Summer Student Program: Project Report

Puppet Module Sharing System

José Molina Colmenero

ABSTRACT

1. Introduction

As a Summer Student at CERN I worked in the IT department, PES section, PS group under the supervision of Steve Matthew Traylen. My occupation as a software developer consisted in developing a sharing system for existing CERN's Puppet modules, so that CERN IT members can clearly visualize which modules and versions are available and their corresponding documentation.

The structure of the balance of this report is as follows. In the second section I give a brief introduction about Puppet and its modules, i.e. the fundamentals of my project. In the third section I define the goals my project has to achieve. In the forth section I explain in detail how my project was built, as well as the initial approach. In the fifth section I present the outcome of the project. Finally, in the sixth section, I provide an overview of the experience gained throughout the Summer Student Program.

2. Puppet & Modules

Puppet is an open source tool design to manage Unix-like system configuration in a declarative way. The user has, in a nutshell, to describe all system resources he needs using specific Puppet declarative language. This information is stored in the so-called Puppet manifests, which are compiled and executed in the final devices.

Nowadays CERN has more than ten thousand compute nodes, so in order to maintain such a big infrastructure it requires using an automated configuration management system like Puppet that provides development, support and maintenance of any machine hosted in the CERN Computer Centre.

At the same time CERN is using an Open Source tool, it is contributing back to the community all work done with Puppet modules which are used to configure different aspects of its services, and to achieve this CERN is using public GitHub repositories inside the CernOps group.

A Puppet module is basically reusable and sharable self-contained bundle of code and data. This way a user can reuse different modules, including its functionality and making life easier for system administrators.

3. Project Goals

In this section I present the goals of the project and the main tasks associated to the three main scopes, which are:

1. Extracting all CERN modules' information.
2. Visualising in a graphic web interface this information.
3. Build a Puppet module in order to automatically run a server instance of the library.

The first part represents the core component of the project. It implements the main functionality of parsing each module's data, including its documentation, and sending this information back to the client as a response. The main tasks to achieve this desired behavior are:

- Provide support for many types of module formats. It includes compressed modules and directly the source code.
- Parse each module metadata in two possible formats: Modulefile and metadata.json.
- Parse the documentation (README files), written in markdown language.
- Sending all this information to the client in a json format.

The second part shows a user graphic interface (GUI) in the web. Its main functionalities are the following:

- Display alphabetically all available Puppet modules, showing its name, author and versions.
- Display an in-detail view of each module, including name, author, versions, description and documentation.

The third part consists in creating a Puppet module to load the server and all its requirements. It has to:

- Install all server requirements in a Scientific Linux system and set the corresponding security parameters.
- Properly configure the server.
- Run indefinitely the server.

4. Puppet Library Project

In this section I focus on the concrete implementation I have done, using a Ruby open source library called Puppet-library and adding some changes to make it more adaptable to the project.

4.1. Original situation

Originally the library provides enough functionality to show some information about packed puppet modules, not including documentation. It offers also some extra functionalities like proxying modules from a remote server or specifying the source code directory of only one module.

However, it does not completely meet al requirements needed at CERN because its puppet modules are stored directly with its source code and they are constantly under development depending on the situation, so using a packed format would be pointless. Also, a basic requirement is to show each module's documentation basically because only knowing the name and the author you do not really know which functionality a concrete module provides, and in a on-working process it would be tedious to navigate

inside its directory to check (probably in a non-friendly environment like a system terminal) its documentation.

4.2. Improvements

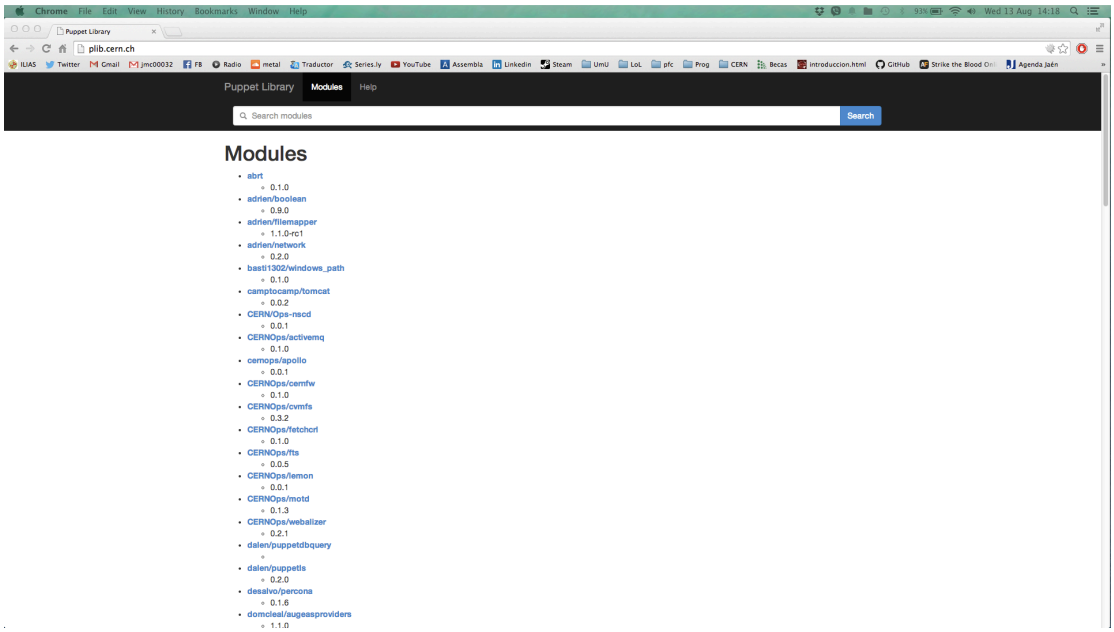
Given the initial situation my work consisted in adding enough functionality to the library to be usable by CERN's purposes. At the same time the project must be open source and contribute actively to the original project. The main improvements I have to develop have been:

- To add documentation visualization to each module.
- To allow Puppet module information retrieving from a source code's root folder of a set of modules.
- To build a Puppet module specifically to set up and run the Puppet Module Service.

5. Project Results

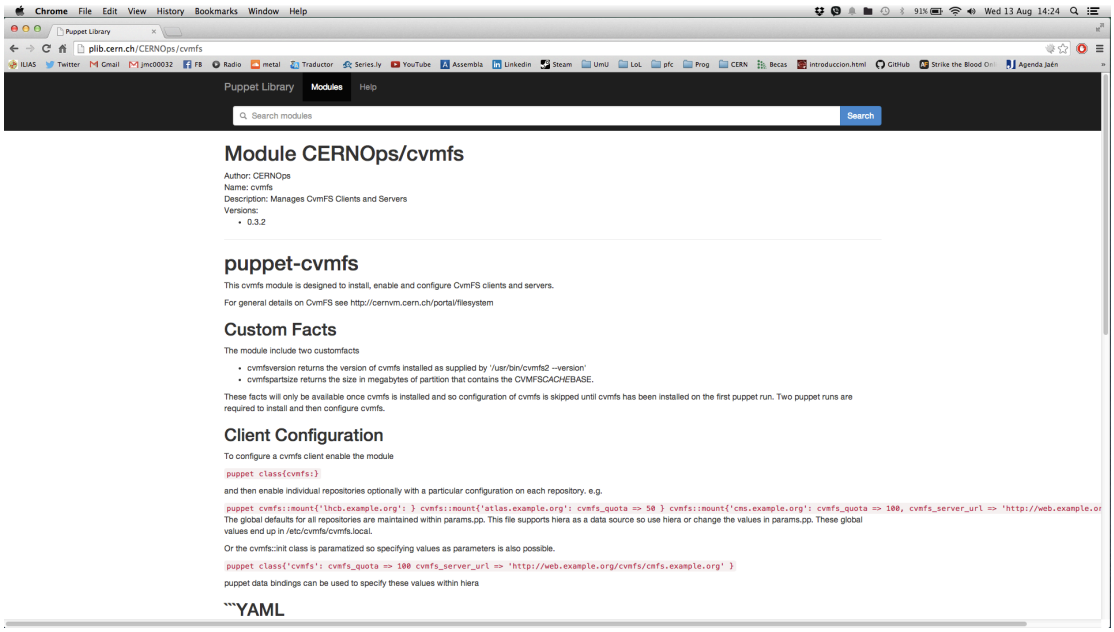
In this section I provide the final outcome of the project presenting the Graphic User Interface:

5.1. Retrieving CERN modules from their source code



As you can appreciate, all CERN's modules appear in the "author/name/versions" format so that the user can easily check which modules are available. At the same time I have alphabetically sorted the modules to provide a faster search.

5.2. Visualizing a concrete module



As you can see, the server has retrieved all basic module data, including its documentation.

5.3. Creating a Puppet module to run the server

In order to check the module implementation I have created a testing environment in it-puppet-hostgroup-puch. Final virtual machine with the server can be accessed in plib.cern.ch.

6. Conclusions

During my stay at CERN, working on this project, I not only improved a lot my knowledge about Ruby language and Ruby on Rails framework, but I also gained experience contributing to open source projects. I learned as well which difficulties I can expect from these situations and how to approach them.

On top of that, I used distributed revision control systems like Git, which will be extremely beneficial for my future jobs, and GitHub platform, where I learned how to contribute to projects by creating pull requests.