Simple OpenStack Monitoring Tool

Design Document

Version 1.3

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 an overview of an OpenStack environment. The tool monitors the OpenStack services Nova, Neutron, Cinder, Keystone, Glance and Heat existing on the nodes. This is second version of the document (version 1.2).

In the remainder of the document, Section II describes briefly about the basic abbreviations used in this document. Section III describes Module 1: Frontend which is divided into Detail design and Unit Test plan. Section IV describes Module 2: database Management which is divided into Detail design and Unit Test plan. Section V describes Module 3: Backend which is divided into Detail design and Unit Test plan. Lastly Section VI shows list of References used in documentation.

Release Version 1.2 on 2015-20-05

- Made changes in D-T2 and D-T3. See sections 4.2
- Made changes in backend. See section 5

Release Version 1.1 on 2015-14-05

- Made changes in Frontend module. Removed notifications by popups. Modified the diagrams in all three modules showing interaction between databases and interface. Provided description of RESTful API used by the product. (See section 3.1).
- Modified the frontends tests. See section 3.2. Modified database module details. See section 4.2. Modified detailed design of backend. See section 5.1. Changed tests in all three modules. See sections 3.2, 4.2, 5.2

Release Version 1.0 on 2015-05-05

- This is the initial version of the document

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 Preprocessor

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.

3) MODULE 1: FRONTEND

This module describes the web interface of the tool. First, the user is authenticated with a login page giving way to the main page containing the Dashboard.

3.1) Detailed Design:

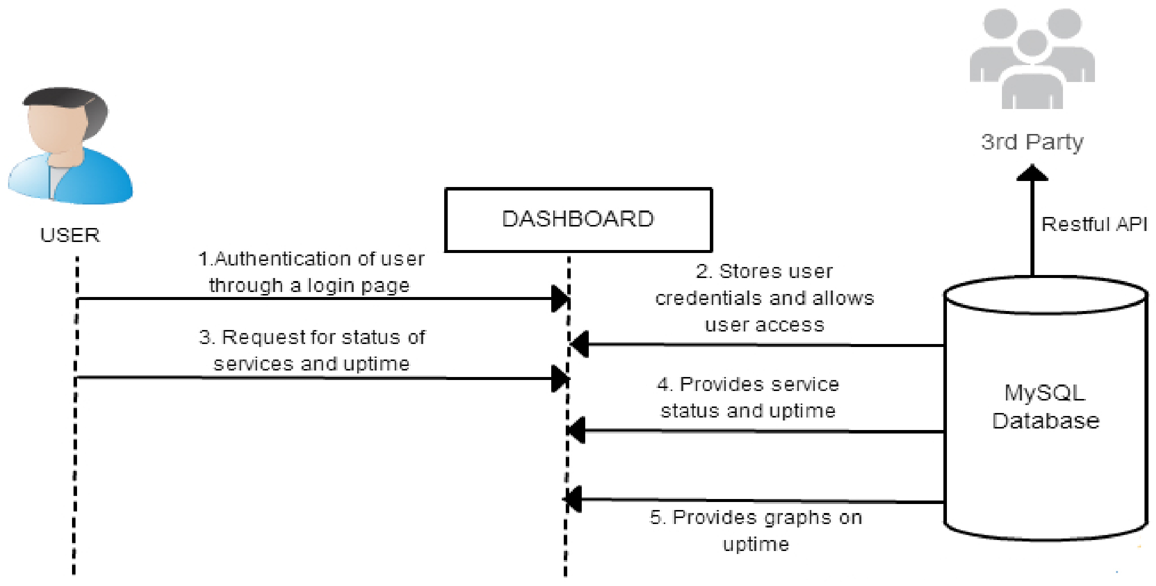
3rd Party
USER
DASHBOARD
Restful API
1.Authentication of user through a login page
2. Stores user credentials and allows user access
3. Request for status of services and uptime
4. Provides service status and uptime
MySQL Database
5. Provides graphs on uptime

Fig.1: GUI

The Dashboard contains list of OpenStack services running on the nodes. The status along with Uptime of monitored services is retrieved using shell commands executed from backend Perl script. It connects to server via SSH connection. The user authentication credentials are stored in MySQL database.

Graphs are plotted for Uptime of each service. Perl script generates these graphs using data from MySQL database.

The notification of the service failure on a node being monitored is shown in red color font with error message. The tool provides user an option to semi-automatically restart the failed services. The number of restarts are stored in MySQL tables.

A RESTful interface of web service is designed with PHP script. It facilitates export of data to a 3rd party through HTTP GET request and retrieves data from MySQL database shows service name, status and uptime in JSON format. A PHP web page is created such that a 3rd party can give a GET request for data related to a particular service, the page accesses the database and retrieves status and uptime of the services and encodes this data in JSON which is displayed on the web page.

3.2) Unit Test Plan:

F-T1: Authentication

Test: Login page test

Purpose: To prevent unauthorized users from gaining access to the dashboard

Requirement: Req_SYSF1, Req_SYSF2, Req_SYSNF2

Environment: Browser for rendering the webpages, PHP5

- Preinstalled php5
    - Open terminal
    - Run command: sudo apt-get install php5
- Preinstalled apache server
    - Run command: sudo apt-get install apache2
- Preinstalled MySQL database
    - Run command: sudo apt-get install mysql-server
    - Database must be created with a table containing user credentials.

Operation:

- Open web browser.
- Go to login.php from localhost and enter user credentials.
- If correct credentials are entered, it will redirect to Dashboard
- If incorrect credentials are entered, alert will be shown and access will be denied

Expected Result: Displays Tool Dashboard upon entering correct username and password

Result:

Comment: Before rendering the webpage apache2 status must be checked: "service apache2 status" on the terminal.

F-T2: Export 3rd party data

Test: REST API test

Purpose: To export data to a 3rd party using RESTful API

Requirement: Req_SYSF1, Req_SYSF2, Req_SYSNF2

Environment: Web browser, apache server

- Preinstalled php5
  - ➢ Open terminal
  - ➢ Run command: sudo apt-get install php5
- HTTP communication environment
- Preinstalled apache server
  - ➢ Run command: sudo apt-get install apache2

Operation:

- Open web browser
- In the web address bar, enter the GET request for data regarding a particular service by typing <SERVER IP>/frontend/Oceans11_rest_api.php/?service=<name_of_service>

Expected Result: Status and uptime data, of the service requested, are displayed in JSON format.

Result:

Comment: Care must be taken not to overload the network that may slow down operation.

F-T3: Receive update on status and uptime of service

Test: Notification test

Purpose: Check the automatic update of service status and service uptime in dashboard

Requirement: Req_SYSF1, Req_SYSF2**,** Req_SYSNF2

Environment: Web browser, apache server

- Preinstalled php5
  - ➢ Open terminal
  - ➢ Run command: sudo apt-get install php5
- Preinstalled apache server
  - ➢ Run command: sudo apt-get install apache2
- Preinstalled MySQL database
  - ➢ Run command: sudo apt-get install mysql-server
  - ➢ Database must be created with tables containing fields for status and uptime for each service
  - ➢ Each service has an independent table containing the list of sub-services

Operation:

- Open web browser
- Enter the dashboard through the login page
- On the home page, to the right, a service status panel displays list of services being monitored and current status for each of them (Running/ERROR)
- Whenever a service has stopped, the status is displayed in red color
- On the statistics page, the uptime of each service is displayed. It is also updated whenever a service is restarted

Expected Result: Status changes from Running to ERROR when a service stops. When a service is restarted, the uptime is updated and reset.

Result:

F-T4: Plot graphs on uptime of each service

Test: Graph test

Purpose: Check generation of graphs by Perl script using data in MySQL database

Requirement: Req_SYSF1, Req_SYSF2**,** Req_SYSNF2

Environment: Web browser, apache server

- Preinstalled php5
  - ➢ Open terminal
  - ➢ Run command: sudo apt-get install php5

- Preinstalled apache server
  - Run command: sudo apt-get install apache2
- Preinstalled MySQL database
  - Run command: sudo apt-get install mysql-server
  - Database must be created with tables containing fields for status and uptime for each service
  - Each service has an independent table containing the list of sub-services
- Preinstalled GD::Graph perl module
  - Open terminal
  - Run command: sudo apt-get install libgd-graph-perl

Operation: Creates graphs on Uptime of each service and passes it to frontend

- Open web browser
- Enter the tool dashboard
- In the Statistics tab, list of services are displayed with their respective uptimes.
- Select a particular service, the Uptime graph for that service is displayed

Expected Result: Uptime graph is plotted for each service based on data stored in MySQL database

Result:

4) MODULE 2: DATABASE MANAGEMENT

This module comprises of the database management portion of the product, it also includes how the database interact with other modules in the product.

The project uses **MySQL** database. This database interacts with the other two modules namely **Frontend** and **Backend.**

4.1) Detailed Design:

This module provides a vivid explanation of the database management in the project. The database used in the tool is a MySQL database. MySQL database contains user credentials along with RESTART operation requests. Different tables are created for each service and sub-services are inserted into each table. The number of restart operations for each service are stored in a column in restart table. MySQL database is also used to store the service status, uptime and graphs for uptime are drawn through a Perl script and shown in the web GUI.

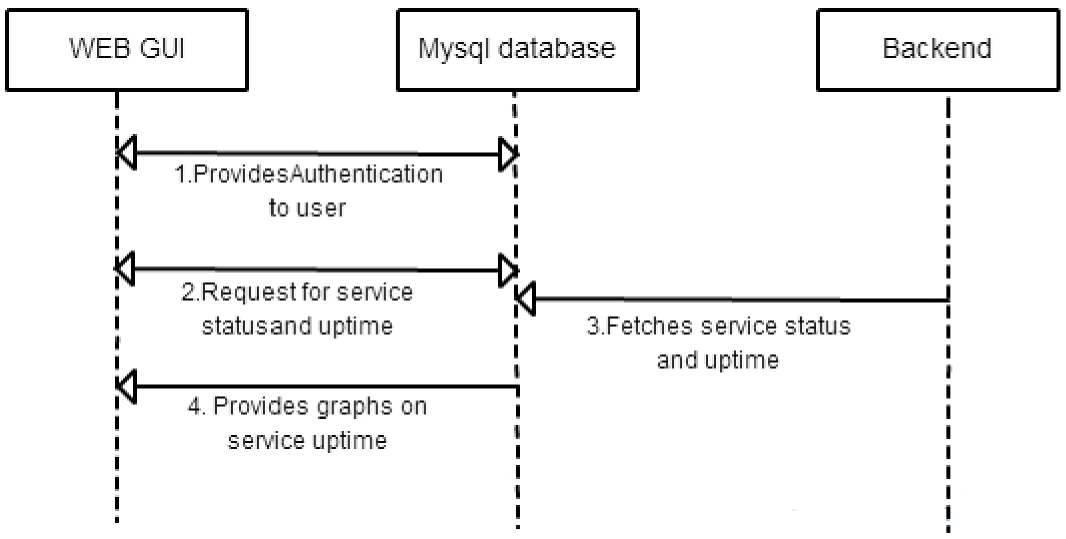
WEB GUI
Mysql database
Backend
1.ProvidesAuthentication to user
2.Request for service statusand uptime
3.Fetches service status and uptime
4. Provides graphs on service uptime

Fig.2: Database Management

PHP- MySQLi

Is a Relational Database Driver that provides an interface between MySQL database and PHP. It includes functions to establish connection to the database as well as functions for manipulating database (create, delete, update, etc.).

Perl-GD::Graph

GD::Graph is a *perl5* module to create charts using the GD module. The following classes for graphs with axes are defined:

GD::Graph::bars

The GD.Graph.bars plugin provides an interface to the GD::Graph::bars class defined by the GD::Graph module. It allows one or more (x,y) data sets to be plotted with each point represented by a bar, in addition to axes and legends.

GD::Graph::Data

This module encapsulates the data structure that is needed for GD::Graph . An object of this class contains a list of X values, and a number of lists of corresponding Y values. This only really makes sense if the Y values are numerical, but you can basically store anything. Undefined values have a special meaning to GD::Graph, so they are treated with care when stored.

4.2) Unit Test Plan:

D-T1: MySQL database contains user credentials

Module: PHP::MySQL

Purpose: For interaction between PHP script and MySQL

Requirement: Req_SYSF1, Req_SYSF2, Req_USRF1, Req_USRF2

Environment: MySQL database contains a table in the database for user credentials

- Preinstalled php5
    - ➢ Open terminal
    - ➢ Run command: sudo apt-get install php5
- Preinstalled MySQL database
    - ➢ Run command: sudo apt-get install mysql-server
    - ➢ Database must be created with tables containing fields for status and uptime for each service
    - ➢ Each service has an independent table containing the list of sub-services

Operation:

- Open web browser
- Enter the tool dashboard, click on "Register new user". Enter the requested credentials. A new user is created
- Open mysql in terminal by entering $ mysql –u root -p
- Check 'users' table in the tool's database for the registered user credentials

Expected Result: The table contains user credentials as created from the frontend

Result:

D-T2: MySQL database contains status and Uptime data

Module: DBI, DBD::mysql

Purpose: To ensure that database contains status and uptime

Requirement: Req_SYSF1, Req_SYSF2, Req_USRF1, Req_USRF2

Environment: MySQL database contains a table in the database for user credentials

- Preinstalled perl DBI
- Preinstalled MySQL database
    - ➢ Run command: sudo apt-get install mysql-server
    - ➢ Database must be created with tables containing fields for status and uptime for each service

Each service has an independent table containing the list of sub-services

Operation:

- Run backend script so that the script can monitor the OpenStack environment and insert status and uptime into the corresponding tables.
- Open mysql in terminal by entering $ mysql –u root –p
- Check '<service_name>' table in the tool's database for the status and uptime of each sub-service.

Expected Result: The '<service name>' table contains the status and uptime data.

Result:

5) MODULE 3: BACKEND

5.1) Detailed Design

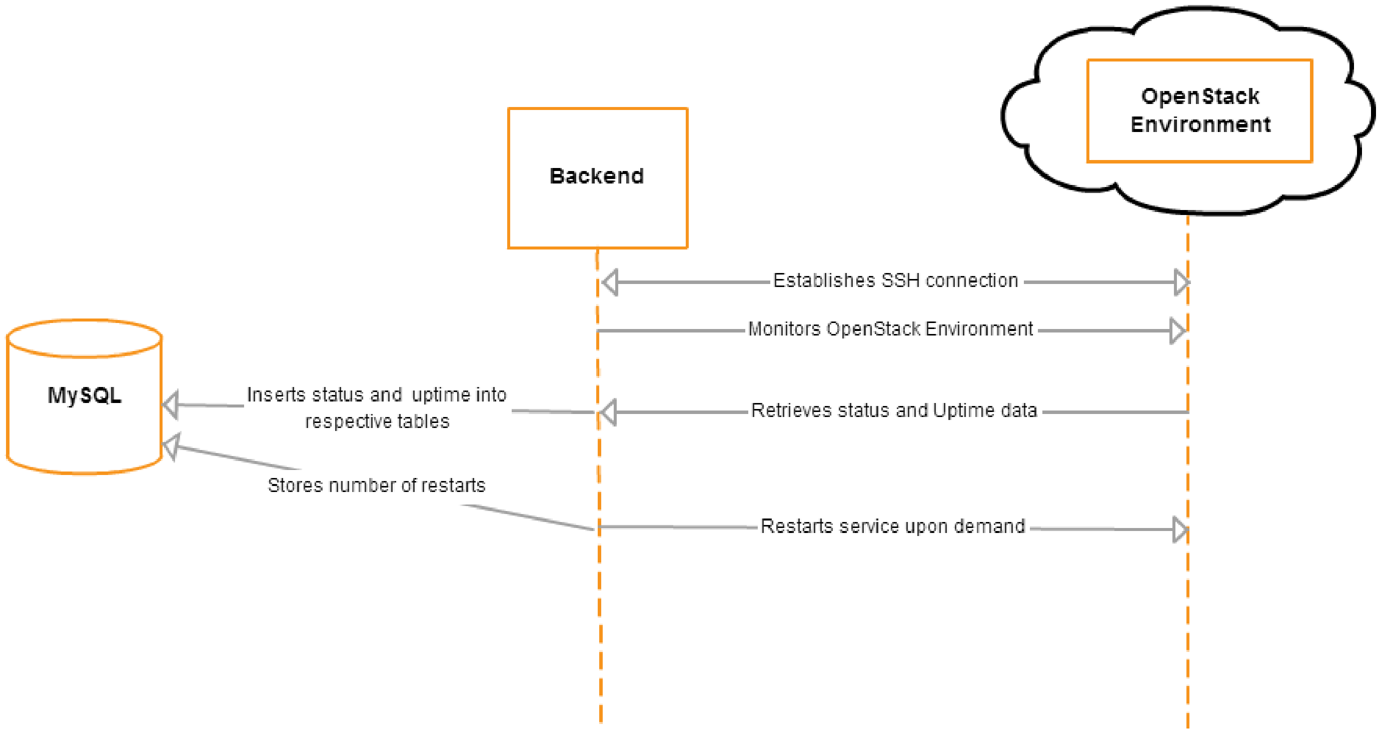
OpenStack Environment
Backend
MySQL
Establishes SSH connection
Monitors OpenStack Environment
Inserts status and uptime into respective tables
Retrieves status and Uptime data
Stores number of restarts
Restarts service upon demand

Fig.3: Backend

The backend interacts with the OpenStack server to carry out operations requested by the user from the frontend. The backend script consists of perl script that contains SSH credentials to establish a secure SSH connection with the server. It retrieves status data of service from the nodes and passes this data to the frontend. It also retrieves data regarding Uptime required to plot graphs with the help of Perl script and show graphs through frontend. The number of restarts are stored in restart table of the database. The number of restarts are incremented by the backend every time a service is restarted.

5.2) Unit Test Plan:

B-T1: Ensure establishment of secure connection between user and server

Module: Net::OpenSSH

Purpose: To execute commands on remote server's terminal

Requirement: Req_USRF3, Req_SYSF3

Environment:

- OpenStack Environment.
- SSH public key to be placed in the authorized_keys directory in /root/.ssh of the host.
- The corresponding public and private key pairs are to be placed

Operation: It is executed from the directory

Expected Result: Successfully establish a secure connection between OpenStack environment and backend

Result:

Comment: To execute RESTART command on terminal of OpenStack nodes

6) REFERENCES:

[1]. http://search.cpan.org/~dtown/Net-SNMP-v6.0.1/lib/Net/SNMP.pm

[2]. http://search.cpan.org/~salva/Net-OpenSSH-0.64/lib/Net/OpenSSH.pm

[3]. http://search.cpan.org/dist/Net-SSH/SSH.pm