The Upstream Tracker Project

Setting up the stage

The project is split into two parts – the web frontend based on Rails and the backend based on Python. In this document, we will look only at the web frontend – how it works, how to set it up, how to use it and it's API calls. In this section, I shall explain in brief, how to set things up using Ubuntu OS as an example.

Installing Ruby on Rails

There are two ways to install Ruby on Rails on an Ubuntu machine. The easiest way is use the installer script while experienced users may want to install using terminal commands themselves. Here, I will try to go over both these methods.

Using the script

I came across a rails installer script while reading the blog at http://blog.sudobits.com/2012/05/02/how-to-install-ruby-on-rails-in-ubuntu-12-04-lts/. The script is not written by me. However, it provides a convinient and a hasle free method to get started.

1. Download the bash script from the link mentioned on the blog (https://github.com/rkjha/RailsOnUbuntu/blob/master/rails-installer.sh). I assume the file to be named rails-installer.sh. Substiture youre file name in the commands below if it is not the same.
2. Open terminal. Go to Edit -> Preferences -> Title and Command tab and check the “Run Command as login shell” checkbox.
3. Use the cd command to switch to the directory where the script was downloaded. Use *chmod +x rails-installer.sh* to set the executable bit.
4. Run the script using *./rails-installer.sh*. Once done, reboot the system and type in “*rails -v*” in the terminal. You should see version number greater than 3. If not, try repeating the above steps. If your version number is above 3, you are good to go.

Manual

1. Install git and curl using : *sudo apt-get install git curl*
2. Install RVM and dependencies : *curl -L get.rvm.io | bash -s stable*
3. Load the RVM using : *source ~/.rvm/scripts/rvm*
4. Type in *rvm requirements* in the terminal to find out additional dependencies that are required. Copy and paste the output to install them.
5. Rails requires a Javascript Runtime to work. In this case, let's install node.js by issuing the command : *sudo apt-get install nodejs*
6. To install Ruby, run : *rvm install 1.9.3*.
7. Set 1.9.3 as the default version of Ruby using : *rvm use 1.9.3 --default*
8. Check the version of ruby installed using : *ruby -v* (Should show 1.9.3)
9. To install Ruby on Rails, install the rails gem for ruby using : *gem install rails*

Testing Rails Environment

To test the rails environment that has been set up, run the following commands.

rails new sample_app
cd sample_app
rails s

Open a web browser and navigate to http://localhost:3000/. This should show you a rails page if everything is working fine. If you get an error page instead, it's possible that you missed out one of the steps above.

Running the Web Frontend

To run the web frontend, clone the files from the git repository using

git clone https://github.com/nbprashanth/Upstream-Tracker-Rails.git

Once the files are cloned, cd into the folder and execute the following commands.

bundle install
rake db:migrate

rails generate devise:install
rails generate devise user

rails s

The bundle install command installs all the missing gemfiles that are required for the app to work properly. The db:migrate command is used to set up the databases for the app to use. The next two generate commands are used to set up devise, a gem used for user authentication.Finally, the rails s command is use to start the webrick server.

You should now be able to see the web frontend at http://localhost:3000/.

Manipulating records

Viewing records

To see existing records, click on the 'Records' link on the top of the webpage. This displays a page showing all the existing records along with links to show more information about the record, edit the record as well as to delete the record.

Records that have been successfully processed are highlighted green while records that have been processed, but have errors are highlighted in red. Records that are yet to be processed are not highlighted.

NOTE: Authentication is required to delete records.

Adding records

To add new records, click on the 'Records' link on the top of the webpage. Once the page loads, click on the 'New Record' link at the bottom of the page. This opens up the new record page where the details can be filled. Details about each field are given below:

- Package Name : This is the name of the package (not the project, though project name and package name tend to be the same mostly).

Ex : For Coherence (http://coherence.beebits.net), the packages can be found at http://coherence.beebits.net/download/. Here, package name is coherence, if you want to track the coherence packages. If you want to track the UPnP-Inspector package, then that should be entered on to the package name field. In general, the filename preceeding the version string is considered as the package name.

- Branch : This field is used to specify the branch of the software to track. To track the latest version/branch of the software, just enter 'latest' (without quotes).

Ex : For tracking the 0.5.x branch of Coherence package, enter 0.5 in the branch field. The branch field accepts multiple inputs separated by a comma. Thus, to track multiple branches, one can enter all the branches separated by comma.
Ex : Entering - 0.5, 0.6 – would track both 0.5.x and 0.6.x branches of the package, coherence.

- Method : This field defines the technique that is used to check for the upstream version. Each method is explained below:
  - HTTPLS : This is used for HTTP page listings where the download files are all listed in a single HTML page.
  - FTPLS : This is used for FTP page listings where the download files are all listed in a FTP page.
  - DualHTTPLS : This is used when a package has two separate download URLs (HTTP). This is usually applicable for projects where separate release and snapshots directories are maintained.
  - SubdirHTTPLS : This is used when the download files are within subdirectories organized on the basis of their version numbers or branches.
  - SVNLS : This is used for SVN web listing when there is a svn server listening on port 80 listing all tarballs.
  - SF : This is used for projects that are hosted on sourceforge.
  - Google : This is used for projects that are hosted on Google Code.
  - LP : This is used for projects that are hosted on launchpad.
  - Trac : This is used for projects where the upstream tarballs are hosted on trac.
  - Custom : This is used when the project does not fall into any of the above categories, or has special requirements not met above or while using DEHS watch files.
- URL : This field accepts the URL that has to be scanned. The URL entered depends on the method selected. For methods such as HTTPLS, FTPLS, SubdirHTTPLS, SVNLS and Trac the download URLs are entered. For methods such as SF, Google and LP, the project names used on the hosting platform is used. If Custom method is used, the URL is specified using REGEX.

Examples :

- For HTTPLS : http://coherence.beebits.net/download/
- For LP : dee (To track project dee hosted at Launchpad)
- For Custom : http://coherence.beebits.net/download/Coherence-(.*)\.tar\.gz

Once all the fields are entered, click on the create record button to create the record.

Adding a record using DEHS watch file

To add a new record using an exisitng DEHS watch file, click on the browse button and locate the watch file. Click on the Extract URL Button to extract the watch URL. This

automatically sets the method to Custom as the watch URL uses REGEX. Enter the package name and click on the create record button to create the record.

Editing an existing record

To edit an existing record, click on the Records link on the top of the webpage. Locate the record to edit and click on the edit link found on the right hand side. Update the fields and click on the update record button to save changes.

Deleting a record

To delete an existing record, click on the Records link on the top of the webpage. Locate the record to delete and click on the destroy button on the right hand side.

NOTE : Authentication is required to delete records.

To revert a record to it's previous state

For each record, unless deleted, each update is recorded and a history of changes is maintained. This is used to revert a record to it's previous state in case of an error. To revert any record to any of it's previous states, open the Records page by clicking on the Records link on the top of the webpage. Locate the record to revert and click on the show button. In the version history section, the previous versions of the records are shown. If the record was processed, error information of historical records are shown as well. To revert, click on the revert link beside any of the previous versions of the records.

NOTE : Authentication is required to revert records.

Searching for records

The current implementation of the upstream tracker contains a search bar on the top right corner, that can be used to search for packages. The search directly displays the record details in case of a single matching result but shows a disambiguation page when more than one results are returned per query. The results are shown in a table along with their URL and method used. The show button can be used to view the record details.

Using the API to access records

Data pertaining to records on the rails frontend can be accessed through it's REST API. Currently, the system supports two formats : JSON and XML. In the following sections, I shall demonstrate how to test/access data using the REST API calls.

To view records

Records can be viewed by accessing the records url (http://localhost:3000/records). To get response using the JSON format, one can open the URL http://localhost:3000/records.json. For XML response, http://localhost:3000/records.xml can be used. This, however, returns all the records that are available in the database.

To view a particular record

Specific records can be accessed by appending the record ID to the records URL. For example, to view record with ID 1, one can use the URL http://localhost:3000/records/1.json or http://localhost:3000/records/1.xml.

Fields returned in an API call

Whenever an API call is made, JSON or XML data is returned. Both these formats reflet the same data. In this section, I will explain the field names that are used in both these formats so that the information obtained can be further processed.

A sample record in XML format is as shown below :

```
<record>
  <branch>latest</branch>
  <created-at type="datetime">2012-08-18T14:58:24Z</created-at>
  <error type="boolean">false</error>
  <errorMessage/>
  <id type="integer">1</id>
  <info>http://coherence.beebits.net/download/</info>
  <latest-ver nil="true"/>
  <loc/>
  <method>httpls</method>
  <pkgName>coherence</pkgName>
  <processed type="boolean">false</processed>
  <updated-at type="datetime">2012-08-18T14:58:24Z</updated-at>
  <verctrl nil="true"/>
  <verctrlInfo nil="true"/>
</record>
```

Let's look at the tags shown above :

- Branch : Contains the branch for which the record was processed.
- Created-at : Contains the timestamp at which the record was created.
- Error : Contains a boolean value. Depicts if the record was processed successfully or not.
- ErrorMessage : Null if error is false. If error is true, additional error information is provided here.
- Id : This is the record ID used to uniquely identify any record.
- Info : This tag contains the URL information. The upstream data is obtained by processing this URL.
- Latest-ver : Contains the latest version of the package. Is null if not processed.
- Loc : Contains the location or the URL of the tarball pertaining to the latest version of the package.
- Method : Contains the method used to process the record.
- PkgName : Contains the name of the package.
- Processed : Contains a boolean value which shows if the record has been processed or not.
- Updated-at : Contains timestamp of last update.
- Vercntrl and vercntrlinfo are not used.

Using curl to fetch records

Curl can be used to test and play around with the API. How to use curl with RESTFul API can be found at http://blogs.plexibus.com/2009/01/15/rest-esting-with-curl/.

What's missing?

The Upstream Tracker was developed as a part of Google Summer of Code 2012 during a three month interval. Hence, there is a lot of scope for the project to be improved. Some of the current features that are lacking which can be worked upon for the future versions are :

1. Improved UI
2. In case of multiple branches, the output is in CSV format. This can be updated to render proper XML with multiple branch tags instead of a single branch tag with CSV content.
3. Comparison tables with particular distros.