Simple OpenStack Monitoring Tool

Software Requirements Specification

Version 1.4

Team Name: Oceans11.

Team Members:

- Tarun Aluguri.
- Nikhil Reddy Araga.
- J N S Sri Harsha Vardhan Kamisetty.
- Prathisrihas Reddy Konduru.
- Sai Anoop Nadella.
- Rohit Pothuraju.
- Dilip Renkila.
- Venkat Sivendra Tipirisetty.
- Sai Bhargava Ravi Teja Vedantam.
- S. Sai Srinivas Jayapala Vemula.
- Rahul Vudutha.

1) PREFACE:

The main concern of the project is to develop a simple and intuitive web based drill down GUI that provides the overview of an OpenStack environment. The tool monitors the OpenStack services Nova, Neutron, Cinder, Keystone, Glance and Heat existing on the node. This is the revised version of the document (version 1.4).

Release Version 1.4 on 2015-28-05:

- Made changes to the diagrams of all three modules. Included RESTful API in the description. See section 3.1
- Made changes regarding graphs produced by the tool. See section 3.1,3.2,3.3

Release Version 1.3 on 2015-20-05:

- Made changes regarding the backend functionality. (See Sections 3.3)
- Made changes regarding description of RESTful API used in the web interface. (See sections 4.1) and gave tests for these functionalities.
- Changed the web server requirement module. (See section 4.2)

Release Version 1.2 on 2015-14-05:

- Made changes regarding the backend module of the tool with emphasis on the service status data retrieved by the script (See section 3.3).
- Added additional login authentication and RESTful API requirements (See section 4.1) and changed Req_USRF4
- Made changes in Section 4.2 in Req_SYSF1, Req_SYSF3, Req_SYSNF3,

Release Version 1.1 on 2015-04-05:

- Made changes regarding the performance User's functional requirements and System's functional requirements (See Sections 4.1 and 4.2 i.e., User's functional requirements and System's functional requirements respectively).

Release Version 1.0 on 2015-04-27:

- Initial Release

In the remainder of the document, Section II describes briefly about the basic abbreviations used in this document. Section III defines the System Architecture which is divided into Frontend, Databases and Backend modules. Lastly Section IV gives the Requirements (User and System).

2) GLOSSARY AND ABBREVIATIONS:

1) **API: Application Programming Interface:**

   An **API** is a set of routines, protocols, and tools for building software applications.

2) **GUI : Graphical User Interface**

   A **GUI** is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces.

3) **PHP: Hypertext Pre-processor**

   PHP is an Open Source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

4) **PERL: Practical Extraction and Report Language**

   Perl is a general-purpose programming language developed for text manipulation and also used for tasks including system administration, web development, network programming, GUI development, and etc.

5) REST: Representational State Transfer

   REST is a software architecture style consisting of guidelines and best practices for creating scalable web services. It is a coordinated set of constraints applied to the design of components in a distributed hypermedia system that can lead to a more performant and maintainable architecture.

3) System Architecture

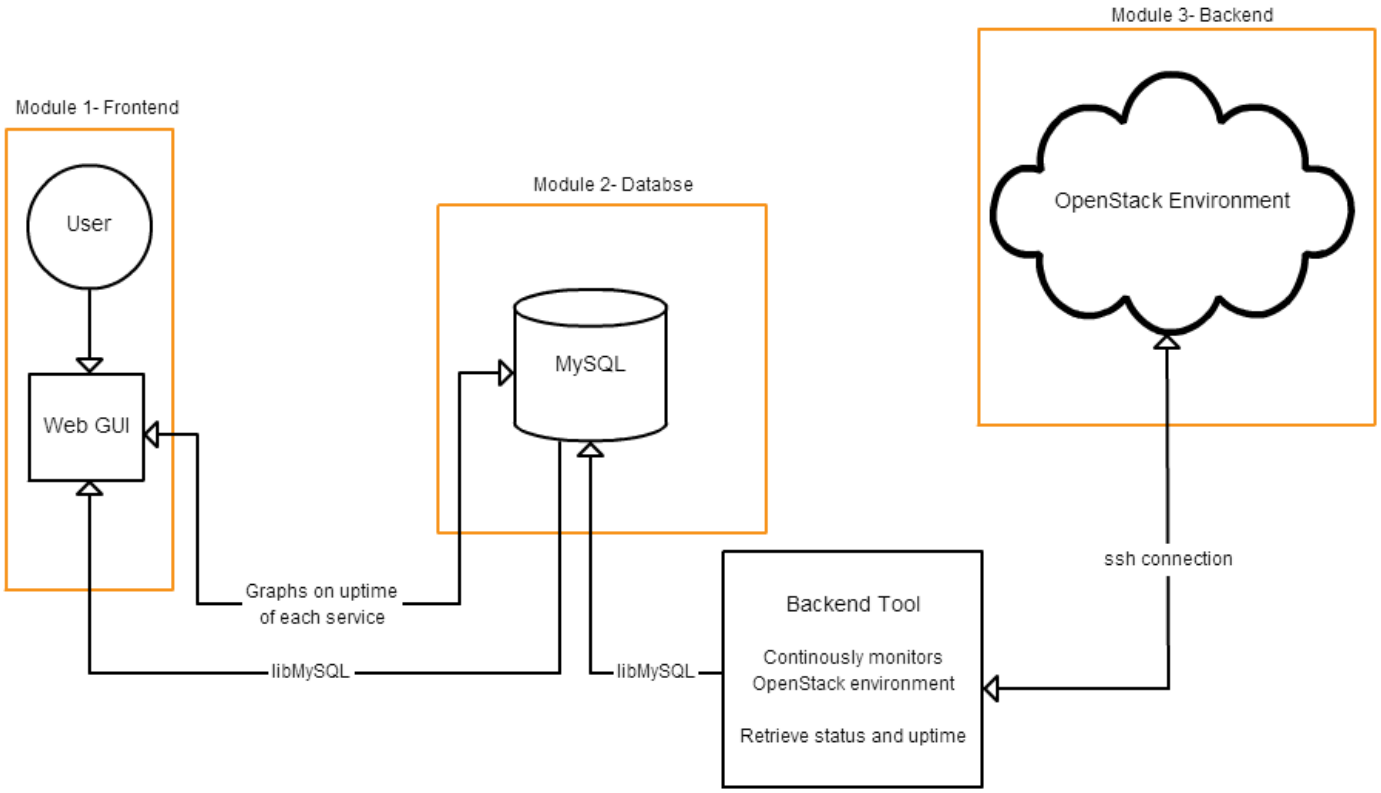

Fig 1: System Architecture

This section provides a high level view of the system architecture and is divided into three subsections. Each subsection describes a major component of the system being developed.

The three main modules in the system are Frontend, Database and Backend.

The Frontend web interface is made RESTful. RESTful API is used to export data to a 3rd party.

Graph images are generated from the data stored in MySQL database which are later uploaded by the Frontend.

3.1 FRONTEND:

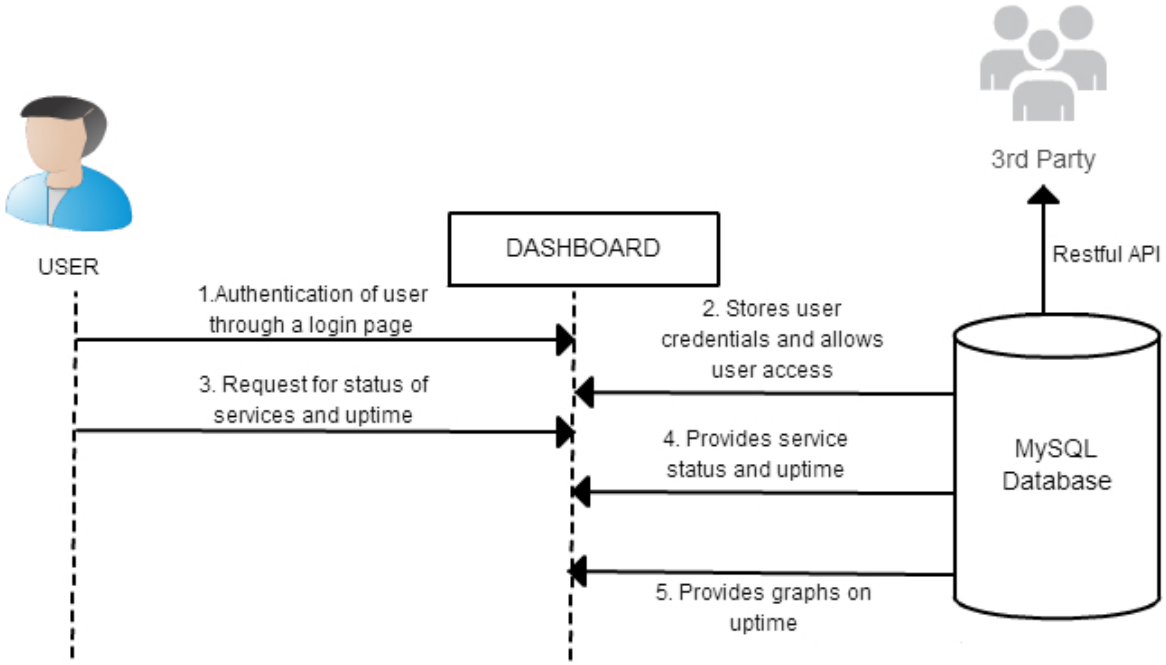

Fig 2: Frontend

This module represents frontend of the system architecture. The customer can check status of OpenStack services via web interface. The customer is given access to the dashboard after authentication through a login webpage.

After entering the dashboard, the customer can see status of currently monitored services. Customer can RESTART services semi-automatically. The number of restarts are also displayed in the restart page. The status and uptime are updated automatically after a restart. Graphs are generated by Perl script using data already inserted to MySQL database by the backend. These images are displayed on the web page.

The web interface is made to be RESTful. An index page is created such that when a user performs a GET request for data related to a particular service. It is redirected to another data page that displays status and uptime retrieved from the database, in JSON format.

+**Requires**: The RESTART feature requires access into the Dashboard, which is a platform for the tool that is given access after authentication of user.

3.2 DATABASES:

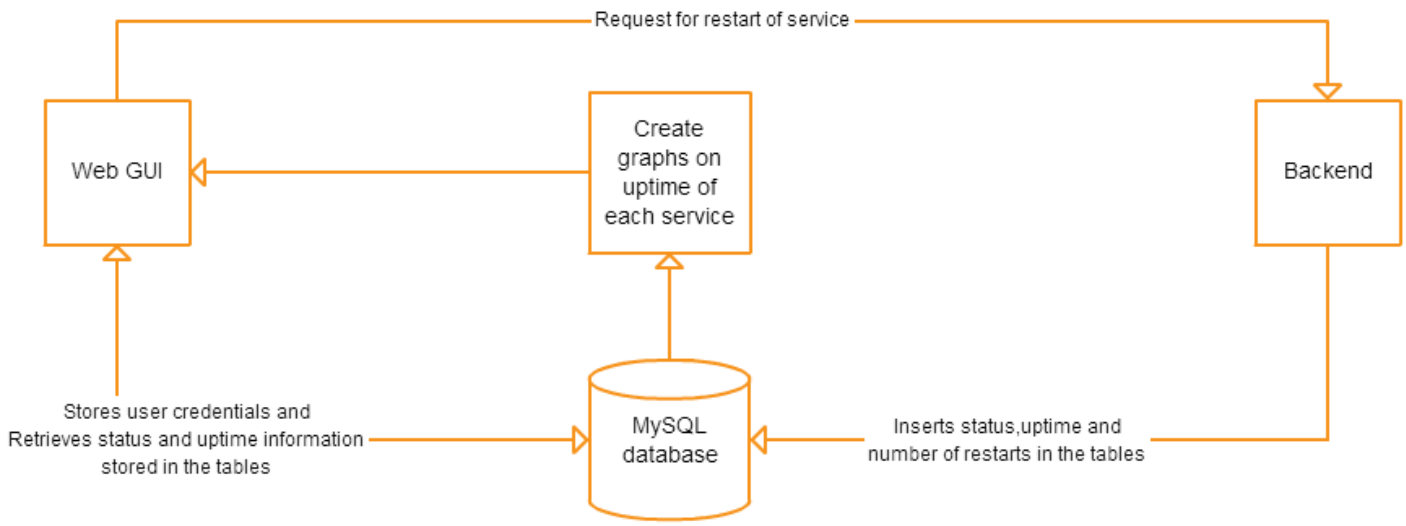

Fig 3: Databases.

This module represents the database management tools used in this product.

The MySQL database contains user authentication data, status and uptime information inserted into their respective service tables.

Different tables are created for each service and sub-services are inserted into each table. For example, in nova table, sub-services nova-api, nova-conductor, nova-cert, nova-scheduler etc. are entered under the service column.

The status and uptime are inserted into each table by the backend script. These values are retrieved by Frontend to be displayed on web interface. The number of restarts are also stored in the tables and displayed in the restart page.

3.3 BACKEND:

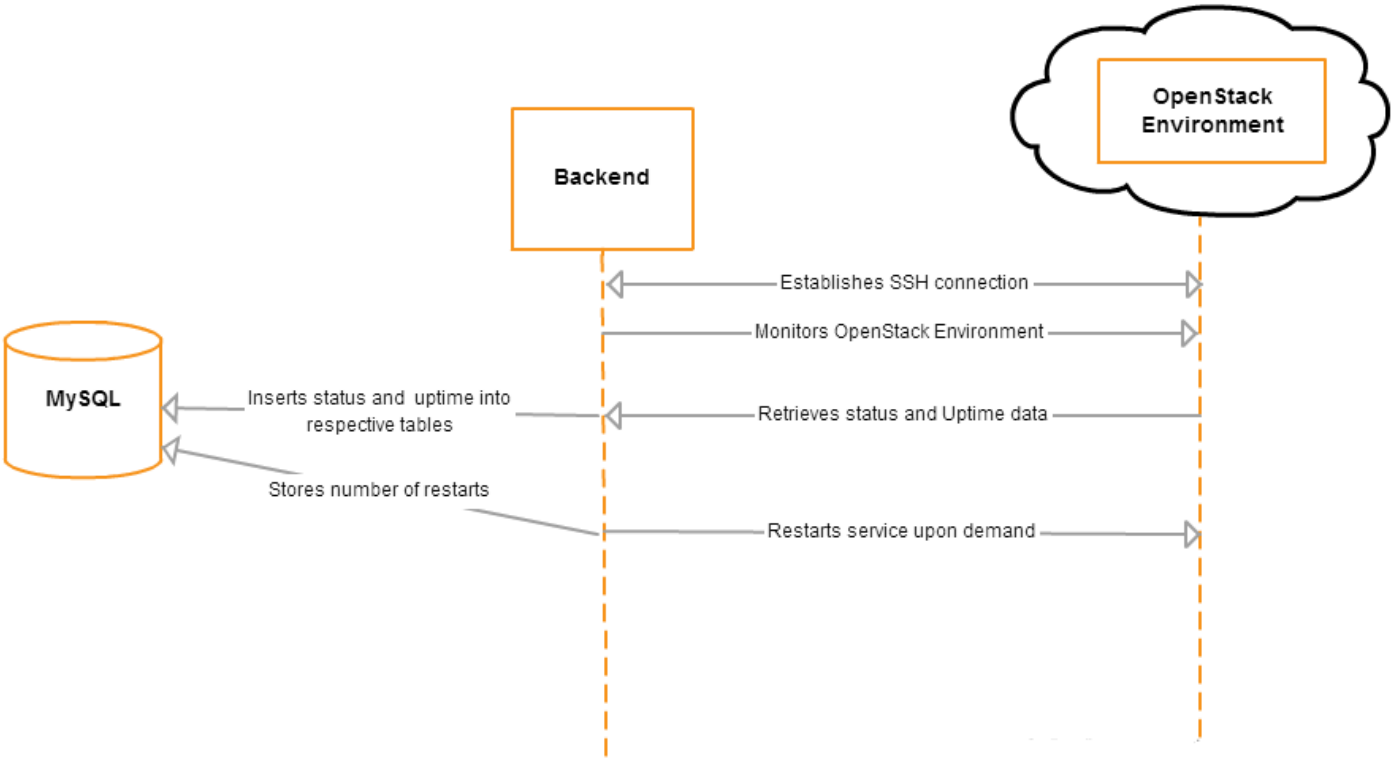

Fig: 4: Backend

This module represents the backend of the system architecture. The backend script remotely connects to the OpenStack environment using SSH connection. This backend Perl script uses variables that contain host IP address, private key and passphrase to establish a secure connection.

The backend retrieves status of each OpenStack service by executing a shell command within the backend script.

It retrieves status and inserts into the MySQL database into tables created with different names for each service.

It also facilitates restart of service upon request by user from frontend. The number of restarts are also measured and stored in the tables.

4) Requirements:

The user must be able to select the service to monitor. The details of services to be monitored will be provided to the user, demanded in accordance with certain fixed regulations. The project requirement defines both user requirements and system requirements.

These are discussed briefly in the list of requirements section.

4.1. User requirements

This section describes the services provided for the user by the product. This section is further divided into functional and non-functional requirements subsections.

S.No	Requirement	Identification S	Creation date	Change date	Module	Type	Dependecies	Test	Description
1	Selection of the service	Req_USRF1	8th May 2015		1. FrontEnd	Functional	Req_SYSF1, Req_SYSF2, Req_SYSF3, Req_SYSNF2	F-T4	The customer requires a simple Web based GUI with easy access to services
2	Service Restart	Req_USRF2	8th May 2015	20th May 2015	3. BackEnd		Req_SYSF1, Req_SYSF3, Req_SYSNF2	B-T3	The customer also requires an option in the web GUI to restart the OpenStack services when needed
3	Notification	Req_USRF3	8th May 2015	20th May 2015	3. FrontEnd		Req_SYSF1, Req_SYSF3, Req_SYSNF2	F-T3	The customer requires notifications during service failures
4	Graphs	Req_USRF4	8th May 2015	20th May 2015	1.Frontend		Req_SYSF1, Req_SYSF2, Req_SYSNF2	F-T4	The customer receives graphs on Uptime. Time scales are not flexible.
5	Documentation	Req_USRNF1	8th May 2015			Non-Functional			Documentation includes user guides/manuals that provide instructions regarding software installation as well as tool operating instructions.
6	Customer Service	Req_USRNF2	8th May 2015						Service Support will be provided to the customer along with the product. It will cover detecting and rectifying the problems related to the functionality of the tool
7	Login Authentication	Req_USRF5	13th May 2015		1.Frontend	Functional	Req_SYSF1, Req_SYSF2, Req_SYSNF2	F-T1	Prevents unauthorised intrusions to the Tool Dashboard. Only the user can access the tool by providing correct credentials.
8	RESTful API	Req_USRF6	13th May 2015		1.Frontend		Req_SYSF1, Req_SYSNF2	F-T2	RESTful API is used to export data to a 3rd party. Status and Uptime of a service can be displayed in web browser on GET request.
9	Status and Uptime of services	Req_USRF7	28th May 2015		3.Backend		Req_SYSF2 Req_SYSF3 Req_SYSNF2	B-T2	Status and Uptime of OpenStack services are inserted into the MySQL database by the backend continously.

Table 1: User Requirements

4.2. System requirements:

This section provides details about the system requirements. These are technical requirements that complement the user requirements and provide information for design and implementation of product. In the same manner as in user requirements, this section is also divided into subsections of functional and non-functional requirements.

S.No	Requirement	Identification S	Creation date	Change date	Module	Type	Dependecies	Test	Description
1	Web Server	Req_SYSF1	8th May 2015		1. FrontEnd	Functional			Web Server is used to render Web Pages requested by client computers. Clients typically request and view Web Pages using a Web Browser such as Firefox. The web server that is required is Apache2 local server
2	Database	Req_SYSF2	8th May 2015	20th May 2015	2. Database			D-T1 D-T2	MySQL: For holding user credentials, status and uptime of OpenStack service, a database is used. Different tables are created for each service
3	OpenStack Environment	Req_SYSF3	8th May 2015		3. BackEnd		Req_SYSNF1	B-T1	OpenStack environment includes nodes having OpenStack services like Nova, Neutron, Glance, etc
4	Operating System	Req_SYSNF1	8th May 2015			Non-Functional			The required operating system as part of the system requirements is Ubuntu 14.04 LTS. The minimum RAM for the operating system is 2GB.
5	Scripting Languages (HTML, CSS, PHP, PERL)	Req_SYSNF2	8th May 2015						Scripting Languages for building web based GUI - HTML, CSS, PHP Scripting Language for building core tool - PERL
6	Compatibility	Req_SYSNF3	8th May 2015						The developed product must be compatible with previous and ongoing OpenStack versions.
7	Testability	Req_SYSNF4	8th May 2015						The developed product must support software testing to high extent in order to detect bugs with ease.

Table 2: System Requirements

5) References:

[1]. http://docs.openstack.org/: OpenStack Documentation

[2]. Ian Sommerville. *Software Engineering*. 9th ed.