Wiki

Docker

- container "init" process

Documentation

- Generating Ruby+Rails documentation

Github

- Keeping Redmine repository in sync with Github without dedicated plugin (Apache CGI + Github Webhook)

Container init process

Problem

To enable running multiple processes, containers require process/service management. This is normally provided by some kind of init task (e.g. from sysvinit).

There are Docker-compatible replacements for full-fledged init's. Unfortunately they require either custom init scripts or service configurations (https://wiki.gentoo.org/wiki/Comparison_of_init_systems). The process of migration from OS-provided OpenRC init scripts is time consuming and error prone.

Usage of system's default sysvinit is hampered by following shortcomings:

- it mostly does not respond to Unix signals, which are used by Docker to manage containers (most importantly: signal termination),
- it does not stop properly on when Docker requests container to stop
  - attempt to stop container with init as PID 1 ends with error code 137:

```
CONTAINER ID  IMAGE         COMMAND       CREATED              STATUS                          P
ORTS  NAMES
b755c0f1b1d8  gentoo-base  "/sbin/init"  About a minute ago  Exited (137) 9 seconds ago
      gentoo-base
```

  - when invoking shutdown from within container, init process remains running afterwards, keeping container in running state:

```
# docker-compose top
gentoo-base
UID    PID    PPID   C    STIME   TTY      TIME       CMD
-------------------------------------------------------------
root   3510   3489   0    17:40   ?     00:00:00   init [0]
```

Solution

Nevertheless it is possible to use sysvinit inside Docker container as an init process. Required steps are following:

1. Change sysvinit to exit init process on hard shutdown (runlevel 0) with following patch

```
--- sysvinit-3.01/src/init.c    2021-12-13 20:21:26.000000000 +0100
+++ sysvinit-3.01/src/init.c    2022-04-18 01:21:47.966751774 +0200
@@ -2367,6 +2367,11 @@
      read_inittab();
      fail_cancel();
      setproctitle("init [%c]", (int)runlevel);
+
+      /*
+       * Exit on halt - causes Docker container to stop.
+       */
+      if (runlevel == '0') exit(0);
    }
  }
        Write_Runlevel_Log(runlevel);
```

On Gentoo it's enough to put this patch inside */etc/portage/patches/sys-apps/sysvinit/exit-on-halt.patch* and reemerge sysvinit.

2. Change Docker signal for container termination to SIGINT and set appropriate action in inittab

Container's *docker-compose.yml*:

```
services:
  gentoo-base:
    ...
```

```
    stop_signal: SIGINT
```

/etc/inittab inside container - replace reboot action with shutdown:

```
# What to do at the "Three Finger Salute".
ca:12345:ctrlaltdel:/sbin/shutdown -h now
```

Generating Ruby+Rails documentation

```
cd /var/www/localhost/htdocs/rubydocs

git clone --branch v3_3_0 --depth 1 https://github.com/ruby/ruby.git
sdoc --all --exclude='./test/*' --exclude='./spec/*' --main=ruby/README.md --output=ruby-3.3.0 rub
y

git clone --branch v7.1.2 --depth 1 https://github.com/rails/rails.git
sdoc --all --exclude='./*/test/*' --main=rails/README.md --output=rails-7.1.2 rails

sdoc-merge --op=ruby-3.3-rails-7.1 ruby-3.3.0 rails-7.1.2

rm -rf ruby/ ruby-3.3.0/ rails/ rails-7.1.2/
```

This is also available as Redmine HowTo

Keeping Redmine repository in sync with Github without dedicated plugin (Apache CGI + Github Webhook)

This is a solution in case you don't want to install additional plugins just to keep repository synchronised. It requires you to have Apache webserver with access to repository you are trying to sync. Apache has to support running CGI scripts.

Clone Github repository

Clone repository and make sure it is accessible by webserver:

```
mkdir /var/lib/redmine/repo
chown apache /var/lib/redmine/repo
su -u apache git -C /var/lib/redmine/repo clone https://github.com/username/repo_name.git
```

Enable WS for repository management in Redmine

Go to https://your.redmine.com/settings?tab=repositories and:

- select: *Enable WS for repository management*
- generate a repository management WS API key and save it for next step

Prepare CGI script

Any script you run on your server will do. Below is an example of Bash script that pulls git repository and notifies Redmine to fetch changesets (substitute <repository-api-key> with your own):

```
#!/bin/sh
# Requires: jq

REPO_PATH='/var/lib/redmine/repo'

# Empty stdin, Apache issue https://bz.apache.org/bugzilla/show_bug.cgi?id=44782
REPO_NAME=$(cat <&0 | jq '.repository.name' | tr -cd 'A-Za-z0-9_-')
if [ -z "${REPO_NAME}" ]  || [ ! -d "${REPO_PATH}/${REPO_NAME}" ]; then
  echo "Status: 400 Bad Request"
  echo "Content-Type: text/plain; charset=utf-8"
  echo
  echo "project: unrecognized"
  exit 0
fi

/usr/bin/git -C "${REPO_PATH}/${REPO_NAME}" pull -n -q
result1=$?

PROJECT_NAME=$(echo "${REPO_NAME}" | tr '_' '-')
/usr/bin/curl --max-time 60 -s "https://your.redmine.com/sys/fetch_changesets?id=${PROJECT_NAME}&k
ey=<repository-api-key>" >/dev/null
result2=$?

if [[ $result1 && $result2 ]]; then
  echo "Status: 200 OK"
else
  echo "Status: 500 Internal Server Error"
fi

echo "Content-Type: text/plain; charset=utf-8"
echo
echo "project: ${PROJECT_NAME}"

if [[ $result1 ]]; then
  echo "git pull: ok"
```

```
else
  echo "git pull: failed"
fi

if [[ $result2 ]]; then
  echo "fetch changesets: ok"
else
  echo "fetch changesets: failed"
fi
```

Let's say you save this script under: */var/www/cgi-bin/update-repo.cgi*

You can test if script executes properly:

```
echo <copy-input-from-github-webhook-request> | sudo -u apache /var/www/cgi-bin/update-repo.cgi
```

Configure Apache to run script whenever particular URL is requested

Inside VirtualHost of your choice just add:

```
  ...
  # Github webhook for repository pull/update
  ScriptAlias /update-repo.cgi /var/www/cgi-bin/update-repo.cgi
  <Directory /var/www/cgi-bin/>
    Options ExecCGI
    AllowOverride None
    Require all granted
  </Directory>
  ...
```

In case you use the same VirtualHost to proxy requests to your Redmine rails server, you should exclude your special URL from being proxied with:

```
ProxyPass /update-repo.cgi !
```

Configure Github Webhook

Go to your Github repository page, choose *Settings -> Webhooks -> Add webhook*. Then set:

- Payload URL: https://your.virtualhost.com/update-repo.cgi
- Content type: application/json
- Which events would you like to trigger this webhook?: Just the push event.
- Active: yes

Update webhook and you're done.