Developer Document

Simple OpenStack Monitoring Tool

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) 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.

2) INTRODUCTION:

This document gives an idea for the further extension of the tool development. This document also serves the assistance for developers for adding new functionalities.

3) INTERPRETATION OF TOOL:

The entire backend of the software deals with monitoring of open stack services. The backend is entirely written in Perl for easy compatibility with front end PHP scripts. All the service monitoring is done on the remote system by executing Perl scripts in an SSH terminal. All the data that is retrieved is stored into SQL Database. The front end PHP and HTML scripts retrieves the status from the database and shows on the WEB GUI. Java Script is used to continuously run the PHP scripts to retrieve information from database.

4) SYSTEM ARCHITECTURE:

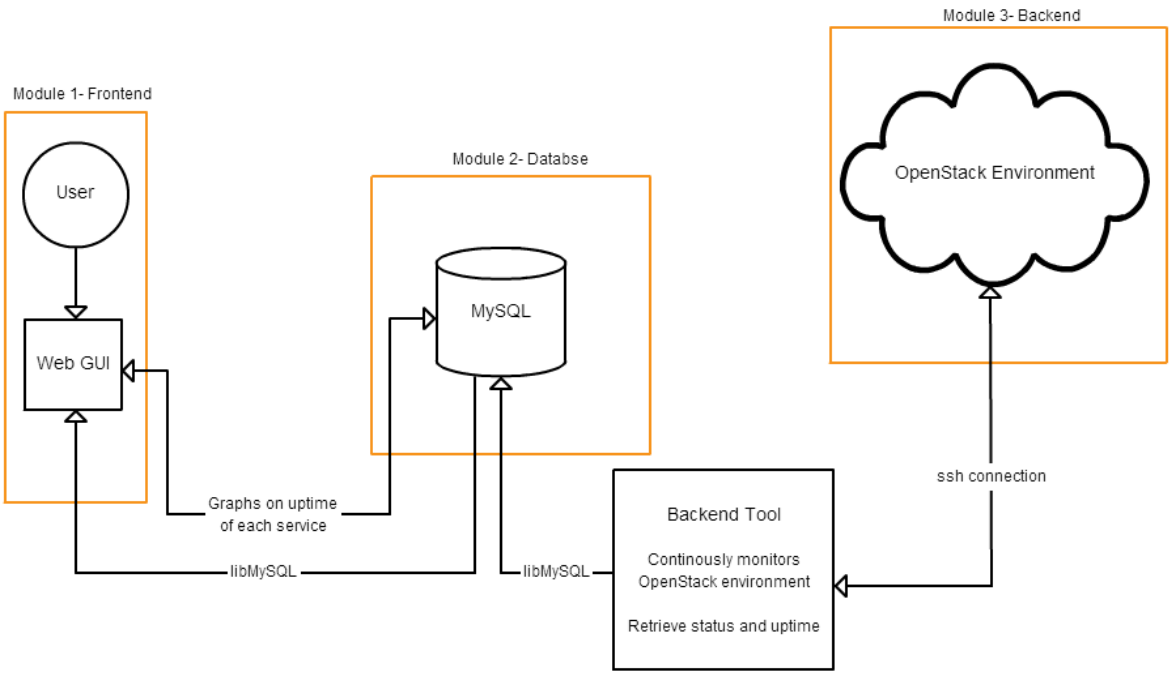


Fig 1: System Architecture

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.

4.1 PROGRAMMING LANGUAGES USED:

- Perl
- Php
- Html
- Java Script
- MySql
- Ajax
- Jquery

4.2 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.

4.3 DATABASES:

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

4.4 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.

4.5 OTHER REQUIREMENTS:

- Open stack environment
- MySQL Database

5) ORGANISATION OF SOURCE CODE:

```
Oceans11/
|
|-----> readme.txt
|-----> install.sh
|-----> run.sh
|-----> Keys/
|       |-----> open.pub
|       |-----> Open
|-----> Front End/
|       |-----> login.php
|       |-----> java.js
|       |-----> os.png
|       |-----> style.css
|-----> Backend/
        |-----> database.pl
        |-----> rrd.pl
        |-----> servicerestart.pl
        |-----> sshcon.pl
        |-----> Uptime of service.pl
```

6) RESTFUL API:

The tool supports getting of data from 3rd party applications via REST API. The data retrieved is exported into JSON file format and later retrieved for plotting of graphs and data analysis.

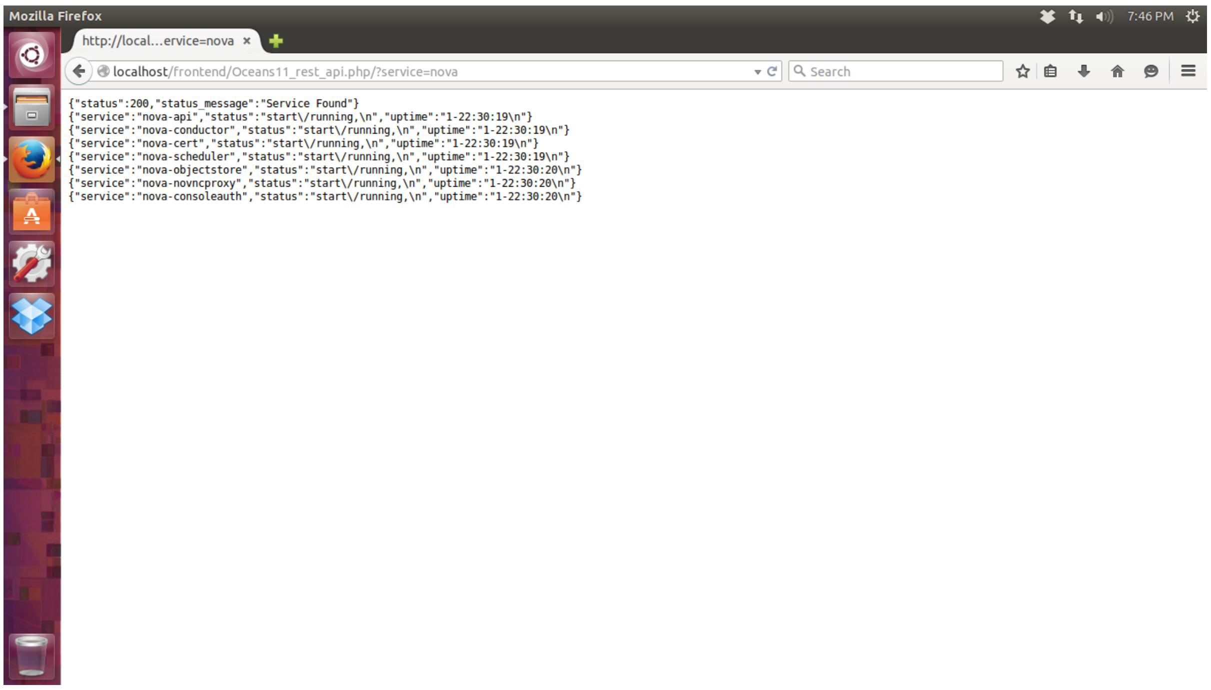
6.1. RESOURCE FOR CPU UTILIZATION DATA:

SOURCE: https://<server_ip>/frontend/Oceans11_rest_api.php/?service=nova

DESCRIPTION: Information can be retrieved about Nova service.

EXAMPLE:

```
{"status":200,"status_message":"Service Found"}
{"service":"nova-api","status":"start\/running,\n","uptime":"1-20:38:11\n"}
{"service":"nova-conductor","status":"start\/running,\n","uptime":"1-20:38:11\n"}
{"service":"nova-cert","status":"start\/running,\n","uptime":"1-20:38:11\n"}
{"service":"nova-scheduler","status":"start\/running,\n","uptime":"1-20:38:12\n"}
{"service":"nova-objectstore","status":"start\/running,\n","uptime":"1-20:38:12\n"}
{"service":"nova-novncproxy","status":"start\/running,\n","uptime":"1-20:38:12\n"}
{"service":"nova-consoleauth","status":"start\/running,\n","uptime":"1-20:38:04\n"}
```


Mozilla Firefox
http://local...ervice=nova
localhost/frontend/Oceans11_rest_api.php/?service=nova
Search
7:46 PM

```
{"status":200,"status_message":"Service Found"}
{"service":"nova-api","status":"start\/running,\n","uptime":"1-22:30:19\n"}
{"service":"nova-conductor","status":"start\/running,\n","uptime":"1-22:30:19\n"}
{"service":"nova-cert","status":"start\/running,\n","uptime":"1-22:30:19\n"}
{"service":"nova-scheduler","status":"start\/running,\n","uptime":"1-22:30:19\n"}
{"service":"nova-objectstore","status":"start\/running,\n","uptime":"1-22:30:20\n"}
{"service":"nova-novncproxy","status":"start\/running,\n","uptime":"1-22:30:20\n"}
{"service":"nova-consoleauth","status":"start\/running,\n","uptime":"1-22:30:20\n"}
```

Fig 2: Rest api

ATTRIBUTE	**DESCRIPTION**
Service:	Gives the type of the service being queried.
Status message:	The unique number assigned to show status of service.
Running Status:	Gives the number about minimum and maximum CPU usage.
Uptime:	Gives the time since system started.

Table 1: nova services

6.2. RESOURCE FOR MEMORY UTILIZATION DATA:

SOURCE: https://server_ip/frontend/Oceans11_rest-api.php/?service=cinder

DESCRIPTION: Information can be retrieved about Cinder service.

EXAMPLE:

```
{"status":200,"status_message":"Service Found"}
{"service":"cinder-scheduler","status":"start\/running,\n","uptime":"4-20:45:41\n"}
{"service":"cinder-api","status":"start\/running,\n","uptime":"4-20:45:41\n"}
```

ATTRIBUTE	DESCRIPTION
Service:	Gives the type of the service being queried.
Status Message:	Give information regarding the status of service.
Uptime:	Uptime of the particular service

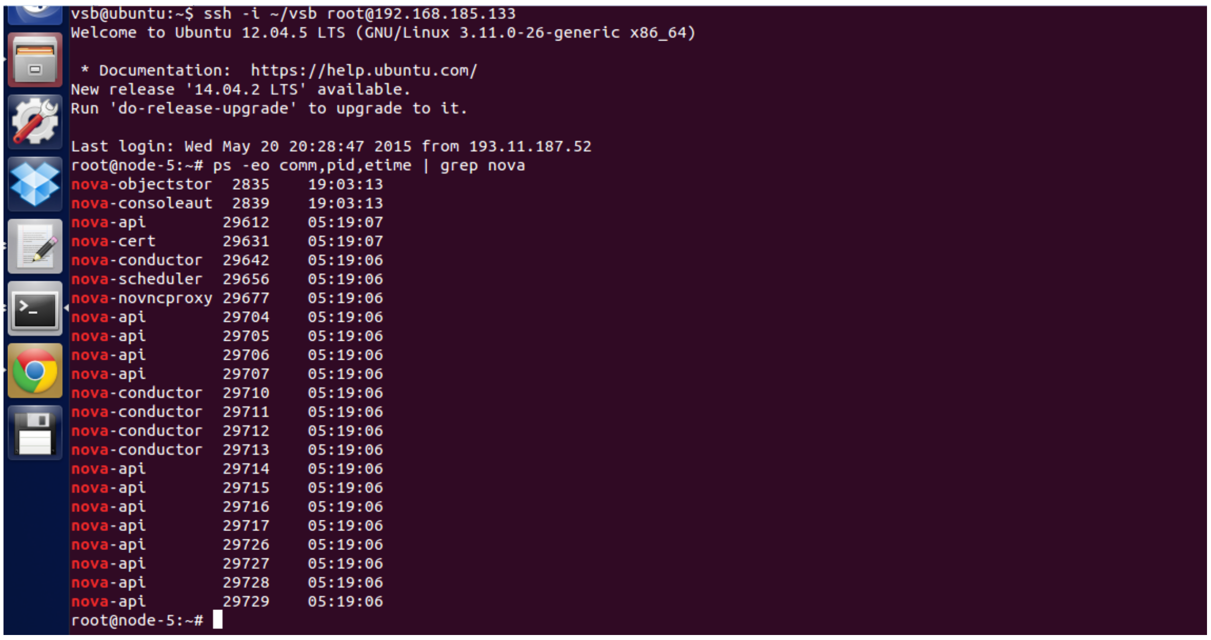
Table 2: cinder service

7) MONITORING PROCESS:

The tool retrieves the information about the services to be monitored from a remote machine having open stack environment via SSH. This allows for remote logging and monitoring of open stack services. The following command establishes ssh connection to remote open stack server:

ssh –i ~/path_to_key root@192.168.185.133

The brief command line implementation can be illustrated as follows:



```
vsb@ubuntu:~$ ssh -i ~/vsb root@192.168.185.133
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.11.0-26-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '14.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed May 20 20:28:47 2015 from 193.11.187.52
root@node-5:~# ps -eo comm,pid,etime | grep nova
nova-objectstor  2835    19:03:13
nova-consoleaut  2839    19:03:13
nova-api        29612    05:19:07
nova-cert       29631    05:19:07
nova-conductor  29642    05:19:06
nova-scheduler  29656    05:19:06
nova-novncproxy 29677    05:19:06
nova-api        29704    05:19:06
nova-api        29705    05:19:06
nova-api        29706    05:19:06
nova-api        29707    05:19:06
nova-conductor  29710    05:19:06
nova-conductor  29711    05:19:06
nova-conductor  29712    05:19:06
nova-conductor  29713    05:19:06
nova-api        29714    05:19:06
nova-api        29715    05:19:06
nova-api        29716    05:19:06
nova-api        29717    05:19:06
nova-api        29726    05:19:06
nova-api        29727    05:19:06
nova-api        29728    05:19:06
nova-api        29729    05:19:06
root@node-5:~#
```

Fig 3: Process Monitoring

8) PARSING OF DATA INTO LOG FILES:

All the outputs of operation that are performed on the remote open stack server are parsed into single log file for reference purposes.

9) EXTENDING THE ABILITIES OF TOOL:

The tool can be further extended by improving the functionality of REST API i.e. we can import the third party tools data to analyze and produce output plots. Also new recipes can be cooked for failure of the services instead of restarting the service every time.

10) REFERENCES:

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

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