine.key /srv/docker/redmine/redmine/certs/
cp redmine.crt /srv/docker/redmine/redmine/certs/
cp dhparam.pem /srv/docker/redmine/redmine/certs/
chmod 400 /srv/docker/redmine/redmine/certs/redmine.key
```

Great! we are now just one step away from having our application secured.

Enabling HTTPS support

HTTPS support can be enabled by setting the `REDMINE_HTTPS` option to `true`.

```
docker run --name=redmine -d \
  --publish=10083:80 --publish 10445:443 \
  --env='REDMINE_PORT=10445' --env='REDMINE_HTTPS=true' \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

In this configuration, any requests made over the plain http protocol will automatically be redirected to use the https protocol. However, this is not optimal when using a load balancer.

Note: If startup prints `SSL keys and certificates were not found.` refer to SSL and verify you put the certs in the correct place. Unless your trying to setup for Using HTTPS with a load balancer

Configuring HSTS

HSTS if supported by the browsers makes sure that your users will only reach your server via HTTPS. When the user comes for the first time it sees a header from the server which states for how long from now this site should only be reachable via HTTPS - that's the HSTS max-age value.

With `NGINX_HSTS_MAXAGE` you can configure that value. The default value is `31536000` seconds. If you want to disable a already sent HSTS MAXAGE value, set it to `0`.

```
docker run --name=redmine -d \
  --env='REDMINE_HTTPS=true' \
  --env='NGINX_HSTS_MAXAGE=2592000'
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

If you want to completely disable HSTS set `NGINX_HSTS_ENABLED` to `false`.

Using HTTPS with a load balancer

Load balancers like nginx/haproxy/hipache talk to backend applications over plain http and as such the installation of ssl keys and certificates are not required and should **NOT** be installed in the container. The SSL configuration has to instead be done at the load balancer. Hoewever, when using a load balancer you **MUST** set `REDMINE_HTTPS` to `true`.

With this in place, you should configure the load balancer to support handling of https requests.

But that is out of the scope of this document. Please refer to Using SSL/HTTPS with HAProxy for information on the subject.

When using a load balancer, you probably want to make sure the load balancer performs the automatic http to https redirection. Information on this can also be found in the link above.

In summation, when using a load balancer, the docker command would look for the most part something like this:

```
docker run --name=redmine -d --publish=10083:80 \
  --env='REDMINE_HTTPS=true' \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

Deploy to a subdirectory (relative url root)

By default redmine expects that your application is running at the root (eg. /). This section explains how to run your application inside a directory.

Let's assume we want to deploy our application to '/redmine'. Redmine needs to know this directory to generate the appropriate routes. This can be specified using the `REDMINE_RELATIVE_URL_ROOT` configuration option like so:

```
docker run --name=redmine -d --publish=10083:80 \
  --env='REDMINE_RELATIVE_URL_ROOT=/redmine' \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6
```

Redmine will now be accessible at the `/redmine` path, e.g. `http://www.example.com/redmine`.

Note: *The `REDMINE_RELATIVE_URL_ROOT` parameter should always begin with a slash and **SHOULD NOT** have any trailing slashes.*

Apache Proxy as frontend

Ref #370

Apache config

```
# REDMINE Pass connections to docker
ProxyRequests Off
ProxyPass /redmine http://127.0.0.1:10083/redmine/
ProxyPassReverse /redmine http://127.0.0.1:10083/redmine/
```

Note the following should be set: `REDMINE_RELATIVE_URL_ROOT=/redmine` and port mapped `--publish=10083:80`

Mapping host user and group

Per default the container is configured to run redmine as user and group `redmine` with `uid` and `gid` `1000`. The host possibly uses this ids for different purposes leading to unfavorable effects. From the host it appears as if the mounted data volumes are owned by the host's user/group `1000`.

Also the container processes seem to be executed as the host's user/group `1000`. The container can be configured to map the `uid` and `gid` of `redmine` user to different ids on host by passing the environment variables `USERMAP_UID` and `USERMAP_GID`. The following command maps the ids to user and group `redmine` on the host.

```
docker run --name=redmine -it --rm [options] \
  --env="USERMAP_UID=500" --env="USERMAP_GID=500" \
  sameersbn/redmine:6.0.6
```

Available Configuration Parameters

Please refer the docker run command options for the `--env-file` flag where you can specify all required environment variables in a single file. This will save you from writing a potentially long docker run command.

Below is the complete list of parameters that can be set using environment variables.

- **REDMINE_HTTPS**: Enable HTTPS (SSL/TLS) port on server. Defaults to `false`
- **REDMINE_PORT**: The port of the Redmine server. Defaults to `80` for plain http and `443` when https is enabled.
- **REDMINE_RELATIVE_URL_ROOT**: The relative url of the Redmine server, e.g. `/redmine`. No default.
- **REDMINE_ATTACHMENTS_DIR**: The attachments directory. Defaults to `/home/redmine/data/files`
- **REDMINE_SECRET_TOKEN**: Secret key for verifying cookie session data integrity. Defaults to a random alphanumeric string.
- **REDMINE_MINIMAGICK_FONT_PATH**: The minimagick_font_path for the png export function of GANTT to work. Defaults to `/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf`.
- **REDMINE_CONCURRENT_UPLOADS**: Maximum number of simultaneous AJAX uploads. Defaults to `2`.
- **REDMINE_SUDO_MODE_ENABLED**: Requires users to re-enter their password for sensitive actions. Defaults to `false`.
- **REDMINE_SUDO_MODE_TIMEOUT**: Sudo mode timeout. Defaults to `15` minutes.
- **REDMINE_FETCH_COMMITS**: Setup cron job to fetch commits. Possible values `disable`, `hourly`, `daily` or `monthly`. Disabled by default.
- **REDMINE_AUTOLOGIN_COOKIE_NAME**: The name of autologin-cookie. Defaults to `autologin`.
- **REDMINE_AUTOLOGIN_COOKIE_PATH**: The path of autologin-cookie. Defaults to `/`.
- **REDMINE_AUTOLOGIN_COOKIE_SECURE**: Set autologin-cookie to secure. Defaults to `true` when `REDMINE_HTTPS` is `true`, else defaults to `false`.
- **REDMINE_BACKUPS_DIR**: The backup folder in the container. Defaults to `/home/redmine/data/backups`
- **REDMINE_BACKUP_SCHEDULE**: Setup cron job to schedule automatic backups. Possible values `disable`, `daily`, `weekly` or `monthly`. Disabled by default
- **REDMINE_BACKUP_EXPIRY**: Configure how long (in seconds) to keep backups before they are deleted. By default when automated backups are disabled backups are kept forever (0 seconds), else the backups expire in 7 days (604800 seconds).
- **REDMINE_BACKUP_TIME**: Set a time for the automatic backups in `HH:MM` format. Defaults to `04:00`.

- **REDMINE_AVATAR_SERVER_URL**: Avatar server for displaying user icons. Defaults to `https://www.gravatar.com`
- **DATABASE_URL**: The database URL. See Configuring a Database. Possible schemes: `postgres`, `postgresql`, `mysql2`, and `sqlite3`. Defaults to no URL.
- **DB_ADAPTER**: The database type. Possible values: `mysql2`, `postgresql`, and 'sqlite3'. Defaults to `mysql`.
- **DB_CREATE**: Whether the db should be automatically created (`bundle exec rake db:create`). Defaults to `true`.
- **DB_ENCODING**: The database encoding. For `DB_ADAPTER` values `postresql` and `mysql2`, this parameter defaults to `unicode` and `utf8` respectively. For full unicode support (all emojis) with mariadb or mysql set this to `utf8mb4` and make sure to also set all tables to `utf8mb4` and use `collate utf8mb4_unicode_ci`. Existing databases can be converted by following this HowTo.
- **DB_HOST**: The database server hostname. Defaults to `localhost`.
- **DB_PORT**: The database server port. Defaults to `3306`.
- **DB_NAME**: The database name. Defaults to `redmine_production`
- **DB_USER**: The database user. Defaults to `root`
- **DB_PASS**: The database password. Defaults to no password
- **DB_POOL**: The database connection pool count. Defaults to `5`.
- **DB_SKIP_CHECK**: Skip waiting for the database to start. Defaults to `false`.
- **DB_SSL_MODE**: Configures the database ssl mode. Valid options for postgresql (disable|allow|prefer|require|verify-ca|verify-full) and mysql (disable||preferred|required|verify_ca|verify_identity). Defaults to ''
- **LOGGER_LEVEL**: Configures the amount of messages that are generated when redmine is running. Possible values are: `debug`, `info`, `warn`, `error`. Defaults to `info`
- **NGINX_ENABLED**: Enable/disable the nginx server. Disabling Nginx is not recommended (see #148), use at your discretion. Defaults to `true`. When disabled publish port `8080` instead of the usual port `80` or `443`.
- **NGINX_WORKERS**: The number of nginx workers to start. Defaults to `1`.
- **NGINX_MAX_UPLOAD_SIZE**: Maximum acceptable upload size. Defaults to `20m`.
- **NGINX_X_FORWARDED_PROTO**: Advanced configuration option for the `proxy_set_header X-Forwarded-Proto` setting in the redmine nginx vHost configuration. Defaults to `https` when `REDMINE_HTTPS` is `true`, else defaults to `$scheme`.
- **NGINX_HSTS_ENABLED**: Advanced configuration option for turning off the HSTS configuration. Applicable only when SSL is in use. Defaults to `true`. See #138 for use case scenario.
- **NGINX_HSTS_MAXAGE**: Advanced configuration option for setting the HSTS max-age in the redmine nginx vHost configuration. Applicable only when SSL is in use. Defaults to `31536000`.
- **NGINX_CORS_ALLOW_ORIGIN**: Sets `Access-Control-Allow-Origin` response header to indicate that the response can be shared with requesting code from the given origin.
- **NGINX_CORS_ALLOW_METHODS**: Sets `Access-Control-Allow-Methods` response header to specify the methods allowed when accessing the resource in response to a preflight request.

- **NGINX_CORS_ALLOW_HEADERS**: Sets `Access-Control-Allow-Headers` response header to specify which headers can be used during the actual request.
- **NGINX_CORS_ALLOW_CREDENTIALS**: Sets `Access-Control-Allow-Credentials` response header to tell browsers whether to expose the response to frontend JavaScript code when the request's credentials mode (Request.credentials) is include.
- **PUMA_WORKERS**: The number of puma workers to start. Defaults to `2`.
- **MEMCACHE_HOST**: The host name of the memcached server. No defaults.
- **MEMCACHE_PORT**: The connection port of the memcached server. Defaults to `11211`.
- **SSL_CERTIFICATE_PATH**: The path to the SSL certificate to use. Defaults to `/home/redmine/data/certs/redmine.crt`.
- **SSL_KEY_PATH**: The path to the SSL certificate's private key. Defaults to `/home/redmine/data/certs/redmine.key`.
- **SSL_DHPARAM_PATH**: The path to the Diffie-Hellman parameter. Defaults to `/home/redmine/data/certs/dhparam.pem`.
- **SSL_VERIFY_CLIENT**: Enable verification of client certificates using the `SSL_CA_CERTIFICATES_PATH` file. Configures ssl_verify_client in nginx, options (`off`, `on`, `optional`, `optional_no_ca`). Defaults to `off`
- **SSL_CA_CERTIFICATES_PATH**: List of SSL certificates to trust. Defaults to `/home/redmine/data/certs/ca.crt`.
- **SMTP_ENABLED**: Enable mail delivery via SMTP. Defaults to `true` if `SMTP_USER` is defined, else defaults to `false`.
- **SMTP_DOMAIN**: SMTP domain. Defaults to `www.gmail.com`
- **SMTP_HOST**: SMTP server host. Defaults to `smtp.gmail.com`
- **SMTP_PORT**: SMTP server port. Defaults to `587`.
- **SMTP_USER**: SMTP username.
- **SMTP_PASS**: SMTP password.
- **SMTP_METHOD**: SMTP delivery method. Possible values: `smtp`. Defaults to `smtp`.
- **SMTP_OPENSSL_VERIFY_MODE**: SMTP openssl verification mode. Accepted values are `none`, `peer`, `client_once` and `fail_if_no_peer_cert`. SSL certificate verification is performed by default.
- **SMTP_STARTTLS**: Enable STARTTLS. Defaults to `true`.
- **SMTP_TLS**: Enable SSL/TLS. Defaults to `false`.
- **SMTP_SSL**: Enable SSL. Defaults to `false`.

  https://www.redmine.org/projects/redmine/wiki/EmailConfiguration#Error-TimeoutError-due-to-SSL-SMTP-server-connection
- **SMTP_AUTHENTICATION**: Specify the SMTP authentication method. Defaults to `:login` if `SMTP_USER` is set.
- **SMTP_CA_ENABLED**: Enable custom CA certificates for SMTP email configuration. Defaults to `false`.
- **SMTP_CA_PATH**: Specify the `ca_path` parameter for SMTP email configuration. Defaults to `/home/redmine/data/certs`.
- **SMTP_CA_FILE**: Specify the `ca_file` parameter for SMTP email configuration. Defaults to `/home/redmine/data/certs/ca.crt`.
- **IMAP_ENABLED**: Enable receiving email via IMAP. Defaults to `false`.

- **IMAP_USER**: IMAP username. Defaults to value of `SMTP_USER`. NOTE: May require escaping special characters for (CRON or Bash). Currently known: '%' needs to be escaped '%'
- **IMAP_PASS**: IMAP password. Defaults to value of `SMTP_PASS`. NOTE: May require escaping special characters for (CRON or Bash). Currently known: '%' needs to be escaped '%'
- **IMAP_HOST**: IMAP server host. Defaults to `imap.gmail.com`.
- **IMAP_PORT**: IMAP server port. Defaults to `993`.
- **IMAP_SSL**: IMAP enable SSL. Defaults to `true`.
- **IMAP_STARTTLS**: IMAP enabled STARTTLS. Defaults to `false`.
- **IMAP_INTERVAL**: The interval in minutes between checking emails. Defaults to `30`. Values allowed in the range `1 - 60`.
- **IMAP_FOLDER**: IMAP folder to read. Defaults to `INBOX`.
- **IMAP_MOVE_ON_SUCCESS**: Move emails that were successfully received to `MAILBOX` instead of deleting them.
- **IMAP_MOVE_ON_FAILURE**: Move emails that were ignored to `MAILBOX`.
- **INCOMING_EMAIL_UNKNOWN_USER**: How to handle emails from an unknown user. Accepted values are `ignore`, `accept` and `create`. Defaults to `ignore`.
- **INCOMING_EMAIL_NO_PERMISSION_CHECK**: Disable permission checking when receiving the email. Defaults to `false`.
- **INCOMING_EMAIL_NO_ACCOUNT_NOTICE**: Disable new user account notification. Defaults to `true`.
- **INCOMING_EMAIL_DEFAULT_GROUP**: Adds created user to foo and bar groups.
- **INCOMING_EMAIL_PROJECT**: Identifier of the target project.
- **INCOMING_EMAIL_PROJECT_FROM_SUBADRESS**: ADDR select project from subaddress of ADDR found in To, Cc, Bcc headers.
- **INCOMING_EMAIL_STATUS**: Name of the target status.
- **INCOMING_EMAIL_TRACKER**: Name of the target tracker.
- **INCOMING_EMAIL_CATEGORY**: Name of the target category.
- **INCOMING_EMAIL_PRIORITY**: Name of the target priority.
- **INCOMING_EMAIL_PRIVATE**: Create new issues as private.
- **INCOMING_EMAIL_ALLOW_OVERRIDE**: Allow email content to override attributes specified by previous options. Value is a comma separated list of attributes. See redmine documentation for acceptable values.
- **USERMAP_UID**: ID of user redmine inside container. Defaults to `1000`.
- **USERMAP_GID**: ID of group redmine inside container. Defaults to `1000`.

Plugins

The functionality of redmine can be extended using plugins developed by the community. You can find a list of available plugins in the Redmine Plugins Directory. You can also search for plugins on github.

Please check the plugin compatibility with the redmine version before installing a plugin.

Installing Plugins

Plugins should be installed in the `plugins` directory at the data store. If you are following the readme verbatim, on the host this location would be `/srv/docker/redmine/redmine/plugins`.

```
mkdir -p /srv/docker/redmine/redmine/plugins
```

To install a plugin, simply copy the plugin assets to the `plugins` directory. For example, to install the recurring tasks plugin:

```
cd /srv/docker/redmine/redmine/plugins
git clone https://github.com/nutso/redmine-plugin-recurring-tasks.git
```

For most plugins this is all you need to do. With the plugin installed you can start the docker image normally. The image will detect that a plugin has been added (or removed) and automatically install the required gems and perform the plugin migrations and will be ready for use.

If the gem installation fails after adding a new plugin, please retry after removing the `/srv/docker/redmine/redmine/tmp` directory

In some cases it might be necessary to install additional packages and/or perform some post installation setup for a plugin to function correctly. For such case the image allows you to install a `pre-install.sh` and `post-install.sh` script at the `/srv/docker/redmine/redmine/plugins` directory that will be executed everytime the image is started.

For example, the recurring tasks plugin requires that you create a cron job to periodically execute a rake task. To achieve this, create the `/srv/docker/redmine/redmine/plugins/post-install.sh` file with the following content:

```
## Recurring Tasks Configuration

# get the list existing cron jobs for the redmine user
crontab -u redmine -l 2>/dev/null >/tmp/cron.redmine

# add new job for recurring tasks if it does not exist
if ! grep -q redmine:recur_tasks /tmp/cron.redmine; then
  echo '@hourly /sbin/entrypoint.sh app:rake redmine:recur_tasks RAILS_ENV=production >> log/cro
  crontab -u redmine /tmp/cron.redmine 2>/dev/null
fi

# remove the temporary file
rm -rf /tmp/cron.redmine
```

```
## End of Recurring Tasks Configuration
```

Now whenever the image is started the `post-install.sh` script will be executed and the required cron job will be installed.

If you need to install additional packages to satisfy a plugins dependencies then install such packages using the `pre-install.sh` script.

Previously this image packaged a couple of plugins by default. Existing users would notice that those plugins are no longer available. If you want them back, follow these instructions:

```
cd /srv/docker/redmine/redmine/plugins
wget http://goo.gl/iJcvCP -O - | sh
```

Please Note: this plugin install script is not maintained and you would need to fix it if required (especially broken links)

Reloading plugins for development

Changing files in /srv/docker/redmine/redmine/plugins won't be automatically loaded. If you want to reload the plugins without restarting the docker, you can run the following.

```
docker exec -it redmine redmine-install-plugins
```

Uninstalling Plugins

To uninstall plugins you need to first tell redmine about the plugin you need to uninstall. This is done via a rake task:

```
docker run --name=redmine -it --rm \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6 \
  app:rake redmine:plugins:migrate NAME=plugin_name VERSION=0
```

Once the rake task has been executed, the plugin should be removed from the `/srv/docker/redmine/redmine/plugins/` directory.

```
rm -rf /srv/docker/redmine/redmine/plugins/plugin_name
```

Any configuration that you may have added in the `/srv/docker/redmine/redmine/plugins/post-install.sh` script for the plugin should also be removed.

For example, to remove the recurring tasks plugin:

```
docker run --name=redmine -it --rm \
  --volume=/srv/docker/redmine/redmine:/home/redmine/data \
  --volume=/srv/docker/redmine/redmine-logs:/var/log/redmine/ \
  sameersbn/redmine:6.0.6 \
  app:rake redmine:plugins:migrate NAME=recurring_tasks VERSION=0
rm -rf /srv/docker/redmine/redmine/plugins/recurring_tasks
```

Now when the image is started the plugin will be gone.

Themes

Just like plugins, redmine allows users to install additional themes. You can find a list of available themes in the Redmine Themes Directory

Installing Themes

Themes should be installed in the `themes` directory at the data store. If you are following the readme verbatim, on the host this location would be `/srv/docker/redmine/redmine/themes`.

```
mkdir -p /srv/docker/redmine/redmine/themes
```

To install a theme, simply copy the theme assets to the `themes` directory. For example, to install the gitmike theme:

```
cd /srv/docker/redmine/redmine/themes
git clone https://github.com/makotokw/redmine-theme-gitmike.git gitmike
```

With the theme installed you can start the docker image normally and the newly installed theme should be available for use.

Previously this image packaged a couple of themes by default. Existing users would notice that those themes are no longer available. If you want them back, follow these instructions:

```
cd /srv/docker/redmine/redmine/themes
wget http://goo.gl/deKDpp -O - | sh
```

Please Note: this theme install script is not maintained and you would need to fix it if required (especially broken links)

Reloading themes for development

Changing files in /srv/docker/redmine/redmine/themes won't be automatically loaded. If you want to reload the themes without restarting the docker, you can run the following.

```
docker exec -it redmine redmine-install-themes
```

Uninstalling Themes

To uninstall themes you simply need to remove the theme from the `/srv/docker/redmine/redmine/themes/` directory and restart the image.

```
rm -rf /srv/docker/redmine/redmine/themes/theme_name
```

For example, to remove the gitmike theme:

```
rm -rf /srv/docker/redmine/redmine/themes/gitmike
```

Now when the image is started the theme will be not be available anymore.

Maintenance

Creating backups

Only available in versions >`3.2.0-2`, >`3.1.3-1`, >`3.0.7-1` and >`2.6.9-1`

The image allows users to create backups of the Redmine installation using the `app:backup:create` command or the `redmine-backup-create` helper script. The generated backup consists of configuration files, uploaded files and the sql database.

Before generating a backup — stop and remove the running instance.

```
docker stop redmine && docker rm redmine
```

Relaunch the container with the `app:backup:create` argument.

```
docker run --name redmine -it --rm [OPTIONS] \
  sameersbn/redmine:6.0.6 app:backup:create
```

The backup will be created in the `backups/` folder of the Data Store. You can change the location using the `REDMINE_BACKUPS_DIR` configuration parameter.

> **NOTE**
>
> Backups can also be generated on a running instance using:
>
> ```
> docker exec -it redmine redmine-backup-create
> ```
>
> To avoid undesired side-effects, you are advised against creating a backup on a running instance.

Restoring Backups

Only available in versions >`3.2.0-2`, >`3.1.3-1`, >`3.0.7-1` and >`2.6.9-1`

Backups created using instructions from the Creating backups section can be restored using the `app:backup:restore` argument.

Before restoring a backup — stop and remove the running instance.

```
docker stop redmine && docker rm redmine
```

Relaunch the container with the `app:backup:restore` argument. Ensure you launch the container in the interactive mode `-it`.

```
docker run --name redmine -it --rm [OPTIONS] \
  sameersbn/redmine:6.0.6 app:backup:restore
```

A list of existing backups will be displayed. Select a backup you wish to restore.

To avoid this interaction you can specify the backup filename using the `BACKUP` argument to `app:backup:restore`, eg.

```
docker run --name redmine -it --rm [OPTIONS] \
  sameersbn/redmine:6.0.6 app:backup:restore BACKUP=1417624827_redmine_backup.tar
```

Automated backups

Only available in versions >`3.2.0-2`, >`3.1.3-1`, >`3.0.7-1` and >`2.6.9-1`

The image can be configured to automatically create backups `daily`, `weekly` or `monthly` using the `REDMINE_BACKUP_SCHEDULE` configuration option.

Daily backups are created everyday at `REDMINE_BACKUP_TIME`, which defaults to `04:00`. Weekly backups are created every Sunday at `REDMINE_BACKUP_TIME`. Monthly backups are created on the 1st of every month at `REDMINE_BACKUP_TIME`.

By default when automated backups are enabled, backups are held for a period of 7 days before they are deleted. When disabled, the backups are held for an infinite period of time. This behavior can be modified using the `REDMINE_BACKUP_EXPIRY` option.

Rake Tasks

The `app:rake` command allows you to run redmine rake tasks. To run a rake task simply specify the task to be executed to the `app:rake` command. For example, if you want to send a test email to the admin user.

```
docker run --name=redmine -d [OPTIONS] \
  sameersbn/redmine:6.0.6 app:rake redmine:email:test[admin]
```

You can also use `docker exec` to run rake tasks on running redmine instance. For example,

```
docker exec redmine /sbin/entrypoint.sh app:rake redmine:email:test[admin] RAILS_ENV=production
```

Similarly, to remove uploaded files left unattached

```
docker run --name=redmine -d [OPTIONS] \
  sameersbn/redmine:6.0.6 app:rake redmine:attachments:prune
```

Or,

```
docker exec redmine /sbin/entrypoint.sh app:rake redmine:attachments:prune RAILS_ENV=production
```

For a complete list of available rake tasks please refer
www.redmine.org/projects/redmine/wiki/RedmineRake.

Upgrading

To upgrade to newer redmine releases, simply follow this 4 step upgrade procedure.

- **Step 1**: Update the docker image.

```
docker pull sameersbn/redmine:6.0.6
```

- **Step 2**: Stop and remove the currently running image

```
docker stop redmine
docker rm redmine
```

- **Step 3**: Create a backup

```
docker run --name redmine -it --rm [OPTIONS] \
    sameersbn/redmine:x.x.x app:backup:create
```

Replace `x.x.x` with the version you are upgrading from. For example, if you are upgrading from version `2.6.4`, set `x.x.x` to `2.6.4`

- **Step 4**: Start the image

```
docker run --name=redmine -d [OPTIONS] sameersbn/redmine:6.0.6
```

When an upgrade is in progress the variable `REDMINE_WAS_UPDATED` will be defined and set to `yes`. This allows easy integration of individual upgrade-steps via `entrypoint.custom.sh`, `pre-install.sh`, and `post-install.sh`.

Shell Access

For debugging and maintenance purposes you may want access the containers shell. If you are using docker version `1.3.0` or higher you can access a running containers shell using `docker exec` command.

```
docker exec -it redmine bash
```

Development

Upgrading to next redmine release

- Commands to run to update image to next redmine release, examples are from 6.0.5 to 6.0.6

```
sed -i 's/6.0.5/6.0.6/g' VERSION README.md docker-compose* Dockerfile
vim Changelog.md # Update change log
make test-release # Runs the following
#  sudo rm -rf /srv/docker/redmine/ # Clean old run
#  sudo mkdir -p /srv/docker/redmine/redmine
#  sudo cp -rf $(CERTS_DIR) /srv/docker/redmine/redmine/ # Copy generated certificates
#  docker compose down
#  docker compose build
#  docker compose up # Test new build
./make_release.sh # Runs the following
#  git add -p
#  git commit -sS -m "release: $(cat VERSION)"
#  git tag -s $(cat VERSION) -m "$(cat VERSION)"
#  git push
#  git push origin --tags
```

- Open https://github.com/sameersbn/docker-redmine/releases and Draft new release
- Select tag 6.0.6 and set release title to 6.0.6
- Publish release
- Check https://quay.io/repository/sameersbn/redmine?tab=info and https://hub.docker.com/r/sameersbn/redmine/builds for build progress

References

```
* http://www.redmine.org/
* http://www.redmine.org/projects/redmine/wiki/Guide
* http://www.redmine.org/projects/redmine/wiki/RedmineInstall
```

Revision #1
Created 2025-08-24 18:40:26 UTC by Administrador
Updated 2025-08-24 18:43:54 UTC by Administrador