STEREO - Software research platform on software quality and engineering

A Short Introduction to the Alitheia-Core Platform: Installation and Running

Vassilios Karakoidas (bkarak@aueb.gr)

This is a mini guide to the Alitheia Core platform. Its purpose is to provide a tutorial for the build and installation process.

Downloading the Source Code

The Alitheia Code source code is available in *GitHub* (http://www.github.com). The project can be found in the following URL:

https://github.com/istlab/Alitheia-Core

To download the code, there are two options:

1. Checkout the repository with *Git* (http://git-scm.com/download)

   Without a Github account, the code can be checked out as follows:

   git clone git://github.com/gousiosg/Alitheia-Core.git (read-only access)

2. Download the code as a Zip file from:

   https://github.com/istlab/Alitheia-Core/downloads

Dependencies

The Alitheia-Core platform is written mainly in Java, thus the latest JDK (1.6 or 1.7) is required in order to build it. The build process is orchestrated by the maven utility (version 3+, http://maven.apache.org/) and it is required.

The Alitheia-Core platform uses numerous third party libraries and dependencies. These are automatically resolved and downloaded by the maven tool, thus an active internet connection is required (at least the first time) to complete the building process.

The Alitheia-Code metadata are saved in a relational database. MySQL (version 5+) is used by default, so a running instance of MySQL with specific database and user configuration must be available in order to run Alitheia-Core.

Choosing the Database Backend

Alitheia-Core supports two database backends; *H2* and *MySQL*.

By default, Alitheia-Core uses *H2* (http://www.h2database.com/html/main.html) as its database backend. This should be ok for a local installation and experimentation, but in general it is recommended to use MySQL.

Enabling Support for MySQL

To enable support for MySQL the following steps must be followed:

1. Edit the file «Alitheia-Core/pom.xml»
    a. Comment out the following lines:

```
<eu.sqooss.db>H2</eu.sqooss.db>
<eu.sqooss.db.host>localhost</eu.sqooss.db.host>
<eu.sqooss.db.schema>alitheia;LOCK_MODE=3;MULTI_THREADED=true</eu.sqooss.db.schema>
<eu.sqooss.db.user>sa</eu.sqooss.db.user>
<eu.sqooss.db.passwd></eu.sqooss.db.passwd>
<eu.sqooss.db.conpool>c3p0</eu.sqooss.db.conpool>
```

    b. Uncomment the following lines:

```
<eu.sqooss.db>MySQL</eu.sqooss.db>
<eu.sqooss.db.host>localhost</eu.sqooss.db.host>
<eu.sqooss.db.schema>alitheia</eu.sqooss.db.schema>
<eu.sqooss.db.user>alitheia</eu.sqooss.db.user>
<eu.sqooss.db.passwd>alitheia</eu.sqooss.db.passwd>
<eu.sqooss.db.conpool>c3p0</eu.sqooss.db.conpool>
```

2. Edit the MySQL main configuration file (usually named «/etc/my.cnf») and add the following lines:

```
default-storage-engine=innodb
transaction_isolation=READ-COMMITTED
```

    The above lines enable *innodb* as default.

3. Create an empty database named «alitheia». Then create a database user named «alitheia» with password «alitheia» and grant full control over the database (@localhost).

Note: The MySQL configuration should always be enabled prior to compilation.

Building Alitheia-Core

After downloading the source code, a directory named Alitheia-Core is created. To build Alitheia-Core the following command must be executed inside the directory:

mvn install

This process may take a while. This process requires Internet connection and will download on-demand libraries and other dependencies.

Running Alitheia-Core

To run the Alitheia-Core execute the following command:

mvn pax:provision

then visit the web interface http://localhost:8080/

Note: The first time Alitheia-Core server runs, the database tables are created. A known-bug (will be addressed soon) dictates to kill the Alitheia-Core process (press ctrl-c in the console) and re-run the mvn pax:provision command.

Administration Web Interface

If Alitheia-Core is correctly initialised (*Note: this part will run EVEN if the database is not properly initialised and all SQO-OSS functions will fail)*, the web interface will be available in http://localhost:8080/ (Figure 1).

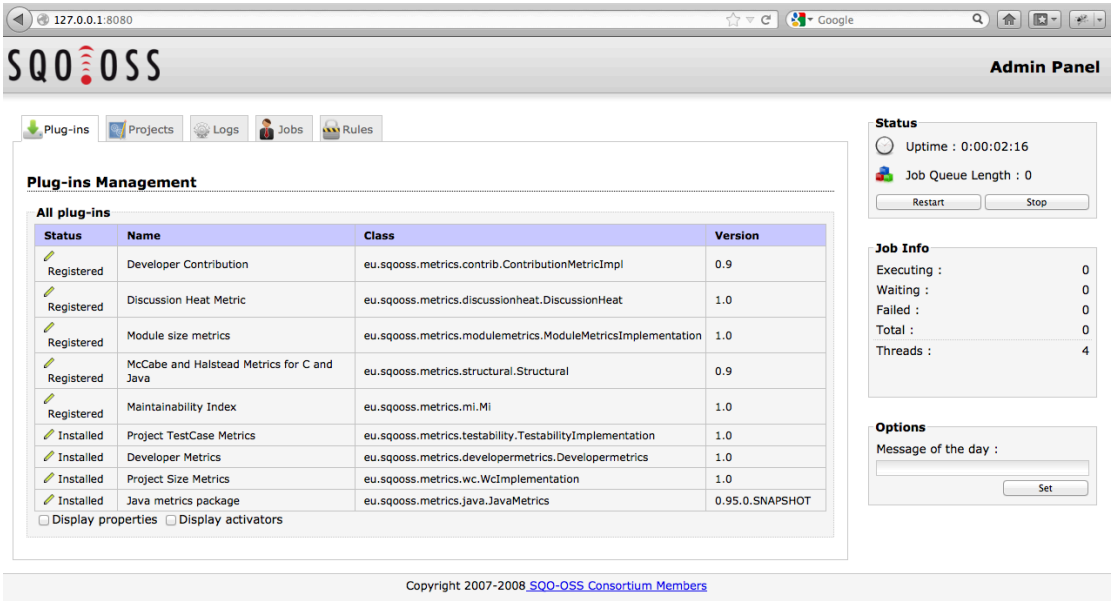

Figure 1: The Alitheia-Core administration Screen

The administration interface provides five (5) tabs, which cover several basic functions of the system:

- **Plug-ins:** The plug-ins calculate the metrics and store them in the database.
- **Projects:** A list of software projects that are stored in the system.
- **Logs:** A simple web page that displays raw logs from the system.
- **Jobs:** Detailed information regarding the jobs (metric update and repository synchronisation) that are executed by the system.
- **Rules:** So far, the rules tabs is inactive.

The Plug-ins Tab

When the system is first initialised, a series plug-ins must be activated. To do so, just select the «Plug-ins» tab and click on the desired plug-in that are available (Figure 1).

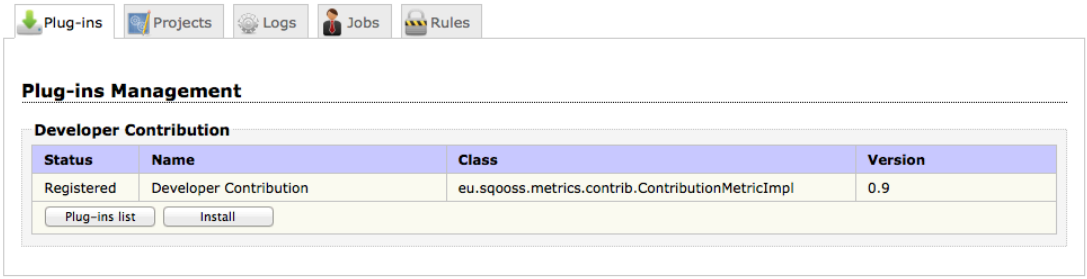

Figure 2: Plug-in Management for Developer Contribution Plug-in

For example, if the «Developer Contribution» plug-in is selected, then the «Plug-ins Management» page will be revealed (Figure 2). Select install to enable it as active plug-in (*Note: Plug-in's status will change from «Registered» to «Installed»)*.

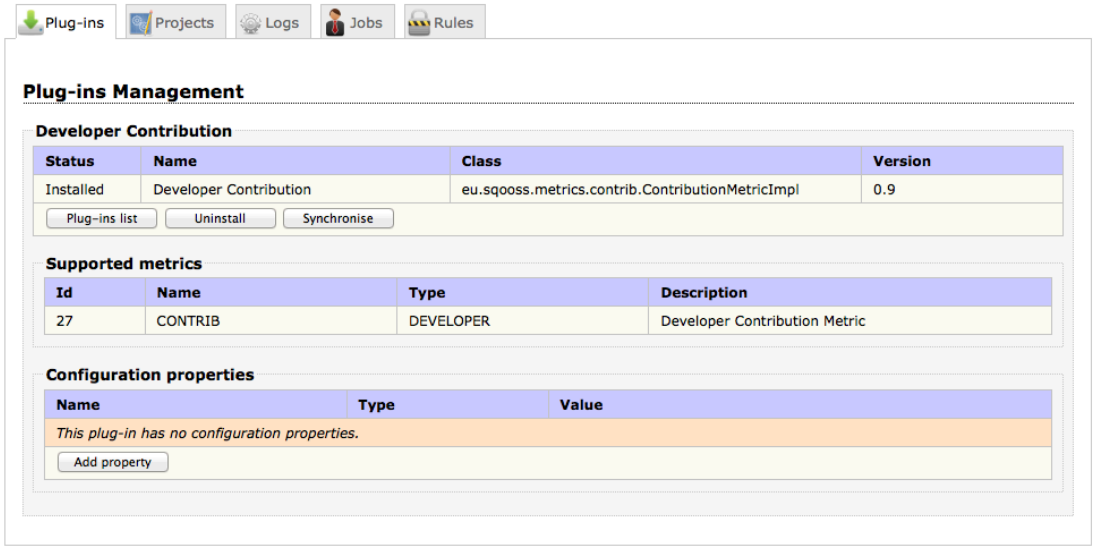

Figure 3: «Plug-in Management» web page for Developer Contribution plug-in

When the plug-in is installed (Figure 3), it can be uninstall if the «Uninstall» option is selected and synchronised, which means to execute this plug-in to all projects that has never been executed in the past. All this information is maintained in the Alitheia-Core database.

Adding a Project to Alitheia-Core

To add a project, go to the Projects tab in the Web administration console, then click on the «Add Project» button. A new form will appear (Figure 4). You should fill in the locations of the project's Subversion repository mirror, the top-level directory of the project's maildir-formatted mailing list mirrors and the directory containing the project's bugs in Bugzilla XML format. If you do not have any such data, you may try the ones on offer at the project's page (http://www.sqo-oss.org/download).

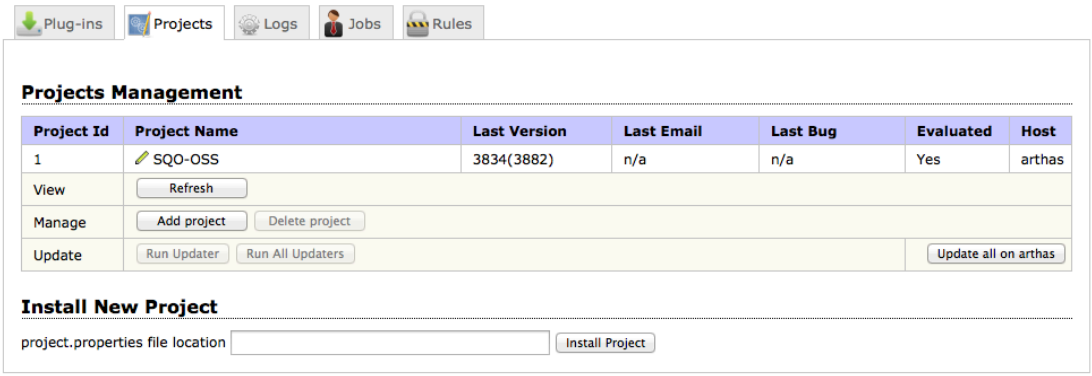

Figure 4: The «Projects Management» Tab

After you click on Add Project button, the project will be added to the database and the various data updaters will start processing the project metadata.

As you can see in Figure 4, the SQO-OSS Project is already installed in the system. This project has been downloaded from the download page of the SQO-OSS site (http://www.sqo-oss.org/download). The easiest way to install a project to the system is through the *project.properties* file. The project's data should follow a specific structure.

Mirror Directory Layout

The Alitheia-Core tool is not concerned with mirroring data from projects; it expects data to be mirrored externally. This choice was made at the beginning of the project to compensate for the large number of different data sources that the system should work with. Several of those already provide the means for synchronizing data across sites. For example:

- Subversion (SVN) offers the *svnsync* tool for mirroring repositories while distributed version control systems already copy the full repository history on checkout.
- Configuring the mail delivery subsystem on the mirroring host to store incoming messages to a specified directory and then subscribing to each mailing list can mirror mailing lists. Mailing lists archive mirroring requires custom scripts per project site, although various mailing list archiving web sites (e.g. marc) offer a unified view over thousands of mailing lists for common projects.
- Bug tracking tools, such as Bugzilla, usually offer programmatic interfaces to retrieve individual bug histories through programmatic interfaces.

Figure 5 shows the structure that Alitheia Core expects the project mirrors to look like. All projects directories are stored under a single directory, the mirroring root. Each project directory contains three subdirectories, one for each mirrored data resource. Since a project can either be hosted in a Git or an SVN repository only one of the Git and SVN directories can be present. All bug reports for a project are stored in a single directory. Finally, email messages are stored in a directory per mailing list; each mailing list directory is formatted according to the Maildir standard. Incoming emails are stored in the new sub-directory while processed emails are stored under directory «cur».

Updating a Project's Metadata and Executing Plug-ins

To see more detailed information regarding the installed project just click on it in the Projects' table. A new page will appear.

The easiest way to update database metadata and execute all plug-ins is by pressing the button «Run All Updaters».

Playing with Metric Data

Each project's metadata and the calculated metrics are stored in the database. There are two methods to retrieve them:

1. By using the REST API (http://localhost:8080/api). See the documentation for a detailed list of calls: http://www.sqo-oss.org/rest
2. Directly from the database server. The database follows the schema: http://www.sqo-oss.org/dbschema

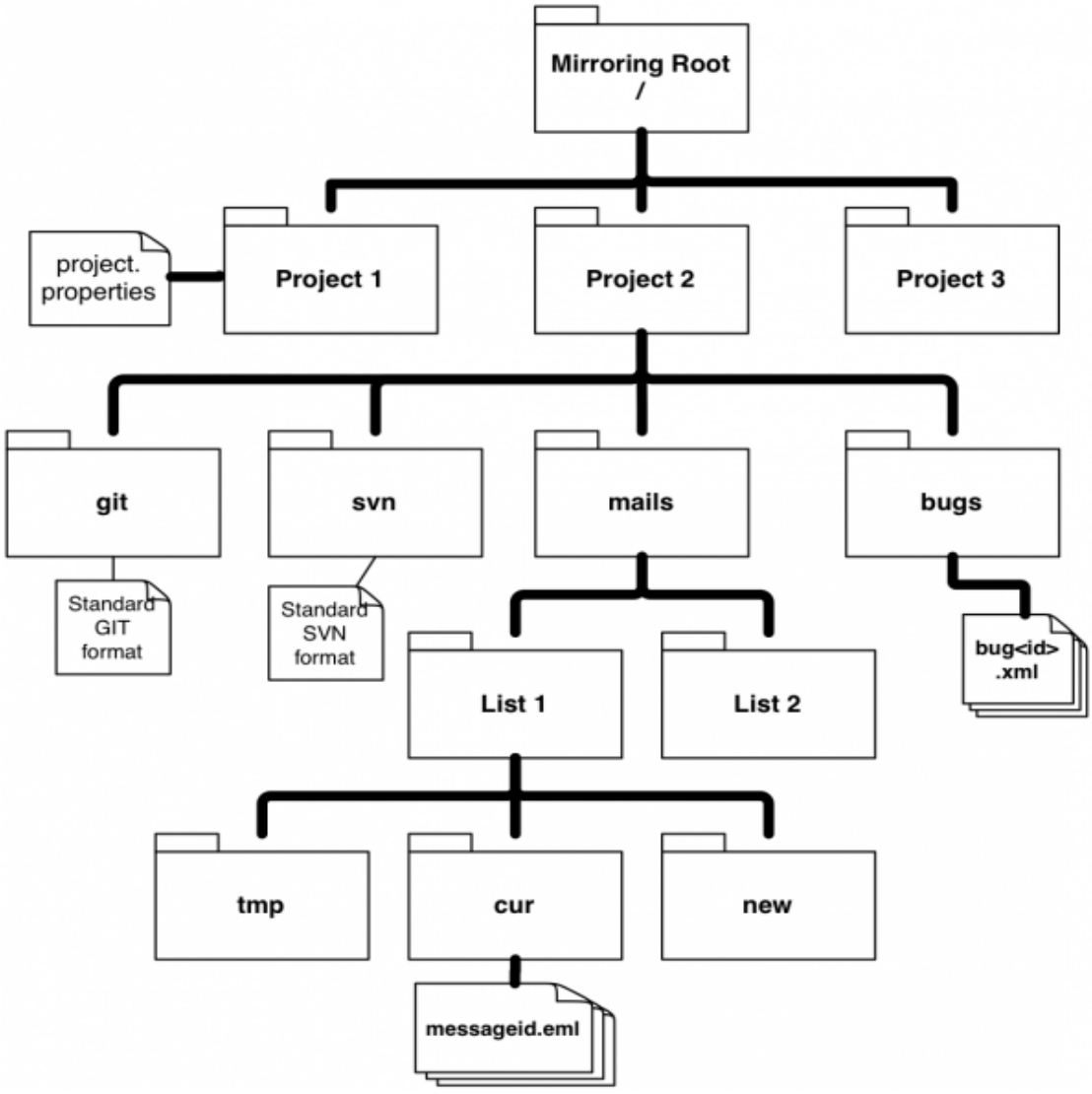

Figure 5: Mirroring Repository Layout

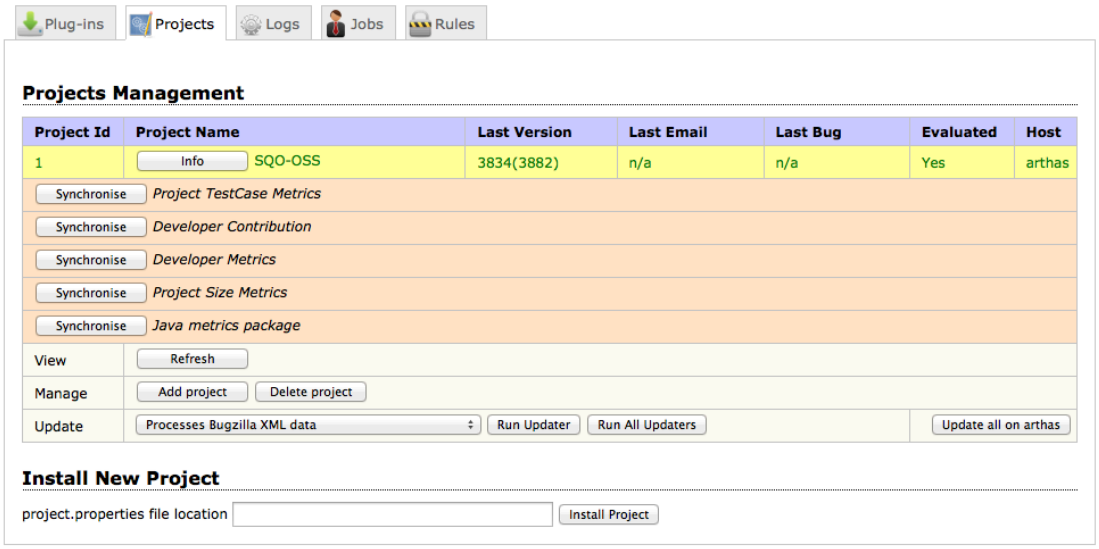

Figure 6: Expanded Project Information

When Things Go Wrong

When things go wrong (and with Alitheia-Core being a research tool things can often go wrong), all the user has to do is restart the metadata or plug-in synchronisation options. Alitheia-Core usually isolates failures and tries hard to maintain metadata consistency. If there is an error, you could do the following:

- Check the logs. There is a lot of information that is being written in various logs. The logs reside in «Alitheia-Core/runner/logs». Errors and debugging information are recorded in at least the following logs:
    - **alitheia.log:** General debugging info and job results (or errors)
    - **updater.log:** Log specific to the updater service functions. Tons of debugging messages.
    - **webadmin.log:** Log specific to the Web Administration console
    - **hibernate.log:** Log specific to the Hibernate ORM system. Check here for exceptions when creating new plug-ins that use custom tables.
- See the reason (exception) that caused the problem. All jobs in Alitheia-Core will die with an exception, which is recorded in either the log or the Jobs web admin tab or both.