Technical Manual of the Universal Webserver

Organisation: Copyright (C) 2019-2020 Olivier Boudeville

Contact: about (dash) universal-webserver (at) esperide (dot) com

Creation date: Saturday, May 2, 2020

Lastly updated: Wednesday, May 27, 2020

Status: Work in progress

Version: 0.0.2

Dedication: Users and maintainers of the `Universal Webserver`.

Abstract: The Universal Webserver, part of the Universal Server umbrella project, provides a multi-domain, multi-virtualhost webserver integrating various web-related services.

We present here a short overview of these services, to introduce them to newcomers.

The next level of information is to read the corresponding source files, which are intensely commented and generally straightforward.

Table of Contents

Overview

We present here a short overview of the services offered by the *Universal Webserver*, to introduce them to newcomers.

The goal of **US-Web** is to provide an integrated web framework in order:

- to better operate websites based on virtual hosting, so that a networked computer can serve as many websites corresponding to as many domains as wanted; this involves reading and interpreting vhost and other configuration information, handling properly 404 errors, producing access logs that are adequate for web analytics, rotating all logs, etc.
- to link to the Universal Server optionally (i.e. if available, knowing both should be able to run in the absence of the other), in order to offer a web front-end for it

Beyond this document, the next level of information about US-Web is to read the corresponding source files, which are intensely commented and generally straightforward.

Layer Stack

From the highest level to the lowest, as summarised here, a software stack involving the Universal Webserver usually comprises:

- the *Universal Webserver* services themselves (i.e. this US-Web layer)
- Cowboy (for a small, fast and modern web framework)
- [optional] Awstats (for the analysis of access log files)
- US-Common (for US base facilities)
- Ceylan-Traces (for advanced runtime traces)
- Ceylan-WOOPER (for OOP)
- Ceylan-Myriad (as an Erlang toolbox)
- Erlang (for the compiler and runtime)
- GNU/Linux

The shorthand for `Universal Webserver` is `uw`.

Server Deployment

For that the `prod` profile defined in the context of rebar3 shall be used.

Currently we prefer re-using the (supposedly already installed) local Erlang environment on the server (to be shared across multiple services), so by default ERTS is not included in a US-Web release.

Sources are not included either, as we prefer re-rolling a release to editing and compiling code directly on a server.

To generate from scratch such a (mostly) standalone release, one may use:

```
$ make release-prod
```

It should generate a tarball such as `us_web-x.y.z.tar.gz`

The `export-release` allows in the same movement to lightly update a pre-existing release and also to transfer it to any target server, designated by setting the `WEB_SRV` to the FQDN of this server.

So we recommend running:

```
$ make export-release
 Generating rebar3 us_web.app file
 Compiling us_web from XXX/us_web
 ===> Verifying dependencies...
 ===> Compiling myriad
 Hiding unsupported sources
 Populating the include symlinks
 [...]
```

We recommend installing a release in `REL_BASE_ROOT=/opt/universal-server`:

```
$ mv /tmp/us_web-x.y.z.tar.gz ${REL_BASE_ROOT}
$ cd ${REL_BASE_ROOT}
$ tar xvf us_web-x.y.z.tar.gz
```

Then various steps are necessary in order to have a functional release.

We automated the full deployment process for that: once the release has been transferred to the server (possibly thanks to the aforementioned `export-release` target), one may rely on our deploy-us-web-release.sh script. One may take inspiration from it in order to devise one's deployment scheme.

Some related information are specified below.

Configuring the Universal-Webserver

As explained in start-us-web.sh and in class_USWebConfigServer.erl, the US configuration files will be searched through various locations.

Let's name `US_CFG_ROOT` the actual directory in which they lie; it is typically either `~/.config/universal-server/` (in development mode), or `/etc/xdg/universal-server/` (in production mode).

Note that, as these files might contain sensitive information (ex: Erlang cookies), they shall be duly protected.

Let's say that the UNIX name chosen for the US user is `us-user`, the one of the US-web user is `us-web-user` and the US group (containing both users, and possibly only them) is `us-group`.

Then we should have in terms of permissions, supposing the us.config designates, in its us_web_config_filename entry, foobar-us-web-for-production.config as the name of the US-Web configuration file[1], 640:

```
-rw-r----- 1 us-user     us-group [...] us.config
-rw-r----- 1 us-web-user us-group [...] foobar-us-web-for-production.config
```

In a Development Setting

The US main configuration file, us.config, is in a directory (US_CFG_ROOT) that is ~/.config/universal-server/ here. This US-level configuration file will reference a US-Web counterpart configuration file, probably in the same directory.

The US-Web configuration file may define a us_web_app_base_dir entry. If not, this application directory will then be found thanks to the US_WEB_APP_BASE_DIR environment variable (if defined, typically through ~/.bashrc); otherwise, as a last resort, an attempt to guess it will be done.

The US webserver may be then run thanks to make debug, from the relevant us_web directory (typically the root of a GIT clone located in the user's directory).

In such a development context, in us_web/conf/sys.config, we recommend to let the batch mode disabled (just let the default {is_batch,false}), so that a direct, graphical trace supervision is enabled (provided that a relevant trace supervisor is available, see Traces for that).

In a Production Setting

The start/stop management scripts will be run initially as root and must access the us.config file. Then, once run, us_web will most probably switch to a dedicated user (see the us_web_username entry in the US-Web configuration file), who will need in turn to be able to read the us.config file and any related one (ex: for US-Web, here supposed to be named foobar-us-web-for-production.config).

As a result, a relevant configuration directory (denoted US_CFG_ROOT in this document), in that shared setting, is the standard /etc/xdg one, resulting in the /etc/xdg/universal-server directory to be used.

As mentioned, care must be taken so that root and also the US and US-Web users can read the content of that directory - at least the US and US-Web configuration files in it - and that the other users cannot.

For that, a dedicated us-group group can be created, and any web user (ex: us-web-user) shall belong to that group. For example:

```
$ id us-web-user
uid=1002(us-web-user) gid=1002(us-web-user) groups=1002(us-web-user),1007(u
```

Then, in /etc/xdg/universal-server, for the US and US-Web configuration files:

```
$ chown us-user us.config
$ chown us-web-user foobar-us-web-for-production.config
```

[1] They shall be in the same US_CFG_ROOT directory (discussed below), and may be symbolic links.

```
$ us_files="us.config foobar-us-web-for-production.config"
$ chgrp us-group ${us_files}
$ chmod 640 ${us_files}

$ chgrp us-group /etc/xdg/universal-server
$ chmod 700 /etc/xdg/universal-server
```

We recommend directly setting the `us_web_app_base_dir` entry to the relevant, absolute path.

Let's name here `US_WEB_REL_ROOT` the root of the US-Web release of interest (ex: corresponding to `${REL_BASE_ROOT}/us_web-latest/`) and `US_WEB_APP_ROOT` the root of the corresponding US-Web application (ex: corresponding to `${US_WEB_REL_ROOT}/lib/us_web-l`

A `systemd` service shall be declared for US-Web, in `/etc/systemd/system`; creating there, as root, a symbolic link to `${US_WEB_APP_ROOT}/priv/conf/us-web.service` will suffice.

This service requires `start-us-web.sh` and `stop-us-web.sh`. Adding for user convenience `get-us-web-status.sh`, they should all be symlinked that way, still as root:

```
$ cd /usr/local/bin
$ for f in start-us-web.sh stop-us-web.sh get-us-web-status.sh; \
  do ln -s ${US_WEB_APP_ROOT}/priv/bin/$f ; done
```

The log base directory (see the `log_base_directory` entry) shall be created and writable; for example:

```
$ LOG_DIR=/var/log/universal-server
$ mkdir -p ${LOG_DIR}
$ chown us-user ${LOG_DIR}
$ chgrp us-group ${LOG_DIR}
$ chmod 770 ${LOG_DIR}
```

In such a production context, in `sys.config` (typically located in `${US_WEB_REL_ROOT}/releases/late` we recommend to enable batch mode (just set `{is_batch,true}`), so that by default no direct, graphical trace supervision is triggered (a server usually does not have a X server anyway).

The traces may then be supervised and browsed remotely (at any time, and any number of times), from a graphical client (provided that a relevant trace supervisor is available locally, see Traces for that), by running the monitor-us-web.sh script.

For that the relevant settings (notably which server host shall be targeted, with what cookie) shall be stored in that client, in a `us-monitor.config` file that is typically located in `~/.config/universal-server/`.

Usage Recommendations

In terms of security, we would advise:

- to stick to the **latest stable version** of all software involved (including US-Web and all its stack, Erlang, and the operating system itself)
- to apply a streamlined, reproducible **deployment process**, preferably based on our deploy-us-web-release.sh script

- to rely on dedicated, **different, low-privileged users and groups** for US and US-Web, which both rely on `authbind`; refer to our start-us-web.sh script for that; see also the `us_username` key of US-Common's us.config, and the `us_web_username` key of the US-Web configuration file it refers to
- still in `us.config`, to set:
  - a strong-enough Erlang **cookie**: set the `vm_cookie` key to a well-chosen value, possibly a random one deriving from an output of `uuidgen`
  - possibly a **limited TCP port range** (see the `tcp_port_range` key)
  - the **execution context** to `production` (see the `execution_context` key)
- to use also the stop-us-web.sh counterpart script, and to have them triggered through `systemd`; we provide a corresponding us-web.service **unit file** for that, typically to be placed in `/etc/systemd/system` and whose `ExecStart/ExecStop` paths shall preferably be symlinks pointing to the latest deployed US-Web release (ex: `/opt/universal-server/us_web-latest`)
- to ensure that a **firewall** blocks everything from the Internet by default, including the EPMD port(s) (i.e. both the default Erlang one and any non-standard one specified through the `epmd_port` key defined in `us.config`); one may get inspiration from our iptables.rules-Gateway.sh script for that
- to **monitor regularly** both:
  - the US-Web server itself (see our monitor-us-web.sh script for that, relying on the trace supervisor provided by the Ceylan-Traces layer)
  - the remote, browser-based, accesses made to the hosted websites, typically by enabling the US-Web "meta" feature, generating and updating automatically a dedicated website displaying in one page all hosted websites and linking to their web analysis report; refer to the `log_analysis` key of the US-Web configuration file (ex: see us-web-for-tests.config as an example thereof)

Licence

The `Universal Webserver` is licensed by its author (Olivier Boudeville) under the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of this license, or (at your option) any later version.

This allows the use of the Universal Webserver code in a wide a variety of software projects, while still maintaining copyleft on this code, ensuring improvements are shared.

We hope indeed that enhancements will be back-contributed (ex: thanks to merge requests), so that everyone will be able to benefit from them.

Current Stable Version & Download

As mentioned, the single, direct prerequisites of the Universal Webserver are:

- Cowboy (version 2.8 or above)

- Awstats as an optional, runtime-only dependency (version 7.8 or above)
- US-Common

The latter relies on Ceylan-Traces, which implies in turn Ceylan-WOOPER, then Ceylan-Myriad and Erlang.

We prefer using GNU/Linux, sticking to the latest stable release of Erlang, and building it from sources, thanks to GNU `make`.

We recommend indeed obtaining Erlang thanks to a manual installation; refer to the corresponding Myriad prerequisite section for more precise guidelines.

The build of the US-Web server is driven by rebar3, which can be obtained by following our guidelines.

If a tool for web analysis is needed (typically if enabling a meta website), this tool must be installed beforehand. Currently US-Web supports Awstats, which can be obtained thanks to your distribution of choice (ex for Arch Linux: `pacman -S awstats`[2]).

If wanting to be able to operate on the source code of the Ceylan and/or US dependencies, you may define appropriate symbolic links in a `_checkouts` directory created at the root one's `US-Web` clone, these links pointing to relevant GIT repositories (see the `create-us-web-checkout` make target for that).

Using Cutting-Edge GIT

This is the installation method that we use and recommend; the Universal Webserver `master` branch is meant to stick to the latest stable version: we try to ensure that this main line always stays functional (sorry for the pun). Evolutions are to take place in feature branches and to be merged only when ready.

Once Erlang and possibly Awstats are available, it should be just a matter of executing our get-us-web-from-sources.sh script for downloading and building all dependencies at once, and run a test server (use its `--help` option for more information.

For example:

```
$ cd /tmp
$ wget https://raw.githubusercontent.com/Olivier-Boudeville/us-web/master/p
$ sh ./get-us-web-from-sources.sh --checkout
Switching to checkout mode.

 Installing US-Web in /tmp...

 Cloning into 'us_web'...
 [...]
 ===> Compiling us_web
 Starting the us_web release (EPMD port: 4506):
 [...]
 US-Web launched, please point a browser to http://localhost:8080 to check

$ firefox http://localhost:8080 &
```

One shall then see a text-only page such as:

[2]To avoid a future reading access error: `chmod -R +r /usr/share/webapps/awstats/icon`.

```
This is static website D. This is the one you should see if pointing
to the default virtual host corresponding to the local host. This
shows that the US-Web server is up and running.
```

Understanding the role of the main US configuration file and of the corresponding US-Web configuration file for this test should be fairly straightforward.

Based on that, devising one's version of them should allow to have one's US-Web server running at the cost of very little efforts.

OTP Considerations

As discussed in these sections of Myriad, WOOPER, Traces and US-Common, the Universal Webserver *OTP application* is generated out of the build tree, ready to result directly in an *(OTP) release*. For that we rely on rebar3, relx and (possibly) hex.

Then we benefit from a standalone, complete Universal Webserver able to host as many virtual hosts on any number of domains as needed.

As for Myriad, WOOPER, Traces and US-Common, most versions of the Universal Webserver will be also published as Hex packages.

For more details, one may have a look at rebar.config.template, the general rebar configuration file used when generating the Universal Webserver OTP application and release (implying the automatic management of all its dependencies).

Support

Bugs, questions, remarks, patches, requests for enhancements, etc. are to be reported to the project interface (typically issues) or directly at the email address mentioned at the beginning of this document.

Please React!

If you have information more detailed or more recent than those presented in this document, if you noticed errors, neglects or points insufficiently discussed, drop us a line! (for that, follow the Support guidelines).

Ending Word

Have fun with the Universal Webserver!

Universal Webserver