Management of Cloud Based Sensor Network using Nagios

MTP Report Submitted to

Indian Institute of Technology Mandi

For the award of degree of

B. Tech

By

Shreya Tangri

Under the guidance of

Prof. Timothy A. Gonsalves

SCHOOL OF COMPUTER AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MANDI

JUNE 2016

CERTIFICATE OF APPROVAL

It is certified that the final Major Technical Report entitled MANAGEMENT OF CLOUD BASED SENSOR NETWORK USING NAGIOS, submitted by SHREYA TANGRI (B12070) to the Indian Institute of Technology Mandi, for the award of the degree of B. Tech. has been accepted after examination held today.

Faculty Advisor

Date: 24th June, 2016

CERTIFICATE

It is certified that the final Major Technical Report entitled MANAGEMENT OF CLOUD BASED SENSOR NETWORK USING NAGIOS, submitted by SHREYA TANGRI (B12070) to the Indian Institute of Technology Mandi, is a record of bonafide work under my supervision and is worthy of consideration for award of the degree of B.Tech of the Institute.

Date: 24th June, 2016 Timothy A. Gonsalves

Place: IIT Kamand, H.P.,India 175005 Supervisor

DECLARATION BY THE STUDENT

This is to certify that the final report titled MANAGEMENT OF CLOUD BASED SENSOR NETWORK USING NAGIOS submitted by me to the Indian Institute of Technology, Mandi for the award of the degree of B. Tech is a bonafide record of work carried out by me under the supervision of Prof. Timothy A Gonsalves. The contents of this MTP, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Date: 24th June, 2016 Shreya Tangri
Place: IIT Kamand, H.P. India 175005 (B12070)

Acknowledgement

I am extremely grateful to my guide and mentor Prof. Timothy A Gonsalves (Director, IIT Mandi) for his continuous support in each and every step for my Major Technical Project. His guidance, patience and keen interest is solely responsible for the successful completion of this project and I enjoyed working constantly in the CNRG Lab for this project.

I am also very thankful to my CNRG friends Merlin Sundar, Subhash Kumar, Reena Singh, Anuksha Jain, Aditi Maan and Tejal Satish for the constant support and timely guidance which lead to successful completion of MTP. I would also like to thanks my faculty advisor Dr. Satinder Sharma.

Last but not the least, I owe my sincere thanks to my parents for their love and support. I a thankful to all those people who have directly or indirectly helped me in completing this project and always motivated me with great confidence.

Shreya Tangri

List of Figures

Contents

Chapter 1

1.1 Introduction

The current networks consist of many complex, interacting hardware and software pieces starting from switches, routers, hosts and other devices that form the physical components of any network to different protocols that control these devices. When thousands of such devices are cobbled together by an organization to form a network, some of these components may become malfunctioned, misconfigured or over utilized. At that time, the network administrator whose job is to keep the network up and running, must be able to respond to such mishaps. For managing such complex networks, he needs network management tools which can help him to monitor, manage and control the network [1].

Network Management is a set of functions to control, plan, deploy, allocate, coordinate and manage network resources. It is the maintenance of all network resources in a good shape to ensure easy availability to the users according to their expectations.

Nagios is an excellent popular open-source computer system and network monitoring application software. It monitors the hosts and services that we add, alerts us when things go down and also shows the status when they come back to the normal state. It is capable of monitoring different services on a server like SMTP, HTTP, SSH, database servers such as MYSQL, SQL, Oracle server and different server resources such as CPU, Memory or Load. It consists of a GUI web interface which depicts the status of the services by three methods i.e. *OK, Warning, Critical*. It supports FCAPS (fault, configuration, accounting, performance and security management).

Cloud Computing provides us a means by which we can access the applications as utilities, over the Internet [2]. It allows us to create, configure and customize applications online. Cloud is an internet or network. Cloud provides a cloud computing services over the network, i.e., on public networks or on private network like WAN, LAN or VPN. Applications like gmail, web conferencing, customer relationship management (CRM) etc are all executing under the cloud.

When we store our photographs, documents or study materials online other than in pendrives or a hard disk, then we are using cloud computing services. Examples of cloud services include online file storage, social networking sites, webmail, and online business applications. It has two models, namely, Deployment model and Service model as mentioned below [2]:

Deployment Model: It defines the type of access to the cloud i.e. public, private, hybrid and community.

1. Public Cloud: Easily accessible to the general public and is not very secure due to its openness. Eg. Gmail.
2. Private Cloud: It allows systems and service to be accessible within an organization. It is more secured due to its private nature.

3. Hybrid Cloud: It is a mixture of private and public clouds. Critical activities are done on the private cloud and non-critical activities are done on the public cloud.
4. Community Cloud: It is accessible to a group of organizations. It is cost effective and more secure than public cloud.

Service Model: It is the reference models on which a group of cloud computing is based.

- Infrastructure as a Service: It provides access to fundamental resources such as virtual machines, storage machines.
- Platform as a Service: It provides runtime environment or a computing platform for applications, development and deployment tools. Eg. LAMP, operating system, database and web server.
- Software as a Service: It allow to use software applications as a service to end-users.

Characteristics of Cloud Computing

- It allows the users to use internet services and resources on a demand basis.
- It is web-based and can be accessed from anywhere.
- It allows many users at a time to share a pool of resources. One can share single physical instance of hardware, database and basic infrastructure.
- It is easy to get the access of the resources at any time. Resources assigned to the customers are automatically monitored.

Sensors are the devices that are used to measure physical variables like temperature, pH, velocity, humidity, pressure etc. in our surroundings. The data is collected and transferred to the computer. The computer software display this data in different formats such as graphs, tables or meter readings.

Networked vibration sensors which are used on the factory floor warn us about a bearing which is about to fall. Mechanics should plan accordingly the overnight maintenance and prevent an expensive disaster. Temperature and humidity sensors used inside a refrigerated grocery truck monitor individual containers and hence reduce the spoilage in the fragile fish. There are few examples of the sensors like wireless humidity sensors for fire detecting, nitrate sensors for detecting industrial and agricultural pollution in rivers.

1.2 The Problem

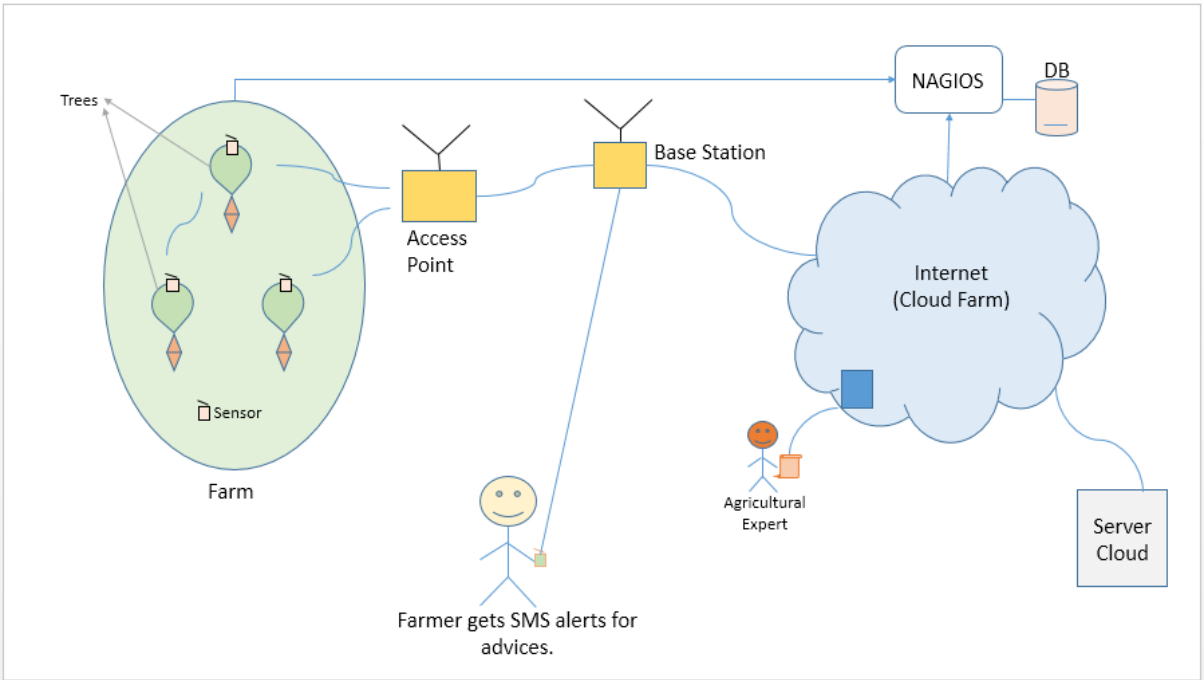


Fig.1.1 An agricultural network with Nagios to monitor sensor and cloud server network [6]

In agriculture too, farmers need network for communicating with other people and take decisions regarding land, labour, livestock etc. They can learn about different chemicals or new techniques on internet if they have a proper connection which will help them in sending queries related to the work. In Fig.1, in a farm, a farmer uses sensors which are the wireless devices useful for collecting information like temperature, humidity etc. These sensors send data via the access point, base station and Internet to the cloud server. Farmers send their queries to the cloud server. The agricultural expert also uses the same cloud server to respond to the farmer's queries. The farmer reads the advice given by the expert.

Proper management of this network is important to ensure timely and correct information to the farmers which will help them to work more efficiently. Nagios is helpful in completing this task.

1.3 The Contributions

1) I have installed Nagios on a PC and Redmine Server in the CNRG Lab. I also added few hosts in both the servers.

2) I have installed pnp4Nagios and configured it with Nagios for getting the graphs for different services defined in Nagios. It is also an important requirement for integrating Nagios with CloudStack. I worked on this in my previous semester.

3) In this semester, I integrated Nagios with sensors and monitored three sensors with Nagios. The three sensors are TMP36, HTU21D and DS18B20 and set up two types of connection i.e. Networked and Standalone Sensor Network.

TMP36 is a low voltage temperature sensor. It provides a voltage output that is linearly proportional to the Celsius temperature. It is easy to use and does not require any

extra calibration for accuracies. The operating temperature range −40°C to +125°C. The device is connected to a ground, 2.7 to 5.5 VDC (power supply) and the voltage can be read on the Vout pin. The output voltage can be converted to temperature easily using the scale factor of 10 mV/8°C [3].

HTU21D is a digital humidity and temperature sensor which provides calibrated, linearized signals in digital in i2c format. It is a low power consumption designed for high volume and cost sensitive applications [4].

DS18B20 is the size of a pea. For this project, we have used waterproof sensor which is enclosed in a stainless thermowell and wrapped with shrinkwrap. It is also called a one wire digital temperature sensor and is very convenient because it just requires one communication wire to communicate with an MCU. This also uses Dallas Semiconductor's (now Maxim) one-wire protocol which allows an array of devices to be controlled with just one communication wire [5].

4) Integration Nagios with CloudStack and Sensor is required for the overall management of the agriculture network.

1.4 Overview

Chapter 1 introduces Nagios, cloud-computing and sensors, the main problem and the contributions. Chapter 2 contains brief background information about Nagios, CloudStack and beagle bone weather station. Chapter 3 presents the design of Nagios and CloudStack, management of Nagios with CloudStack, management of beagle bone weather station i.e. networked and standalone connection. Chapter 4 discusses the implementation and testing of the networks. We first explain how Nagios is installed on the server and integrated with the CloudStack followed by how pnp4Nagios is installed and configured for getting the graphs in Nagios. We also discuss the integration of Nagios with the sensors. It also contains the design for integrating Nagios with multiple sensors. It also shows the results in the form of graphs for the testing done. Chapter 5 presents the conclusions and summary followed by future work and references.

Chapter 2 Background

In this chapter we discuss Nagios, CloudStack and Beaglebone in brief.

2.1 Nagios

Nagios runs on a server as a daemon. It reports a service and host status to the database when its plugins are executed. It was developed by Ethan Galstad, as an open source unix-based enterprise monitoring package with a web-based front-end or console. We can create our own plugins for monitoring a particular host which send the information to Nagios due to which we can see the status of a host we added using the web interface. It also send us email or SMS when a host goes down [7].

The Nagios Architecture Diagram (Fig. 2.1) explains how Nagios works.

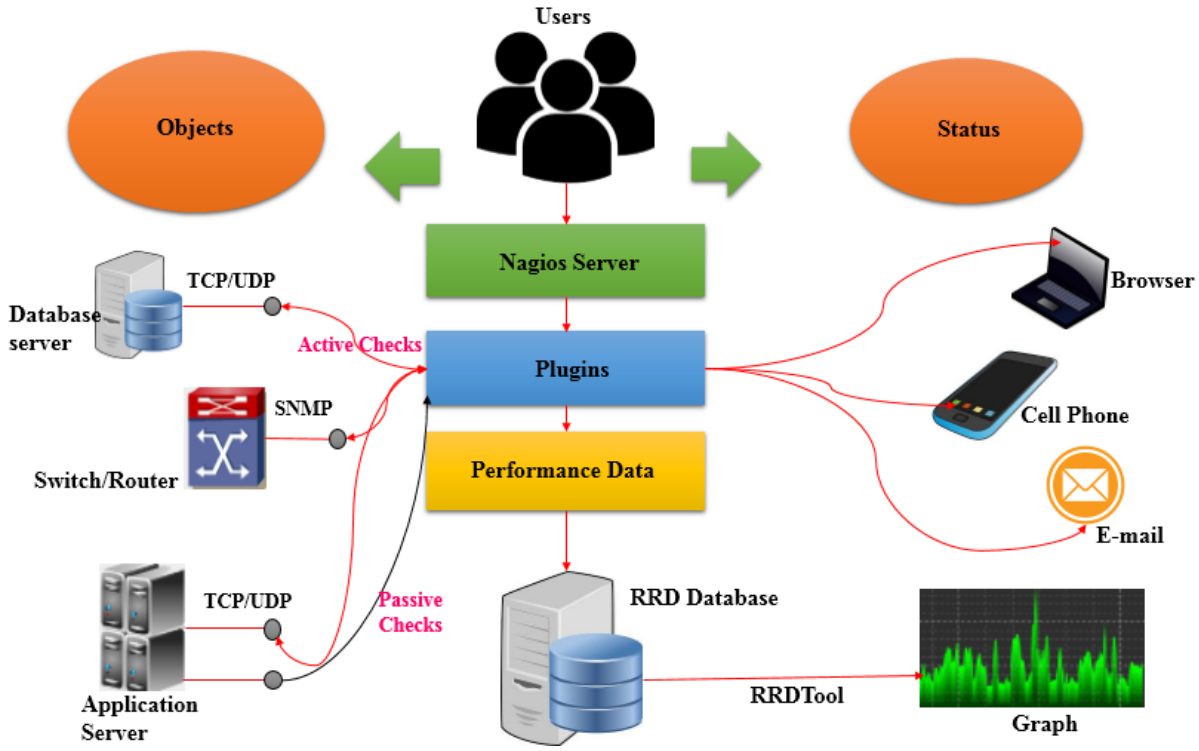


Fig.2.1 Nagios Architecture Diagram

Nagios is selected because of the following reasons:

- Robust and reliable
- Highly Configurable
- Easily Extensible
- Limitless Monitoring
- Runs on many operating systems
- Widely used in IIT Kamand Campus and CNRG Lab

1.4.1 Introduction to RRDTool

Nagios store the collected data in a database for analysis and drawing graphs. For this purpose, it uses RRDtool. It is an acronym for Round Robin Database Tool which is a graphing utility that takes data sets we collect and graphs those points [13]. It correlates time-series data like network bandwidth, temperatures, CPU load or any other data type Protocol collected via SNMP (Simple Network Management Protocol). It is stored in a round-robin database which allows the system storage footprint to remain constant over time. When new data comes in, the oldest data set is removed. Hence, dataset does not grow in size and it requires no maintenance.

There are three basic steps for using RRDtool [13]:

1. **Initializing Database:** We create the rrd database and input data into it. We specify how much data we want, how frequently the data has to be updated and what type of data should be collected like gauge, counter which are the variable types.
2. **Collecting the data sets over time:** Data is collected using a script which we write and enter the same periodically into the database.
3. **Creating Graph:** The last step is to take the data from the rrd database, perform some calculations on the data and create the graph. Later, the graph is used easy viewing but in Nagios, graph comes automatically after the readings are uploaded as in case of standalone or if it is configured with the Nagios as in case of networked.

Inputs of RRDtool for creating a database [14]:

1. "DS:temperature:GAUGE:100000:U:U". In this, DS stands for data source, temperature is a variable name, GAUGE is a variable type,100000 is the heartbeat while U is the limits on maximum and minimum variable values.
2. "RRA:AVERAGE:0.5:1:24, RRA:AVERAGE:0.5:6:10". In this, RRA stands for round robin archive, average is a consolidation function, 0.5 means upto 50% of consolidated points may be unknown. 1:24 RRA averages the data over one five minutes sample twenty four times i.e. for two hours while 6:10 RRA averages over six five minutes values which is thirty minutes for ten times i.e. five hours.
3. We create a database as "sudo rrdtool create Temperature.rrd –start 1451606400 DS:temperature:GAUGE:100000:U:U RRA:AVERAGE:0.5:1:2400 RRA:AVERAGE:0.5:6:2400".
4. We need to specify the start time in epoch format since the code is written in that format.

2.2 CloudStack

CloudStack is an open source IAAS platform that manages and orchestrates pods of storage, network and computer resources to build a public/private IAAS compute cloud [8].

- It provides on demand computing service faster and cheaper.
- It can manage thousands of physical servers installed in geographically distributed basestations.
- It automatically configures the network and storage settings for each virtual machine deployment. A pool of virtual appliances support the operation of configuration of the cloud. These appliances offer services such as firewalling, routing, DHCP, VPN, console proxy, storage access, and storage replication.
- It offers administrator web interface used for provisioning and managing the cloud, as well as the end-user's Web interface which is used for running VMs and managing VM templates.
- It provides a REST-like API for the operation, management and use of the cloud.

CloudStack Infrastructure Overview

- REGIONS: It consists of one or more zones which are managed by one or more management servers.
- ZONES: It consists of one or more pods and secondary storage. CloudStack can have one or more availability zones.
- CLUSTERS: It consists of hosts and primary storage.
- PODS: It is a rack that contains clusters.
- HOSTS: A single mode or hypervisor on which services will be provisioned.

CloudStack Storage

Primary storage:

- Configured at cluster-level close to hosts for better performance.
- Stores all disk-volumes for VMs in a cluster.
- Cluster can have more than one primary storage.
- Local disk.

Secondary Storage:

- Configured at the zone-level.
- Stores templates, ISOs and Snapshots.
- Zone can have more than one secondary storages.
- OpenStack Swift.

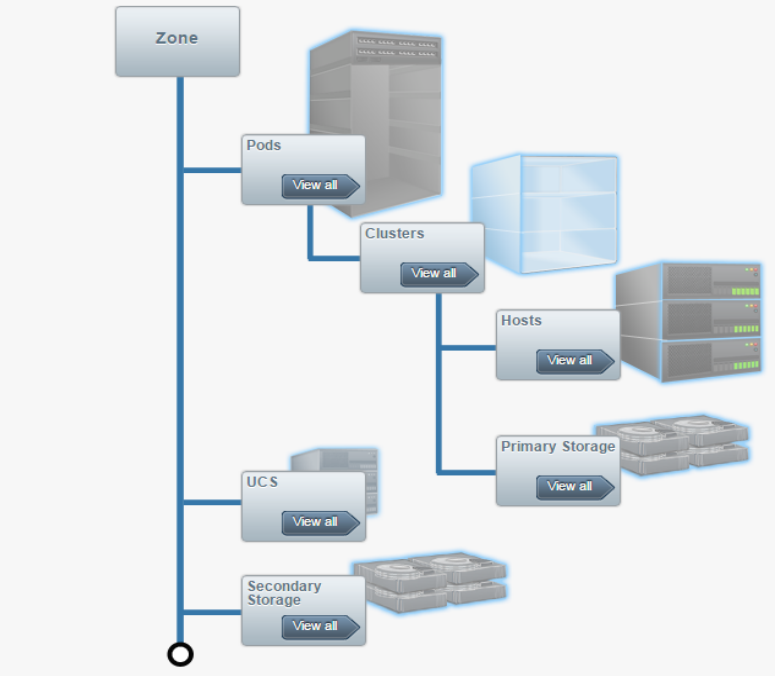


Fig.2.2 Cloud Architecture Overview [9]

2.3 Beagle Bone Weather Station

Beagle Bone is a low-cost credit-card-sized Linux computer that connects with the Internet and runs software such as Android and Ubuntu. Beagle Bone has built in networking i.e. it do not rely on Ethernet connection but we can use services like FTP, Telnet, SSH on board. We can remotely login into the bone and update the code. It act as both the USB device and USB host. It has in-built file system which allows storing, organizing and retrieving data is easy.

It has 720MHz super-scalar ARM Cortex-A8 (armv7a) processor and connectivity is USB client: power, debug and device, USB host, Ethernet etc. The software is 4GB microSD card with Angstrom Distribution and Cloud9 IDE on Node.JS with Bonescript library [10].

The objective of the Weather Station Project is to set up various weather sensors at different locations around the IIT Mandi, Kamand campus. The data collected is stored on a server for analysis. A sensor setup is placed at the Director's Residence. The readings are first stored locally on the beagle bone. The system takes a reading every 5 minutes and records it in the internal storage of the bone. The battery of the bone is to be charged and data has to be transported to the server on a regular basis.

The signals (e.g. Voltage etc.) from the sensor are read at the intervals of 5 minutes. Calculations are performed to convert the signal into appropriate values to determine the temperature, pressure, humidity, wind speed and dust content. The original signal from the sensor, calculated result, date and time are saved on the bone. The data from the bone is transferred using USB Cable into the Nagios server. The data is updated in Nagios and we can see the graph plotted.

Chapter 3 Design

In this chapter we present the design of Nagios and CloudStack and management of beagle bone weather station.

1.5 Management of CloudStack with Nagios

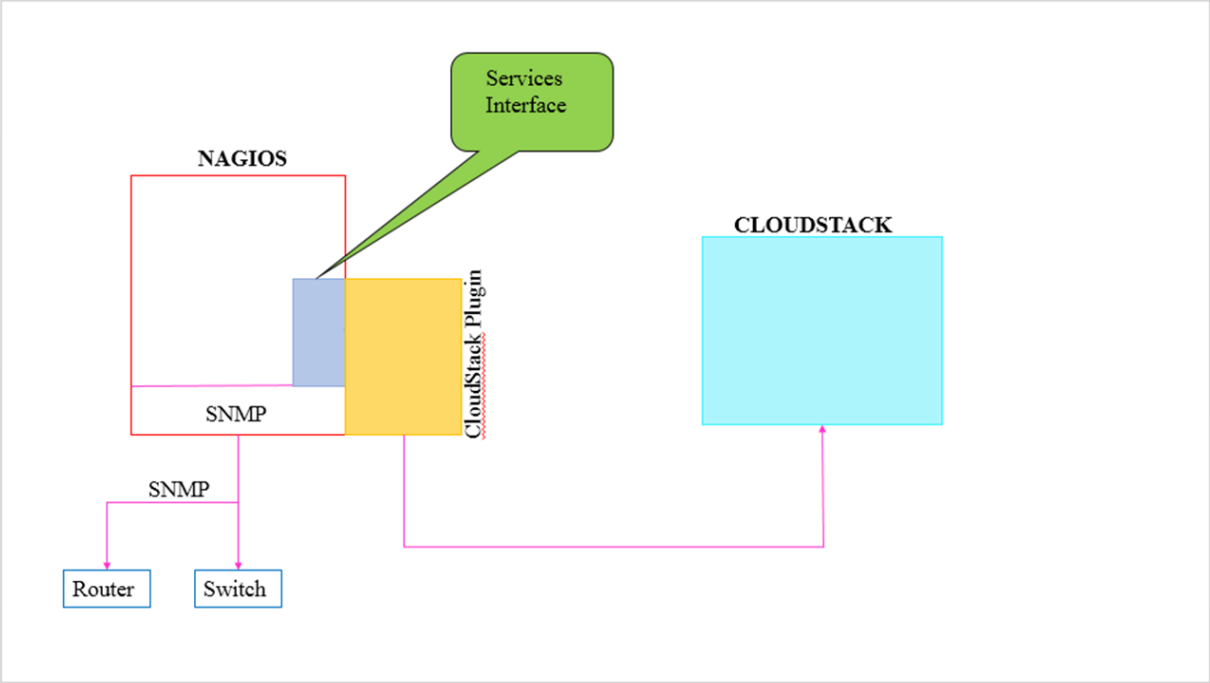


Fig. 3.1 Nagios and CloudStack Integration

Nagios is easily configured with the CloudStack by using the Nagios-CloudStack plugins. Nagios is based on SNMP which is a widely used internet-standard protocol in the network management arena. It is employed to monitor the health of the clouds and other network components. It control the network devices and take immediate actions when the problem arises and also monitors amount of traffic for the interface or how much is the pressure inside the router [7].

For integrating Nagios with CloudStack, we need plugins which are essential for monitoring. Services Interface provides a platform for Nagios to configure services for the monitoring of the CloudStack. Also, installation of *pnp4Nagios* is the most important task since it is an add-on necessary for displaying the graphs.

The "*Nagios-CloudStack*" plugins are written in PHP because we already have a PHP library for CloudStack API and in the Nagios server as we have installed pnp4Nagios before installing these plugins.

APIs (Application Programming Interfaces) are the sets of requirements that govern how one application talks with another. It is a set of routines, protocols and tools for building software applications [11]. It is possible to move information between programs like using sniptool for cutting and pasting.

1.6 Management of Beagle Bone Weather Station

There are two types of connection for the Beagle Bone. One is Networked Connection and the other is Standalone Sensor Network. The whole process is explained in the following sections.

3.2.1 Sensor with Network Connection

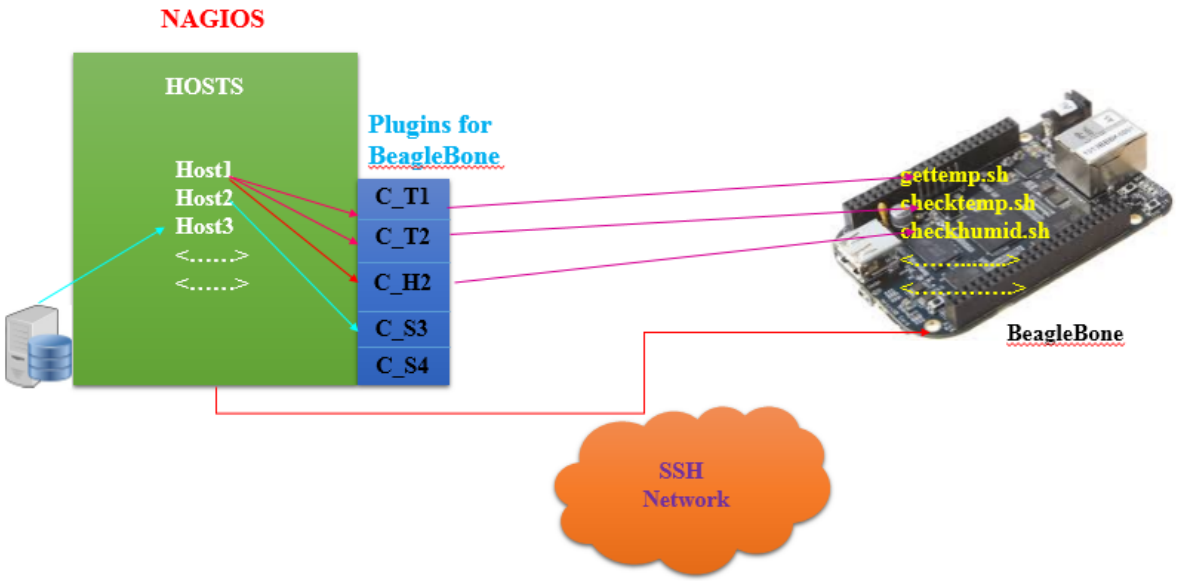


Fig.3.2 Networked Sensor Connection

This is a networked beagle bone connection. The setup is deployed in the CNRG Lab which keeps on collecting data the whole day and consequently the readings file are created in the system. I logged into the setup from my system to monitor the sensors.and made some changes in the configuration file of Beagle Bone.

For logging into another system, we need to have passwordless ssh login. There are three steps involved [12]:

Step 1: Login as Nagios and give the command “ssh-keygen”.

Step 2: ssh-copy-id Nagios@192.168.7.2

Step 3: Login from the system into 192.168.7.2 without password and check if it worked.

Screenshots of ssh login into beagle bone:

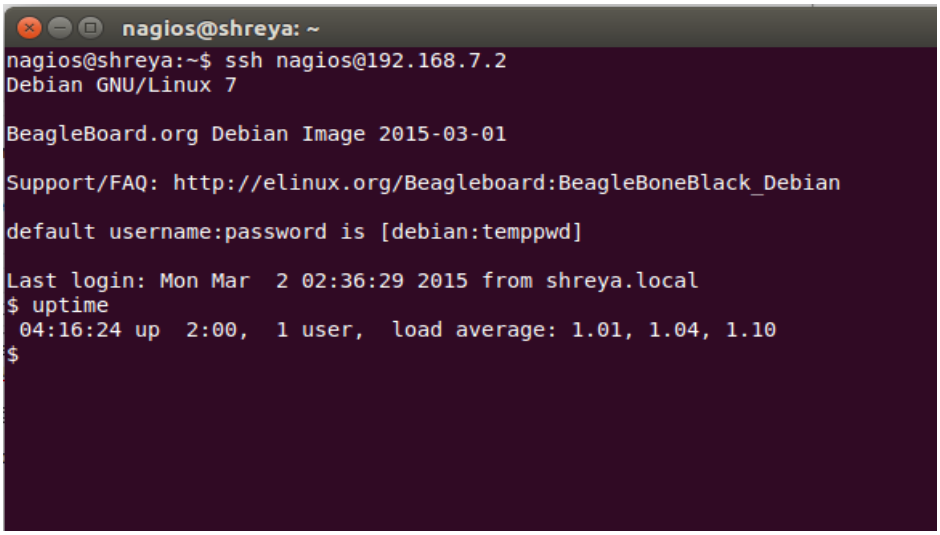


Fig.3.3 SSH Passwordless Login into Beagle Bone

3.2.2 Standalone Sensor Network

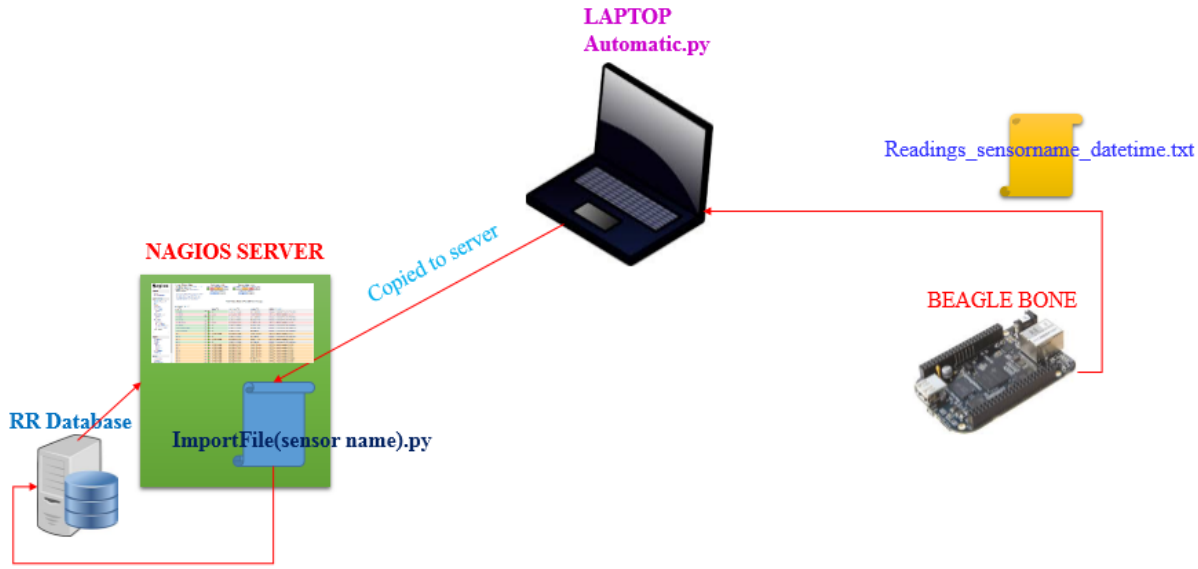


Fig.3.4 Standalone Multiple Sensor Setup

Procedure for the transfer of Sensor Data:

1. Beagle Bone switch periodically polls sensor readings and stores in local files "Readings_sensorname_datetime.txt".
2. Once a day or every few days, Readings_date_sensorname.txt files are manually copied to a laptop.
3. Files are copied from laptop to the Nagios server.
4. ImportFile.py is used to insert data with time-temp value into RRD databases of Nagios.
5. User can view graphs etc.in Nagios.

Chapter 4 Implementation

We present installation of Nagios on the server and integration with the CloudStack. The pnp4Nagios is installed and configured for getting the graphs in Nagios after integrating it with CloudStack. We also discuss the integration of Nagios with the sensors. It also contains the design for integrating Nagios with a number of sensors

4.1 Nagios Installation and Configuration

This section explains how Nagios is installed in the system, how we can add a new host or a service and the configuration of the Nagios [15].

Prerequisites for Nagios installation:

- Linux O/S basics
- Linux basic commands
- Editor: vi and gedit
- Utilities:
  - make
  - ssh
  - tar
  - yum
- MySQL basics
- HTTP basics
- Firewall and routing basics

4.1.1 Installation and Configuration [15]

- Download the latest version of Nagios-4.0.8.tar.gz from http://www.Nagios.org/download/
- Extract Nagios-4.0.8.tar.gz:
  - `$ gunzip Nagios-4.0.8.tar.gz`
- Unpack the tarball:
  - `$ tar –xvf Nagios-4.0.8.tar`
- Nagios is extracted in the directory "Nagios-4.0.8"
- Change the directory to Nagios as:
  - `$ cd Nagios-4.0.8`
- We need to create the user Nagios will run under:
  - `$ adduser Nagios`
  - `$ chown Nagios.Nagios /usr/local/Nagios`

4.1.2 Compilation and Integration

In order for Nagios to be of any use, we need to install some plugins [15]. Go to the directory where we untared "Nagios-4.0.8" and run:

- `./configure`
- `make install`
- `make install-init`
- `make install-commandmode`
- `make install-config`

"make install" will install the files by default in the path "/usr/local/Nagios". For start, stop and restarting the Nagios following commands where used:

- `$ /etc/rc.d/init.d/Nagios start`
- `$ /etc/rc.d/init.d/Nagios stop`
- `$ /etc/rc.d/init.d/Nagios restart`
- `$ /etc/rc.d/init.d/Nagios status`

4.1.3 Nagios Configuration

Once Nagios is installed in our system, the main configuration file "Nagios.cfg" is created in the path "/usr/local/Nagios/etc/Nagios.cfg" [15]. This file helps in starting the Nagios whenever a new host is added to Nagios. There are several other files like commands.cfg, localhost.cfg, templates.cfg which are present by default in the Nagios. We can define or add a new file for a particular host which we want to monitor. The path of the new host has to be specified in "Nagios.cfg" file. We have created several files such as IUATC.cfg, REDMINE.cfg, VM1.cfg, etc. We define the host definition and services we want to monitor for that host as:

1. Host Definitions

```
define host {

        use                     Ubuntu-server,host-pnp  ;Name of
host template to use

        host_name                       IUATC

        address                         10.8.*.*

        check_command            check-host-alive

        check_period                    24x7

        check_interval                  10

        max_check_attempts              3

        initial_state                   u

        notification_interval           30

        notification_options            d,u,r

        }

define host{

        use                     Ubuntu-server,host-pnp  ; Name of
host template to use

        host_name                       REDMINE

        address                         10.8.*.*

        check_command            check-host-alive

        check_period                    24x7

        check_interval                  10

        max_check_attempts              3
```

```
        initial_state                   u
        notification_interval           30
        notification_options            d,u,r
        }
```

2. Service Definitions

Service Definition to “ping” the local machine:

```
define service{
        use                             local-service,srv-pnp
;Name of service template to use
        host_name                       IUATC
        service_description             PING
        check_command                 check_ping!100.0,20%!500.0,60%
        }
```

Service Definition to “check the load” on the local machine.

```
define service{
        use                             local-service,srv-pnp
;Name of service template to use
        host_name                       REDMINE
        service_description             Current Load
       check_command       check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
        }
```

3. “Commands.cfg” is the file which contains the definition of the service defined in a particular host as:

'check_ping' command definition:

```
define command{
        command_name    check_ping
        command_line    $USER1$/check_ping -H $HOSTADDRESS$ -w
$ARG1$ -c $ARG2$ -p 5
        }
```

'check_local_load' command definition:

```
define command{
        command_name    check_local_load
        command_line    $USER1$/check_load -w $ARG1$ -c $ARG2$
        }
```

4. After adding the path in Nagios.cfg, Nagios is first verified and then restarted by:

“/usr/local/Nagios/bin/Nagios -v /usr/local/Nagios/etc/Nagios.cfg”

“sudo service Nagios restart”

5. Screenshots of Nagios’s Hosts

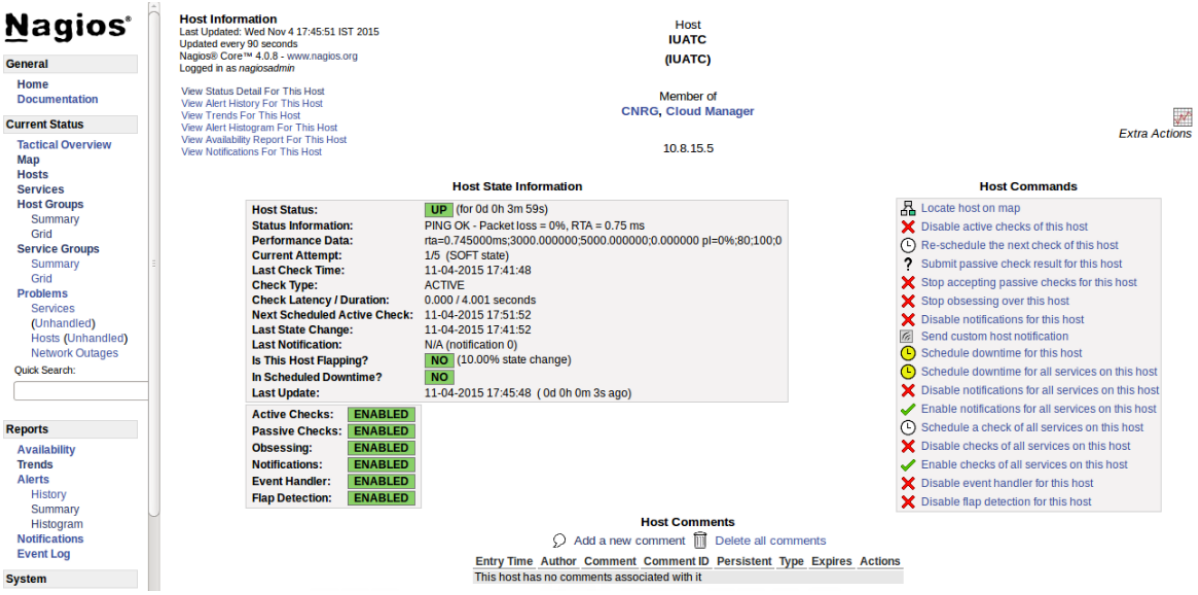


Fig.4.1 IUATC Host on the Nagios Screen

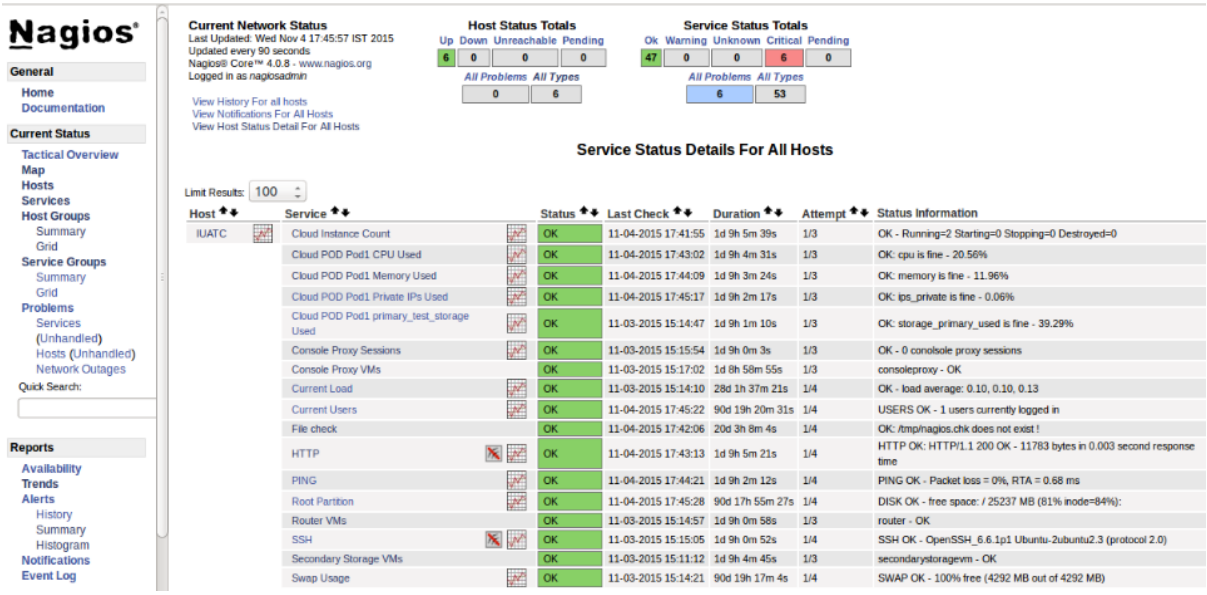


Fig.4.2 Detailed services for the host IUATC

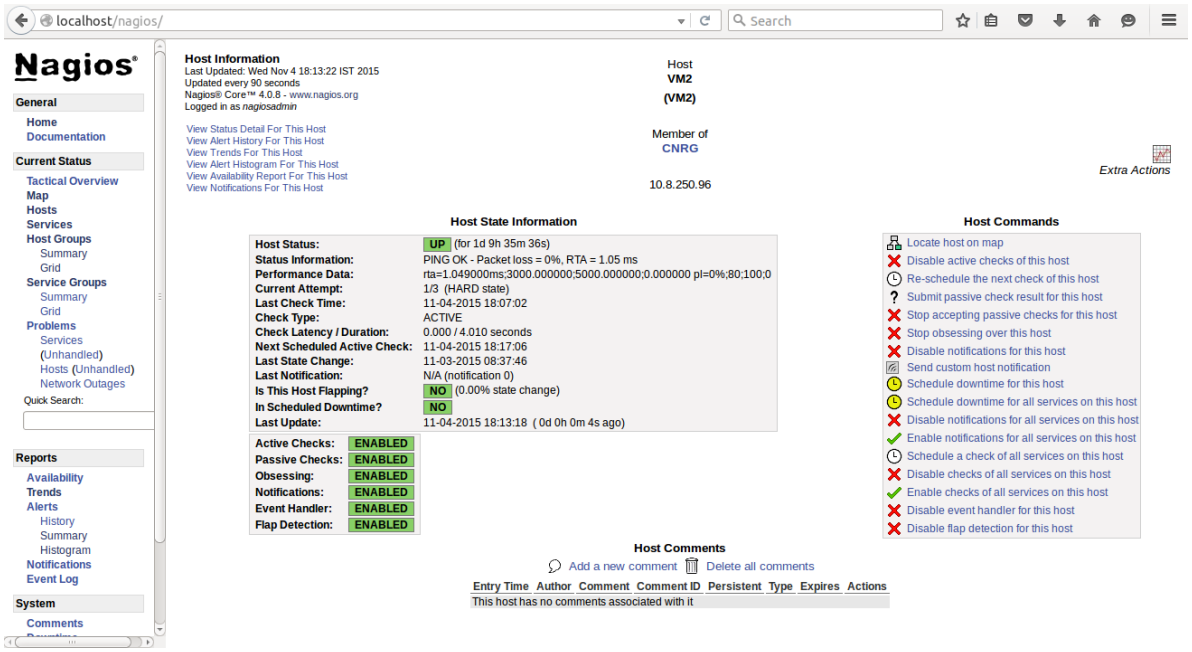


Fig.4.3 Host VM2 on the Nagios Screen

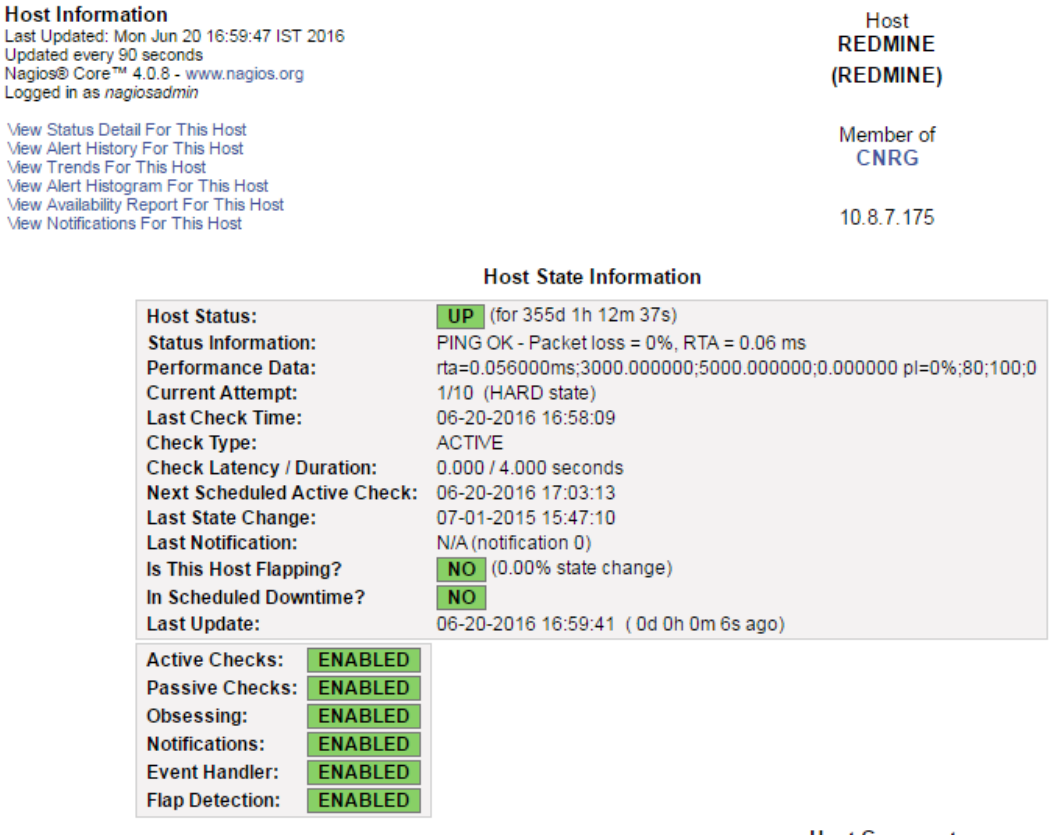


Fig.4.4 Host Redmine on the Nagios Screen

4.2 Introduction to PNP4Nagios:

Pnp4Nagios is a framework written in perl, PHP and C for automatically analysing performance data collected by the Nagios plugins. The data is collected into RRD databases for the display in the Nagios Server. It is integrated well with the standard Nagios plugins and create useable graphs. The appearance of the graphs can be customized according to our needs. Below steps explain the installation, configuration of pnp4Nagios with Nagios and CloudStack [16].

4.2.1 Installation and Configuration of Pnp4Nagios [16]:

1. Download pnp4Nagios-0.6.10.tar.gz and extract the same by giving the following commands:
   - `$ gunzip pnp4Nagios-0.6.10.tar.gz`
   - `$ tar –xvf pnp4Nagios-0.6.10.tar`
2. Go to the extracted pnp4Nagios directory and run the following commands:
   - `./configure`
   - `make all`
   - `make install`
   - `make install-webconf`
   - `make install-config`
3. Next, we need to add few lines in the Nagios.cfg file in the following way:

```
process_performance_data=1
service_perfdata_file=/usr/local/pnp4Nagios/var/service-
perfdata
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\
tTIMET::$TIMET$\tHOSTNAME::$HOSTNAME$\tSERVICEDESC
::$SERVICEDESC$\tSERVICEPERFDATA::$SERVICEPERFDATA$\tSERV
ICECHECKCOMMAND::$SERVICECHECKCOMMAND$\tHOSTSTATE
::$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$\tSERVICESTA
TE::$SERVICESTATE$\tSERVICESTATETYPE::$SERVICESTATETYPE$
service_perfdata_file_mode=a
service_perfdata_file_processing_interval=15
service_perfdata_file_processing_command=process-service-
perfdata-file
host_perfdata_file=/usr/local/pnp4Nagios/var/host-
perfdata
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET
::$TMET$\tHOSTNAME::$HOSTNAME$\tHOSTPERFDATA
::$HOSTPERFDATA$\tHOSTCHECKCOMMAND::$HOSTCHECKCOMMAND$\tH
OSTSTATE::$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$
host_perfdata_file_mode=a
host_perfdata_file_processing_interval=15
host_perfdata_file_processing_command=process-host-
perfdata-file
```

4. At the end of the commands.cfg file which is in the path "/usr/local/Nagios/etc/objects/commands.cfg", add the following lines:

```
define command{
command_name                process-service-perfdata-file
command_line
/usr/local/pnp4Nagios/libexec/process_perfdata.pl --
bulk=/usr/local/pnp4Nagios/var/service-perfdata
}

define command{
command_name                process-host-perfdata-file
command_line
/usr/local/pnp4Nagios/libexec/process_perfdata.pl --
bulk=/usr/local/pnp4Nagios/var/host-perfdata
}
```

5. Add the following two lines in your templates.cfg file which is in the path "/usr/local/Nagios/etc/objects/templates.cfg":

```
define host {
name                    host-pnp
register                0
action_url
      ../../../pnp4Nagios/share/index.php/graph?host=$HOST
NAME$
}

define service {
name                    srv-pnp
register                0
action_url
      ../../../pnp4Nagios/share/index.php/graph?host=$HOST
NAME$&srv=$SERVICEDESC$
}
```

6. We have to make few changes in our host definition as shown:

```
define host{
use                             Ubuntu-server,host-pnp ;
Name of host template to use
host_name                       IUATC
address                         10.8.*.*
check_command                   check-host-alive
check_period                    24x7
check_interval                  10
max_check_attempts              5
initial_state                   u
notification_interval           30
notification_options            d,u,r
}
```

```
define service{
use                                 local-service,srv-pnp ;
Name of service template to use
host_name                           IUATC
service_description                 Current Load
check_command    check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
}
```

7. Restart Nagios and Apache2:

- `sudo service Nagios restart`
- `sudo service apache2 restart`

Go to your browser and type http://localhost/pnp4Nagios/. If everything goes well, we will get the following screen:

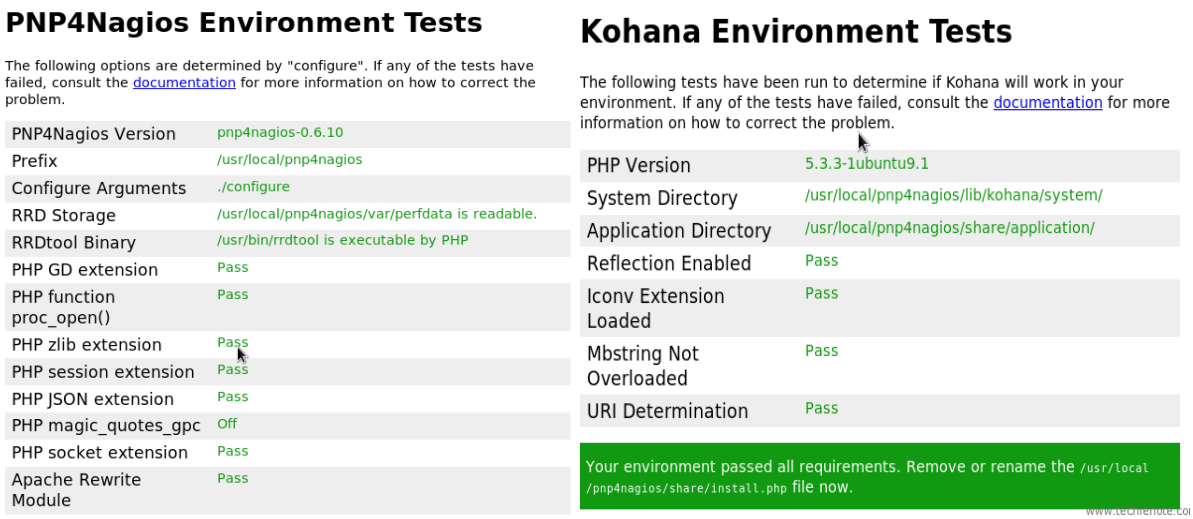


Fig.4.5 Output for correct installation of pnp4Nagios

8. Delete “install.php” file which is in the path “/usr/local/Nagios/pnp/” due to which we will see some graphs.

9. Screenshots of pnp4Nagios graph:

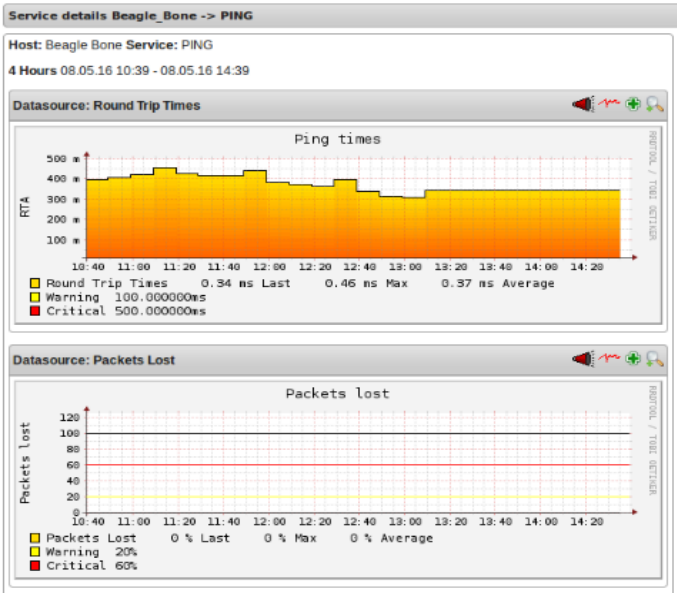


Fig.4.6 Pnp4Nagios Graph for Ping Service

4.2.2 Configuration of Nagios with CloudStack [17]:

This explains the procedure for integration, configuration of Nagios with CloudStack.

1. Install the 'Nagios-CloudStack' zip from "`https://github.com/jasonhancock/Nagios-CloudStack`" in the path "`/usr/local/Nagios/`" as it the Nagios plugins required for working with CloudStack.

2. Extract the Nagios-CloudStack-master.zip by giving command:

   - "`Unzip Nagios-CloudStack-master.zip`"

3. A directory named 'Nagios-CloudStack-master' is formed. Go to that directory:

   - `cd Nagios-CloudStack-master`

4. This directory has another sub-directories and files like plugins, pnp4Nagios, README etc. Go to the "plugins" folder and copy the php files from the plugins' directory into the Nagios' plugin directory i.e. "`/usr/local/Nagios/libexec/`" (path on my machine). "libexec" contains all the plugins of Nagios.

5. Next, go to the "pnp4Nagios" directory inside the "Nagios-CloudStack-master" which contains a folder named templates having a "check_cloud_instances.php" file. This file is to be copied into the Nagios' "`/pnp4Nagios/templates/`" directory i.e. "`/usr/local/pnp4Nagios/templates/`" (path on my machine). These plugins are written in php is because we have a php library for the CloudStack API and php is already installed on the Nagios server when we installed pnp4Nagios.

6. Plugins retrieve information from CloudStack via the API, thus they require that the CloudStack php client ("`https://github.com/jasonhancock/CloudStack-php-client`") is present. Download "CloudStack-php-client-master" zip from the above link in the path "`/usr/local/Nagios/`". Extract the "CloudStack-php-client-master.zip" by giving command: `Unzip CloudStack-php-client-master.zip`. A folder named "CloudStack-php-client-master" is formed [18].

7. Go to this directory and open the file config.php in the editor of your choice. In this file, give your CloudStack's API_ENDPOINT, API_KEY, API_SECRET_KEY since this config.php file tells the plugins where the CloudStack API is and what credential to use for it. This configuration file can live anywhere on the filesystem. We can tell the plugins about this location via the -f parameter.

8. Now, we select the host (IUATC is the host which I am configuring) in Nagios which is a cloud computing machine and we have to make changes in its commands.cfg file by adding following lines:

```
#'check_cloud_instances' command definition
define command {
command_name                check_cloud_instances
command_line                /usr/bin/php
$USER1$/check_cloud_instances.php -f
/usr/local/Nagios/CloudStack-php-client/config.php
}

#'Check_cloud_consoleproxy_sessions' command definition
define command {
command_name                check_cloud_consoleproxy_sessions
command_line                /usr/bin/php
$USER1$/check_cloud_consoleproxy_sessions.php -f
/usr/local/Nagios/CloudStack-php-client/config.php -w 5 -c 10
}

#'check_cloud_systemvms' command definition
define command {
command_name                check_cloud_systemvms
command_line                /usr/bin/php
$USER1$/check_cloud_systemvms.php -f
/usr/local/Nagios/CloudStack-php-client/config.php -t $ARG1$
}

#'check_cloud_capacity' command definition
define command {
command_name                check_cloud_capacity
command_line                /usr/bin/php
$USER1$/check_cloud_capacity.php -f
/usr/local/Nagios/CloudStack-php-client/config.php -t $ARG1$ -
w 80 -c 90 -n $ARG2$
}
```

9. In these commands, “$USER1$” is the path for Nagios plugins “`/usr/local/Nagios/libexec/`”. Similarly we add few lines in “templates.cfg” for the service.

```
define service {

name                    generic-service-graphed

use                     generic-service
```

```
action_url
../../../pnp4Nagios/share/index.php/graph?host=$HOSTNAME$&srv=
$SERVICEDESC$
register              0
}
```

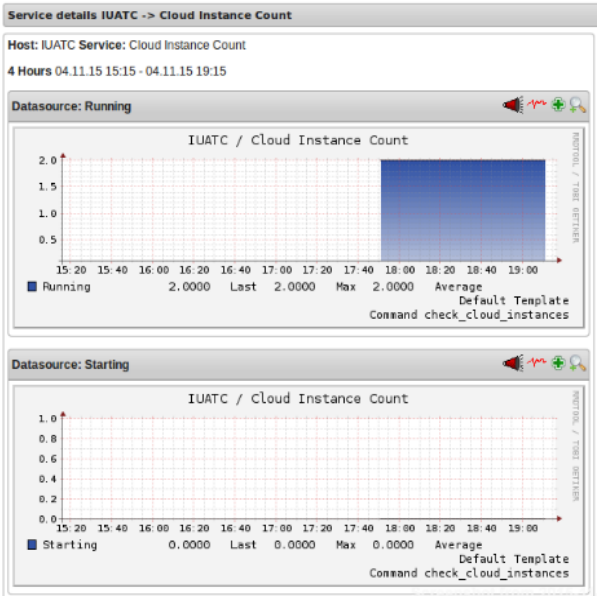
10. We also add few lines for the service that we want our Nagios to configure for CloudStack.

```
define service{
use                    generic-service-graphed
host_name              IUATC
service_description    Cloud Instance Count
hostgroups             Cloud Manager
check_command          check_cloud_instances
}
define service{
 use                   generic-service-graphed
 service_description   Console Proxy Sessions
 hostgroups            Cloud Manager
 check_command         check_cloud_consolesproxy_sessions
}
define service{
 use                    generic-service-graphed
 host_name              IUATC
 service_description    Cloud POD **** CPU Used
 hostgroups             Cloud Manager
 check_command          check_cloud_capacity!cpu!Pod ****
}
define service{
 use                    generic-service-graphed
 service_description    Cloud POD **** Memory Used
 hostgroups             Cloud Manager
 check_command          check_cloud_capacity!memory!Pod ****
}
define service{
use                    generic-service
```

```
service_description   Router VMs
hostgroups            Cloud Manager
check_command         check_cloud_systemvms!router
}
```

11. After making these changes, restart Nagios and we can see the graphs.

12. Screenshots



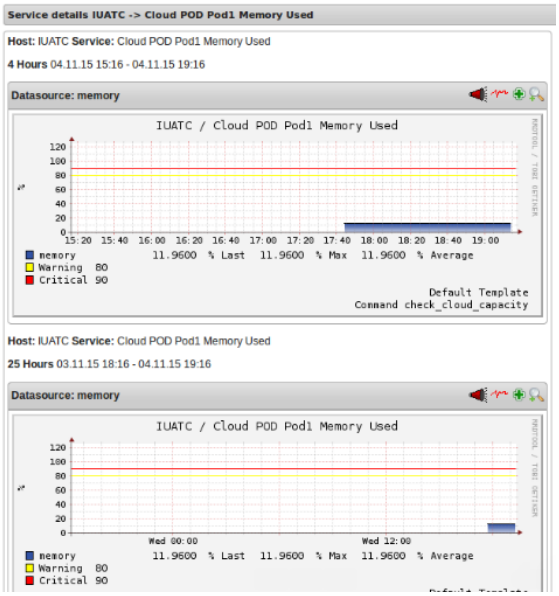


Fig.4.7 Cloud Instance Count for the host IUATC Fig.4.8 Memory Used for the host IUATC

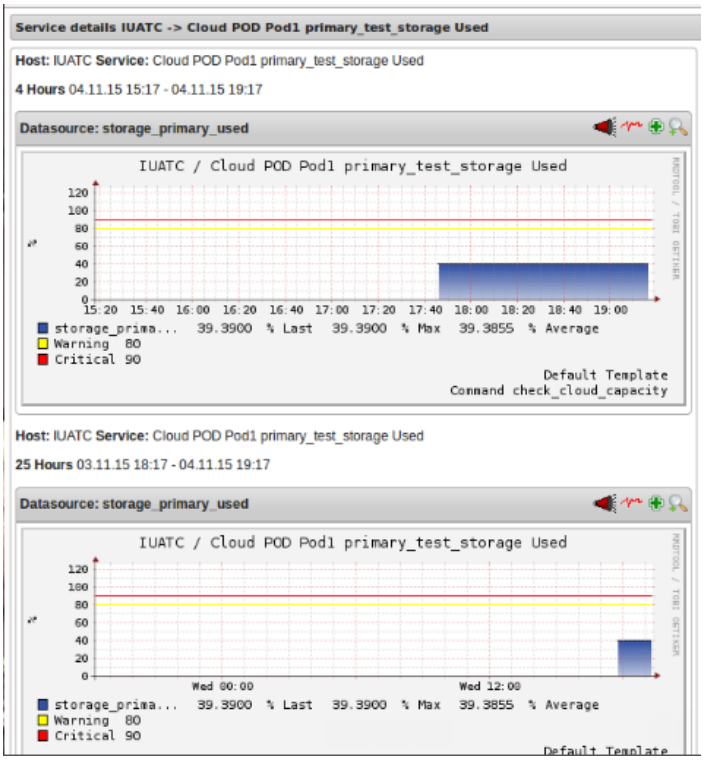


Fig.4.9 Primary Storage for the host IUATC

4.3 Automatic Host Discovery

This task was done to detect an automatic host which is added in the CloudStack. Nagios should be able to detect that automatically added host. Below is the pseudo code and step by step procedure.

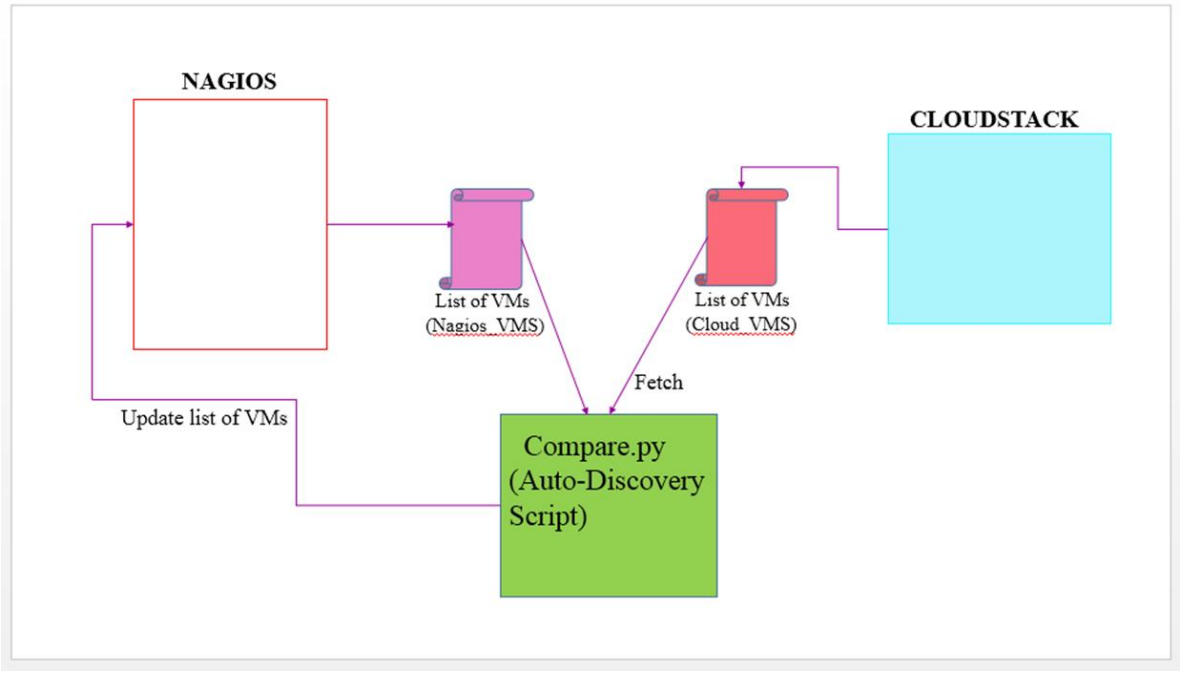


Fig.5.0 Design for Auto-Discovery Host

4.3.1 Pseudo Code for the Auto-Discovery Script

```
while(1)
{
  Get VMs list from Nagios and CloudStack-Api
      If (cloud_list file is equal to Nagios_list file)
  No change in Nagios
      If (New VMs list is not equal to the old VMs list)
  For (VM added or deleted)
{
    Create a file in /path/objects/VM_template.cfg
    Open /objects/Vm_template.cfg and read into VMn.cfg
    Replace @host-name with VMn
    Find IP address of VMn from cloud_ips.list
    Replace @ip-address with Ipn
    Write VMn config to /path/VMn.cfg
}
    To delete a VMn from Nagios.cfg
{
    Delete /path/VMn.cfg
}
```

4.3.2 Procedure and Results

1. Get list of VMs from CloudStack API

```
$python
Python 2.7.6
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> import urllib2
>>> import urllib
>>> import hashlib
>>> import hmac
>>> import base64
>>> baseurl='http://10.8.*.*:****/client/api?'
```

```
>>> request={}

>>> request['command']='listVirtualMachines'

>>> request['response']='json'

>>> request ['apikey']
='cMIAE35gr_vFIK2Ve6TeS14qzk7iRc9fxFJqC4ji8eHAKaSdZhmt8UCBKJIJBqG
bbQtG9xt8jiXsFFaDRXy6-A'

>>> secretkey ='xdP1f3RGybsoSoSPqjfRvmyFZ9SpXlXE8JyevJG3WlROs0a-
W9ppa6M-Nkwk9HiE_NiyrCX3A2ctZTS_xquiNw'

>>> request_str='&'.join (['='.join
([k,urllib.quote_plus(request[k])]) for k in request.keys()])

>>> request_str

'apikey
=cMIAE35gr_vFIK2Ve6TeS14qzk7iRc9fxFJqC4ji8eHAKaSdZhmt8UCBKJIJBqGb
bQtG9xt8jiXsFFaDRXy6-A&command=listVirtualMachines&response=json'

>>>
sig_str='&'.join(['='.join([k.lower(),urllib.quote_plus(request[k
].lower().replace('+','%20'))])for k in
sorted(request.iterkeys())])

>>> sig_str
'apikey=cMIAE35gr_vFIK2Ve6TeS14qzk7iRc9fxFJqC4ji8eHAKaSdZhmt8UCBK
JIJBqGbbQtG9xt8jiXsFFaDRXy6-
A&command=listVirtualMachines&response=json'

>>> sig = hmac.new(secretkey,sig_str,hashlib.sha1).digest()

>>> sig

'F! \Xf6.....................................................'

>>> sig

'RiH2zq +7ChnC80DsHsc5MJD3Wig=\n'

>>>sig=urllib.quote_plus (base64.encodestring
(hmac.new(secretkey,sig_str,hashlib.sha1).digest()).strip())

>>> req

'http://10.8.*.*:8080/client/api?apikey=cMIAE35gr_vFIK2Ve6TeS14qz
k7iRc9fxFJqC4ji8eHAKaSdZhmt8UCBKJIJBqGbbQtG9xt8jiXsFFaDRXy6-
A&command=listVirtualMachines&response=json&
signature=RiH2zq%2B7ChnC8oDsHsc5MjD3W1g%3D'

>>> res = urllib2.urlopen(req)

>>> res.read()
```

2. Output of CloudStack-API in JSON

```
{ "listvirtualmachinesresponse" : { "count":3 ,"virtualmachine" :
[ {"id":"2842ca78-8267-4492-9a53-
0857f73e1235","name":"VM3","displayname":"VM3","account":"admin",
"domainid":"87735755-5ba5-11e5-acb5-
001e674aae88","domain":"ROOT","created":"2016-02-
```

08T12:41:37+0530","state":"Running","haenable":false,"zoneid":"e0
0e1ef4-0b78-4463-b7f1-
86fddc0a92d9","zonename":"Zone1","hostid":"0e89cb44-1dc4-4f67-
ae80-489a75be949e","hostname":"iuatc","templateid":"364b67f9-
8c42-49d0-b1de-
79e471c5c35b","templatename":"ubuntu15.04","templatedisplaytext":
"iso-ubuntu15.04","passwordenabled":false,"isoid":"364b67f9-8c42-
49d0-b1de-
79e471c5c35b","isoname":"ubuntu15.04","isodisplaytext":"iso-
ubuntu15.04","serviceofferingid":"9c3ce452-a0cc-42be-95b6-
0aee0cbe6d43","serviceofferingname":"large
instance","diskofferingid":"5b86a3ee-8f79-4aea-b30f-
79744dcdf377","diskofferingname":"Large","cpunumber":2,"cpuspeed"
:2000,"memory":2048,"cpuused":"2.84%","networkkbsread":1220982,"n
etworkkbswrite":3797,"diskkbsread":2034692,"diskkbswrite":6264088
,"diskioread":83266,"diskiowrite":64686,"guestosid":"8f64eb26-
5ba5-11e5-acb5-
001e674aae88","rootdeviceid":0,"rootdevicetype":"ROOT","securityg
roup":[{"id":"3cd44df1-5ba8-11e5-8b19-
001e674aae88","name":"default","description":"Default Security
Group","account":"admin","ingressrule":[],"egressrule":[],"tags":
[]}],"nic":[{"id":"bfcb640d-939e-4d24-b7c7-
86dd9788258b","networkid":"5cdc3782-0614-4c5a-a327-
c2e6aa65de48","networkname":"defaultGuestNetwork","netmask":"255.
255.0.0","gateway":"10.8.0.1","ipaddress":"10.8.250.12","broadcas
turi":"vlan://untagged","traffictype":"Guest","type":"Shared","is
default":true,"macaddress":"06:2f:f0:00:00:72"}],"hypervisor":"KV
M","instancename":"i-2-6-
VM","tags":[],"affinitygroup":[],"displayvm":true,"isdynamicallys
calable":false,"ostypeid":103}, {"id":"b6ea23df-93ee-4bcb-b816-
0cbb712f3b91","name":"VM2","displayname":"VM2","account":"admin",
"domainid":"87735755-5ba5-11e5-acb5-
001e674aae88","domain":"ROOT","created":"2015-09-
21T11:50:45+0530","state":"Running","haenable":false,"zoneid":"e0
0e1ef4-0b78-4463-b7f1-
86fddc0a92d9","zonename":"Zone1","hostid":"0e89cb44-1dc4-4f67-
ae80-489a75be949e","hostname":"iuatc","templateid":"123fa0e0-
9bb5-4146-b9e1-
28edfbe87159","templatename":"ubuntu15.04","templatedisplaytext":
"ubuntu15.04","passwordenabled":false,"serviceofferingid":"71047c
fe-036e-4025-81da-42290bd0bede","serviceofferingname":"Medium
Instance","diskofferingid":"5b86a3ee-8f79-4aea-b30f-
79744dcdf377","diskofferingname":"Large","cpunumber":1,"cpuspeed"
:1000,"memory":1024,"cpuused":"2.16%","networkkbsread":1281810,"n
etworkkbswrite":3821,"diskkbsread":1230849,"diskkbswrite":1237720
,"diskioread":49058,"diskiowrite":45564,"guestosid":"8f2e2357-
5ba5-11e5-acb5-
001e674aae88","rootdeviceid":0,"rootdevicetype":"ROOT","securityg
roup":[{"id":"3cd44df1-5ba8-11e5-8b19-
001e674aae88","name":"default","description":"Default Security
Group","account":"admin","ingressrule":[],"egressrule":[],"tags":
[]}],"nic":[{"id":"b21c8eb3-e777-4a86-a23c-
34255968c019","networkid":"5cdc3782-0614-4c5a-a327-
c2e6aa65de48","networkname":"defaultGuestNetwork","netmask":"255.
255.0.0","gateway":"10.8.0.1","ipaddress":"10.8.250.96","broadcas
turi":"vlan://untagged","traffictype":"Guest","type":"Shared","is

```
default":true,"macaddress":"06:6b:58:00:00:c6"}],"hypervisor":"KV
M","instancename":"i-2-5-
VM","tags":[],"affinitygroup":[],"displayvm":true,"isdynamicallys
calable":true,"ostypeid":99}, {"id":"2dc722d9-a029-4ace-a47a-
7e267cc2098f","name":"VM1","displayname":"VM1","account":"admin",
"domainid":"87735755-5ba5-11e5-acb5-
001e674aae88","domain":"ROOT","created":"2015-09-
15T18:45:52+0530","state":"Running","haenable":false,"zoneid":"e0
0e1ef4-0b78-4463-b7f1-
86fddc0a92d9","zonename":"Zone1","hostid":"0e89cb44-1dc4-4f67-
ae80-489a75be949e","hostname":"iuatc","templateid":"364b67f9-
8c42-49d0-b1de-
79e471c5c35b","templatename":"ubuntu15.04","templatedisplaytext":
"iso-ubuntu15.04","passwordenabled":false,"isoid":"364b67f9-8c42-
49d0-b1de-
79e471c5c35b","isoname":"ubuntu15.04","isodisplaytext":"iso-
ubuntu15.04","serviceofferingid":"71047cfe-036e-4025-81da-
42290bd0bede","serviceofferingname":"Medium
Instance","diskofferingid":"a447ca0c-b14c-437d-b137-
6318a059d430","diskofferingname":"Medium","cpunumber":1,"cpuspeed
":1000,"memory":1024,"cpuused":"0.59%","networkkbsread":1279316,"
networkkbswrite":3086,"diskkbsread":172842,"diskkbswrite":0,"disk
ioread":4772,"diskiowrite":0,"guestosid":"8f64eb26-5ba5-11e5-
acb5-
001e674aae88","rootdeviceid":0,"rootdevicetype":"ROOT","securityg
roup":[{"id":"3cd44df1-5ba8-11e5-8b19-
001e674aae88","name":"default","description":"Default Security
Group","account":"admin","ingressrule":[],"egressrule":[],"tags":
[]}],"nic":[{"id":"c538ceb4-9747-4dc6-87fb-
7d3a1b67d1b0","networkid":"5cdc3782-0614-4c5a-a327-
c2e6aa65de48","networkname":"defaultGuestNetwork","netmask":"255.
255.0.0","gateway":"10.8.0.1","ipaddress":"10.8.250.63","broadcas
turi":"vlan://untagged","traffictype":"Guest","type":"Shared","is
default":true,"macaddress":"06:d5:20:00:00:a5"}],"hypervisor":"KV
M","instancename":"i-2-3-
VM","tags":[],"affinitygroup":[],"displayvm":true,"isdynamicallys
calable":false,"ostypeid":103} ] } }
```


3. Get the list of hosts from the Nagios-API

- Install Nagios-API by the following steps [19]:
  - sudo git clone https://github.com/xb95/Nagios-api.git
    sudo apt-get install python-pip
  - Install Nagios-API Dependencies(diesel,greenlet,twiggy and python-openssl) by giving command 'sudo pip install diesel'
- In order to test whether Nagios-API is running or not, give the following command:

```
"./Nagios-api -s /usr/local/Nagios/var/status.dat -l
/usr/local/Nagios/var/Nagios.log -c
/usr/local/Nagios/var/rw/Nagios.cmd"
```

- The output is:

```
[2016/02/11 04:53:57] {diesel} INFO|Listening on port 6315,
starting to rock and roll!
[2016/02/11 04:53:57] {diesel} WARNING|Starting diesel <hand-
rolled select.epoll>
```

- By running the command "./Nagios-cli --raw state | python -mjson.tool", we get the JSON OUTPUT:


```
{"IUATC": {
"active_checks_enabled": "1",
"comments": {},
"current_attempt": "1",
"current_state": "0",
"downtimes": {},
"last_check": "1455111236",
"last_hard_state": "0",
"last_notification": "1455099510",
"last_state_change": "1455099510",
"max_attempts": "5",
"notifications_enabled": "1",
"performance_data": {
"pl": "0%",
"rta": "0.801000ms"
},
"plugin_output": "PING OK - Packet loss = 0%, RTA = 0.80 ms",
"problem_has_been_acknowledged": "0",
"scheduled_downtime_depth": "0",
"services": {
"Cloud Instance Count": {
"active_checks_enabled": "1",
"comments": {},
"current_attempt": "1",
"current_state": "0",
"downtimes": {},
"last_check": "1455111567",
"last_hard_state": "0",
"last_notification": "0",
"last_state_change": "1455099567",
"max_attempts": "3",
"notifications_enabled": "1",
"performance_data": {
"Destroyed": 0,
"Running": 3,
"Starting": 0,
"Stopping": 0
}
```


4. Code for converting the Json response into Python response

```
#!/usr/bin/python
import urllib2
import json
from preety import pretty
req = urllib2.Request('http://localhost:6315/state')
opener = urllib2.build_opener()
f = opener.open(req)
response = json.loads(f.read())
pretty(response)
def pretty(d, indent=0):
for key, value in d.iteritems():
print '\t' * indent + str(key) + " :"
if isinstance(value, dict):
pretty(value, indent+1)
```

```
else:
print '\t' * (indent+1) + str(value)
```

5. Output of Nagios-API response in python

```
content:
VM1:
active_checks_enabled : 1
current_attempt : 1
downtimes : problem_has_been_acknowledged : 0
last_hard_state : 1
notifications_enabled : 1
current_state : 1
plugin_output : PING CRITICAL - Packet loss = 100%
last_check : 1455111457
performance_data : rta : 5000.000000ms pl : 100%
last_state_change : 1447388641
scheduled_downtime_depth : 0
services : HTTP : active_checks_enabled : 1
current_attempt : 4
downtimes : problem_has_been_acknowledged : 0
last_hard_state : 2
notifications_enabled : 0
current_state : 2
plugin_output : CRITICAL - Socket timeout after 10 seconds
last_check : 1455111404
performance_data :
last_state_change : 1444963541
scheduled_downtime_depth : 0
max_attempts : 4
last_notification : 0
comments :
PING : active_checks_enabled : 1
current_attempt : 4
downtimes : problem_has_been_acknowledged : 0
last_hard_state : 2
notifications_enabled : 1
current_state : 2
plugin_output : PING CRITICAL - Packet loss = 100%
last_check : 1455111471
performance_data : rta : 500.000000ms pl : 100%
last_state_change : 1447389030
scheduled_downtime_depth : 0
max_attempts : 4
last_notification : 0
```

5. Code for getting CloudStack-API Response in Python

```
#!/usr/bin/python
import urllib2
import simplejson as json
import requests
from preety import pretty
url=('http://10.8.15.5:8080/client/api?apikey=cMIAE35gr_vFIK2Ve6T
eS14qzk7iRc9fxFJqC4ji8eHAKaSdZhmt8UCBKJIJBqGbbQtG9xt8jiXsFFaDRXy6
-A&command=listVirtualMachines&response=json&
```

```
signature=RiH2zq%2B7ChnC8oDsHsc5MjD3W1g%3D')
response = json.loads(urllib2.urlopen(url).read())
pretty(response)
```

6. Output of CloudStack-API Response in Python

```
listvirtualmachinesresponse :
count :
3
virtualmachine :
[{'domain': 'ROOT', 'domainid': '87735755-5ba5-11e5-acb5-
001e674aae88', 'haenable': False, 'isoid': '364b67f9-8c42-49d0-
b1de-79e471c5c35b', 'templatename': 'ubuntu15.04', 'diskioread':
83266, 'diskiowrite': 64686, 'securitygroup': [{'egressrule': [],
'account': 'admin', 'description': 'Default Security Group',
'tags': [], 'ingressrule': [], 'id': '3cd44df1-5ba8-11e5-8b19-
001e674aae88', 'name': 'default'}], 'zoneid': 'e00e1ef4-0b78-
4463-b7f1-86fddc0a92d9', 'isodisplaytext': 'iso-ubuntu15.04',
'ostypeid': 103, 'passwordenabled': False, 'instancename': 'i-2-
6-VM', 'id': '2842ca78-8267-4492-9a53-0857f73e1235', 'cpuused':
'2.84%', 'rootdeviceid': 0, 'isoname': 'ubuntu15.04', 'hostname':
'iuatc', 'displayvm': True, 'diskofferingname': 'Large', 'state':
'Running', 'guestosid': '8f64eb26-5ba5-11e5-acb5-001e674aae88',
'networkkbswrite': 3797, 'cpuspeed': 2000, 'serviceofferingid':
'9c3ce452-a0cc-42be-95b6-0aee0cbe6d43', 'zonename': 'Zone1',
'isdynamicallyscalable': False, 'displayname': 'VM3', 'tags': [],
'diskkbsread': 2034692, 'nic': [{'networkid': '5cdc3782-0614-
4c5a-a327-c2e6aa65de48', 'macaddress': '06:2f:f0:00:00:72',
'networkname': 'defaultGuestNetwork', 'gateway': '10.8.0.1',
'traffictype': 'Guest', 'broadcasturi': 'vlan://untagged',
'netmask': '255.255.0.0', 'type': 'Shared', 'ipaddress':
'10.8.250.12', 'id': 'bfcb640d-939e-4d24-b7c7-86dd9788258b',
'isdefault': True}], 'memory': 2048, 'diskkbswrite': 6264088,
'templateid': '364b67f9-8c42-49d0-b1de-79e471c5c35b',
'affinitygroup': [], 'account': 'admin', 'hostid': '0e89cb44-
1dc4-4f67-ae80-489a75be949e', 'name': 'VM3', 'networkkbsread':
1220982, 'created': '2016-02-08T12:41:37+0530', 'hypervisor':
'KVM', 'rootdevicetype': 'ROOT', 'cpunumber': 2,
'diskofferingid': '5b86a3ee-8f79-4aea-b30f-79744dcdf377',
'serviceofferingname': 'large instance', 'templatedisplaytext':
'iso-ubuntu15.04'}, {'domain': 'ROOT', 'domainid': '87735755-
5ba5-11e5-acb5-001e674aae88', 'haenable': False, 'templatename':
'ubuntu15.04', 'diskioread': 49058, 'diskiowrite': 45564,
'securitygroup': [{'egressrule': [], 'account': 'admin',
'description': 'Default Security Group', 'tags': [],
'ingressrule': [], 'id': '3cd44df1-5ba8-11e5-8b19-001e674aae88',
'name': 'default'}], 'zoneid': 'e00e1ef4-0b78-4463-b7f1-
86fddc0a92d9', 'cpunumber': 1, 'ostypeid': 99, 'passwordenabled':
False, 'instancename': 'i-2-5-VM', 'id': 'b6ea23df-93ee-4bcb-
b816-0cbb712f3b91', 'cpuused': '2.16%', 'hostname': 'iuatc',
'displayvm': True, 'diskofferingname': 'Large', 'state':
'Running', 'guestosid': '8f2e2357-5ba5-11e5-acb5-001e674aae88',
'networkkbswrite': 3821, 'cpuspeed': 1000, 'serviceofferingid':
'71047cfe-036e-4025-81da-42290bd0bede', 'zonename': 'Zone1',
'isdynamicallyscalable': True, 'displayname': 'VM2', 'tags': [],
'diskkbsread': 1230849, 'nic': [{'networkid': '5cdc3782-0614-
```

```
4c5a-a327-c2e6aa65de48', 'macaddress': '06:6b:58:00:00:c6',
'networkname': 'defaultGuestNetwork', 'gateway': '10.8.0.1',
'traffictype': 'Guest', 'broadcasturi': 'vlan://untagged',
'netmask': '255.255.0.0', 'type': 'Shared', 'ipaddress':
'10.8.250.96', 'id': 'b21c8eb3-e777-4a86-a23c-34255968c019',
'isdefault': True}], 'memory': 1024, 'diskkbswrite': 1237720,
'templateid': '123fa0e0-9bb5-4146-b9e1-28edfbe87159',
'affinitygroup': [], 'account': 'admin', 'hostid': '0e89cb44-
1dc4-4f67-ae80-489a75be949e', 'name': 'VM2', 'networkkbsread':
1281810, 'created': '2015-09-21T11:50:45+0530', 'hypervisor':
'KVM', 'rootdevicetype': 'ROOT', 'rootdeviceid': 0,
'diskofferingid': '5b86a3ee-8f79-4aea-b30f-79744dcdf377',
'serviceofferingname': 'Medium Instance', 'templatedisplaytext':
'ubuntu15.04'}, {'domain': 'ROOT', 'domainid': '87735755-5ba5-
11e5-acb5-001e674aae88', 'haenable': False, 'isoid': '364b67f9-
8c42-49d0-b1de-79e471c5c35b', 'templatename': 'ubuntu15.04',
'diskioread': 4772, 'diskiowrite': 0, 'securitygroup':
[{'egressrule': [], 'account': 'admin', 'description': 'Default
Security Group', 'tags': [], 'ingressrule': [], 'id': '3cd44df1-
5ba8-11e5-8b19-001e674aae88', 'name': 'default'}], 'zoneid':
'e00e1ef4-0b78-4463-b7f1-86fddc0a92d9', 'isodisplaytext': 'iso-
ubuntu15.04', 'ostypeid': 103, 'passwordenabled': False,
'instancename': 'i-2-3-VM', 'id': '2dc722d9-a029-4ace-a47a-
7e267cc2098f', 'cpuused': '0.59%', 'rootdeviceid': 0, 'isoname':
'ubuntu15.04', 'hostname': 'iuatc', 'displayvm': True,
'diskofferingname': 'Medium', 'state': 'Running', 'guestosid':
'8f64eb26-5ba5-11e5-acb5-001e674aae88', 'networkkbswrite': 3086,
'cpuspeed': 1000, 'serviceofferingid': '71047cfe-036e-4025-81da-
42290bd0bede', 'zonename': 'Zone1', 'isdynamicallyscalable':
False, 'displayname': 'VM1', 'tags': [], 'diskkbsread': 172842,
'nic': [{'networkid': '5cdc3782-0614-4c5a-a327-c2e6aa65de48',
'macaddress': '06:d5:20:00:00:a5', 'networkname':
'defaultGuestNetwork', 'gateway': '10.8.0.1', 'traffictype':
'Guest', 'broadcasturi': 'vlan://untagged', 'netmask':
'255.255.0.0', 'type': 'Shared', 'ipaddress': '10.8.250.63',
'id': 'c538ceb4-9747-4dc6-87fb-7d3a1b67d1b0', 'isdefault':
True}], 'memory': 1024, 'diskkbswrite': 0, 'templateid':
'364b67f9-8c42-49d0-b1de-79e471c5c35b', 'affinitygroup': [],
'account': 'admin', 'hostid': '0e89cb44-1dc4-4f67-ae80-
489a75be949e', 'name': 'VM1', 'networkkbsread': 1279316,
'created': '2015-09-15T18:45:52+0530', 'hypervisor': 'KVM',
'rootdevicetype': 'ROOT', 'cpunumber': 1, 'diskofferingid':
'a447ca0c-b14c-437d-b137-6318a059d430', 'serviceofferingname':
'Medium Instance', 'templatedisplaytext': 'iso-ubuntu15.04'}]
```

7. Code for List of Hosts from Nagios

```
#!/usr/bin/python
import urllib2
import simplejson as json
import requests
#from tree import IPNetwork, IPAddress
from preety import pretty
from netaddr import *
from pprint import pprint
```

```
def Nagios_list():
obj =[]
url=('http://localhost:6315/state')
response = json.loads(urllib2.urlopen(url).read())
a = response['content']
for key in a.keys():
obj.append(key)
return(obj)
a = Nagios_list()
print a
```

8. Code for list of VMS from CloudStack

```
#!/usr/bin/python
import urllib2
import simplejson as json
import requests
#from tree import IPNetwork, IPAddress
from netaddr import *
from preety import pretty
def l2():
obj ={}
url=('http://10.8.*.*:****/client/api?apikey=cMIAE35gr_vFIK2Ve6Te
S14qzk7iRc9fxFJqC4ji8eHAKaSdZhmt8UCBKJIJBqGbbQtG9xt8jiXsFFaDRXy6-
A&command=listVirtualMachines&
response=json&signature=RiH2zq%2B7ChnC8oDsHsc5MjD3W1g%3D')
response = json.loads(urllib2.urlopen(url).read())
a = response['listvirtualmachinesresponse']
b = a['virtualmachine']
for i in range(0, len(b)):
c = b[i]
if IPAddress(c["nic"][0]["ipaddress"]) in
IPNetwork("10.8.*.*/16"):
obj[c["name"]] = c["nic"][0]["ipaddress"]
#obj.append(c["nic"])
return(obj)
a = l2()
print a
```

9. Compare.py

```
#!/usr/bin/python
#Get VMs list from Nagios-Api and CloudStack-Api:
NagiosApiResponse="$(curl http://localhost:6315/state)"
CloudStackApiResponse="$(curl http://10.8.*.*:****/client/api?api
key=cMIAE35gr_vFIK2Ve6TeS14qzk7iRc9fxFJqC4ji8eHAKaSdZhmt8UCBKJIJB
qGbbQtG9xt8jiXsFFaDRXy6-
A&command=listVirtualMachines&response=json&signature=RiH2zq%2B7C
hnC8oDsHsc5MjD3W1g%3D)"
echo "$NagiosApiResponse"
echo "$CloudStackApiResponse"
#In Nagios_list.py: write each host name in a new line in
Nagios_hosts.list
python /usr/local/Nagios/Nagios-api/Nagios_list.py
```

```
# In cloud_list.py:
#1.1 write each host name in a new line in cloud_hosts.list
python /usr/local/Nagios/Nagios-api/cloud_list.py
#1.2 write each host name and its IP in a new line in
cloud_ips.list
#less /usr/local/Nagios/Nagios-api/cloud_ips.list
#Comparison between Nagios_hosts.list and cloud_hosts.list using
diff command and the output is being saved into diff.txt file:
diff cloud_hosts.list Nagios_hosts.list > tmp/diff.txt
./Manage_VM.py
```

10. Manage_VM.py:

```
#!/usr/bin/env python
#manage_VM.py: Written to add or delete a VM in Nagios.cfg.
#Last Modified on 23/12/15 10:00 pm
import os
from path import *
with open(file_diff) as f:
for line in f:
if line.startswith(">"):
a, machine = line.split()
#print "machine is being deleted " + machine
command = "./deleteVM.py "+machine
os.system(command)
#print "Machine " + machine + "removed from cloud service"
elif line.startswith("<"):
a, machine = line.split()
#print "machine is being added " + machine
command = "./addVM.py "+machine
os.system(command)
#print "machine added to cloudVMs"
#Add machine
```

11. addVM.py

```
#!/usr/bin/env python
#addVM.py: Written for adding a new host to Nagios.cfg
#Last modified on 23/12/15 8:15pm
''' Module to add new VM '''
import sys
import os.path
from path import *
VM_name = sys.argv [1] #VMn is passed as an argument
ip = {} #Finding ipaddress of VMn from cloud_ips.list
with open("cloud_ips.list") as cloud_ip:
print cloud_ip
for list in cloud_ip:
(key , value) = list.split(" ")
ip[key] = value
filein = "/usr/local/Nagios/etc/objects/VM_template.cfg"
#VM_template.cfg file is being created
fileout = "/usr/local/Nagios/etc/cloudVMs/" + str(VM_name) +
".cfg" #The VM_template.cfg is being read and hence is copied
into cloudVMs.
if (os.path.isfile(fileout)):
```

```
print("warning "+ fileout +" host already exists")
else:
f1 = open(filein, 'r')
f2 = open(fileout, 'w')
for line in f1:
f2.write(line.replace('@host_name',
VM_name).replace('@ip_address', ip[VM_name])) #Host name and
ipaddress is being replaced
f1.close()
f2.close()
with open("/usr/local/Nagios/etc/Nagios.cfg", "a") as myfile:
text = "cfg_file=/usr/local/Nagios/etc/cloudVMs/" + VM_name
+"\n"
myfile.write(text)
```

12. When we give the command `"./addVM.py VM1"`, VM1.cfg file is being created in the folder `"/usr/local/Nagios/etc/cloudVMs/VM1.cfg"` and this path is added in the file "Nagios.cfg".
    If we want to delete a file from the Nagios.cfg:

13. deleteVM.py

```
#!/usr/bin/env python
#This file is used for deleting a Vm from Nagios.cfg.
#Last modified on 23/12/15 by Shreya at 9:45pm
import os
import sys
from path import *
machine = sys.argv [1]
fileName = cloudPath+machine+".cfg"
if (os.path.isfile(fileName)):
command = "rm "+fileName
os.system(command)
print "Host Removed from CloudVMs"
else:
print machine + " Host is not Present in cloudVMs"
```

14. Screenshots of the five python Codes

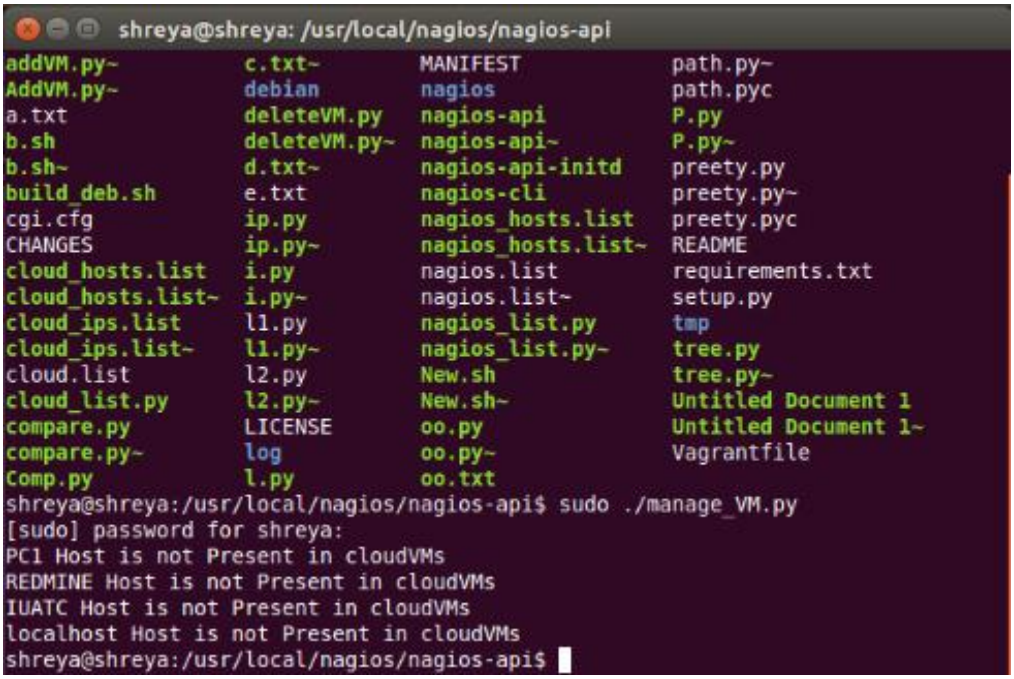


Fig.5.1 Output of manage_VM.py

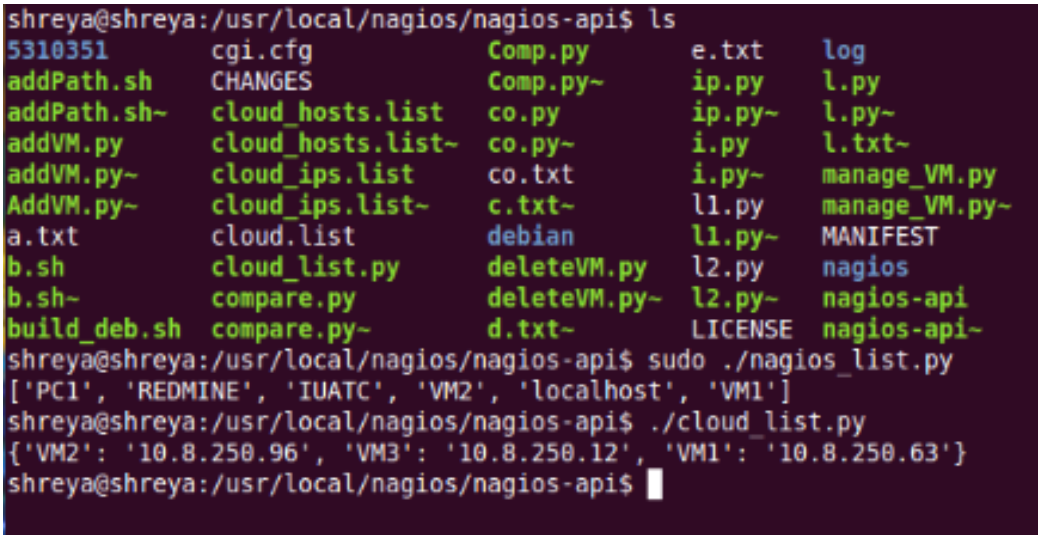


Fig.5.2 Output of Nagios's host and Cloud's VM List

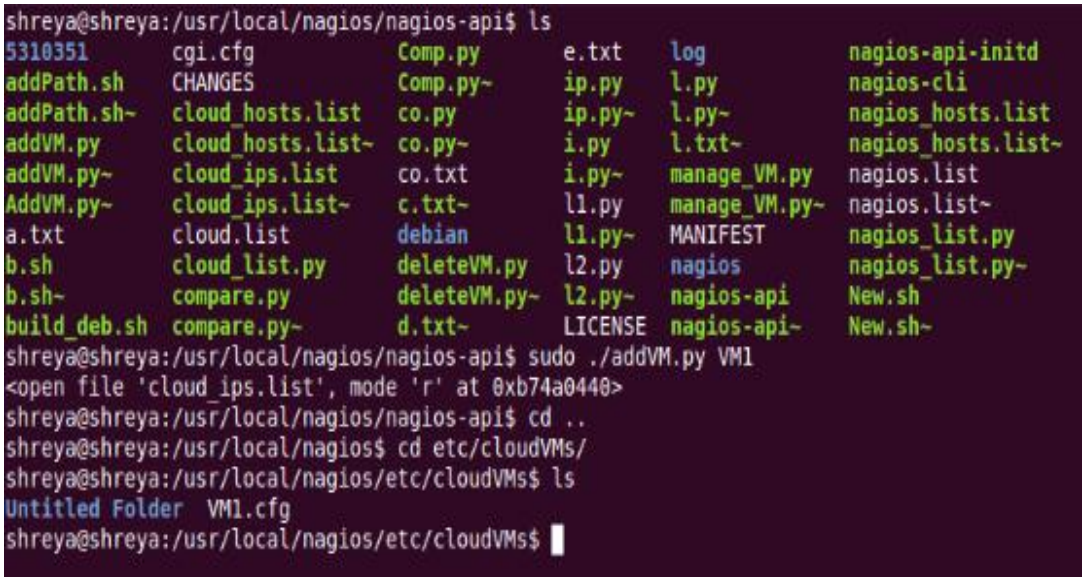
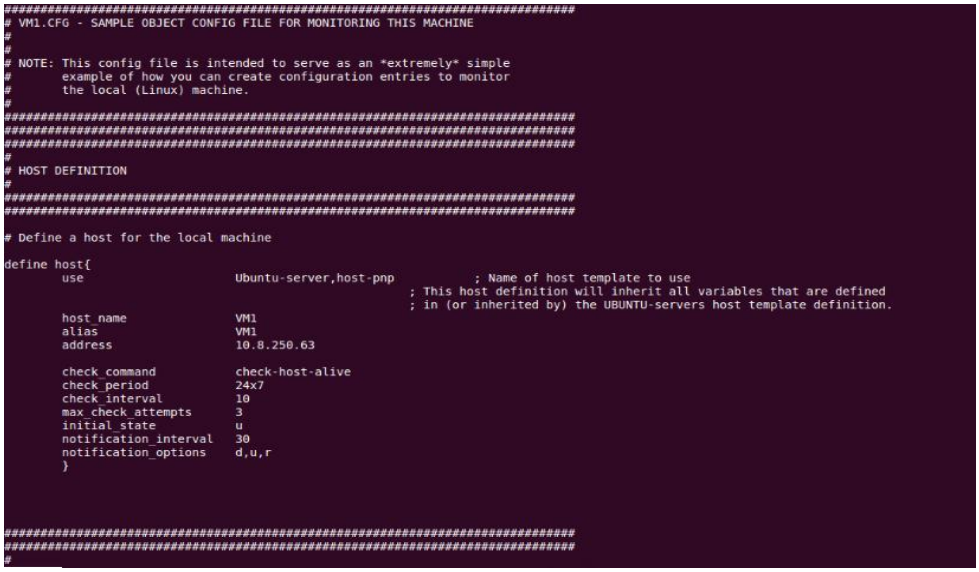


Fig.5.3 Output of AddVM.py



File.5.4 New VM.cfg file added in Nagios.cfg

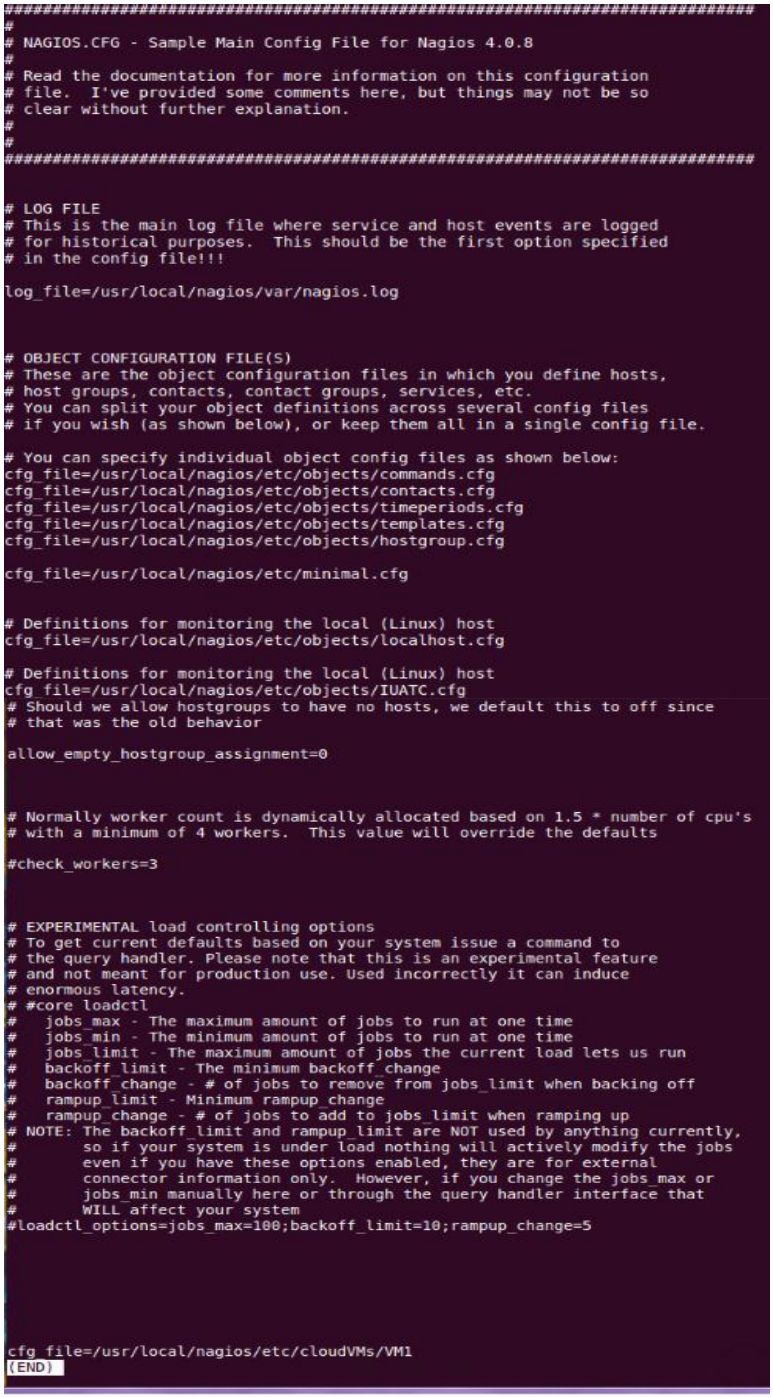


Fig.5.5 Output of the file Nagios.cfg

4.4 Networked Sensor Connection

The following changes were made in the file “BB1.cfg”:

```
define service{
use                          generic-service-graphed ;
Name of template to use
host_name                          BB1
```

```
check_interval                  2
service_description             Temperature_TMP36
check_command                   check_ssh_temp1
}

define service{
use                         generic-service-graphed ;
Name of template to use
host_name                       BB1
check_interval                  2
service_description             Temperature_HTU21D
check_command                   check_ssh_temp2
}

define service{
use                             generic-service-
graphed ; Name of t$
host_name                       BB1
check_interval                  2
service_description             Humidity_HTU21D
check_command                   check_ssh_humidity2
}

define service{
use                             generic-service-
graphed ; Name of template to use
host_name                       BB1
check_interval                  2
service_description             Temperature_DS18B20
check_command                   check_ssh_temp3
}
```

In commands.cfg, this is the following change:

```
#check_ssh_temp1 command definition
define command{
command_name                check_ssh_temp1
command_line                $USER1$/check_by_ssh -H
10.8.8.38 -i /home/Nagios/.ssh/id_rsa -C "sh
/home/Nagios/gettemp.sh" -E
}

#check_ssh_temp2 command definition
define command{
command_name                check_ssh_temp2
command_line                $USER1$/check_by_ssh -H
10.8.8.38 -i /home/Nagios/.ssh/id_rsa -C "sh
/home/Nagios/test/OpenPythonSensor/lib_htu21d/checkt
emp.sh" -E
}

#check_ssh_humidity2 command definition
```

```
define command{
command_name                    check_ssh_humidity2
command_line                    $USER1$/check_by_ssh -H
10.8.8.38 -i /home/Nagios/.ssh/id_rsa -C "sh
/home/Nagios/test/OpenPythonSensor/lib_htu21d/checkh
umidity.sh"-E
}

#check_ssh_temp3 command definition
define command{
command_name                    check_ssh_temp3
command_line                    $USER1$/check_by_ssh -H
10.8.8.38 -i /home/Nagios/.ssh/id_rsa -C "sh
/home/Nagios/digtemp.sh" -E
}
```

4.4.1 Screenshots of the Networked Sensor Connection

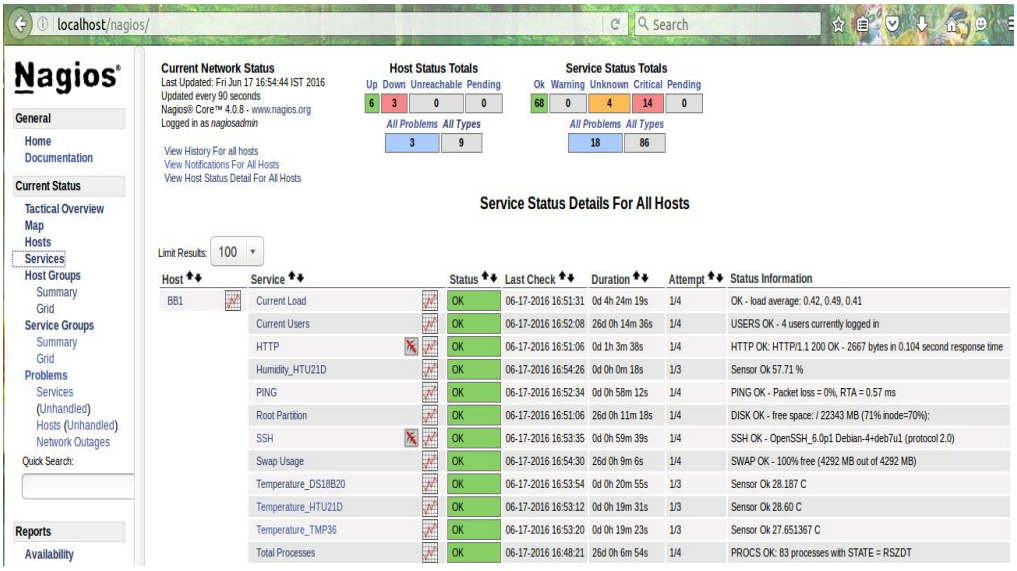


Fig.5.6 Status of the Networked BB1 (BeagleBone) and the services defined in it

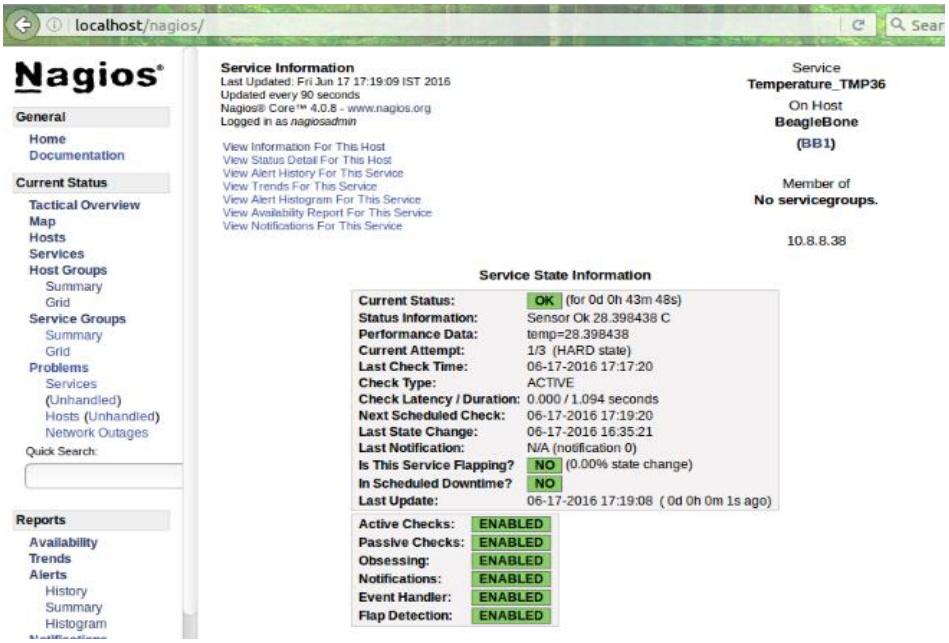


Fig.5.7 Temperature_TMP36 Service's Status

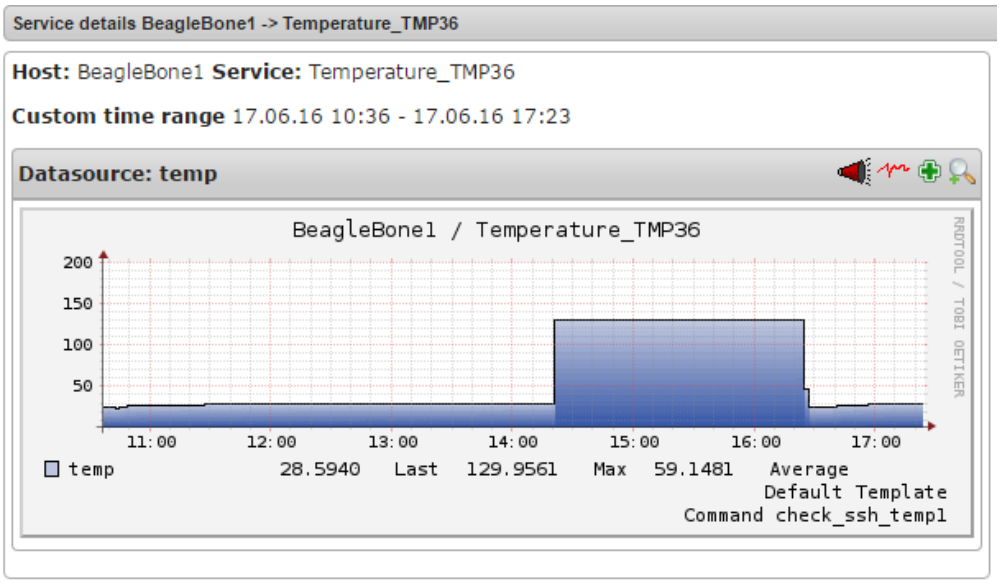


Fig. 5.8 Temperature_TMP36's Graph for six hours

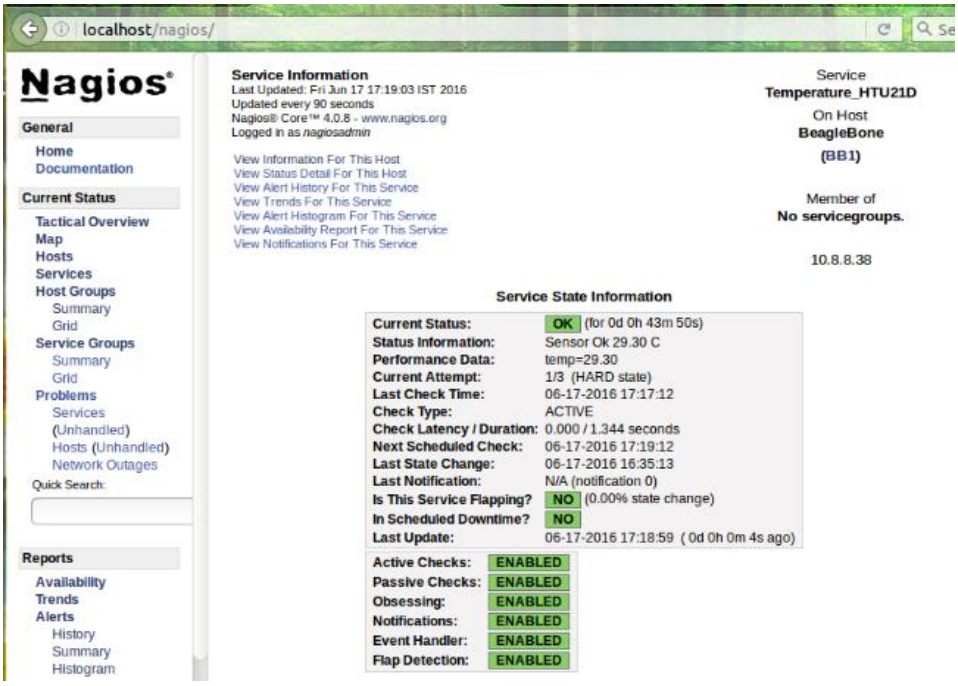


Fig.5.9 Temperature_HTU21D Service's Status

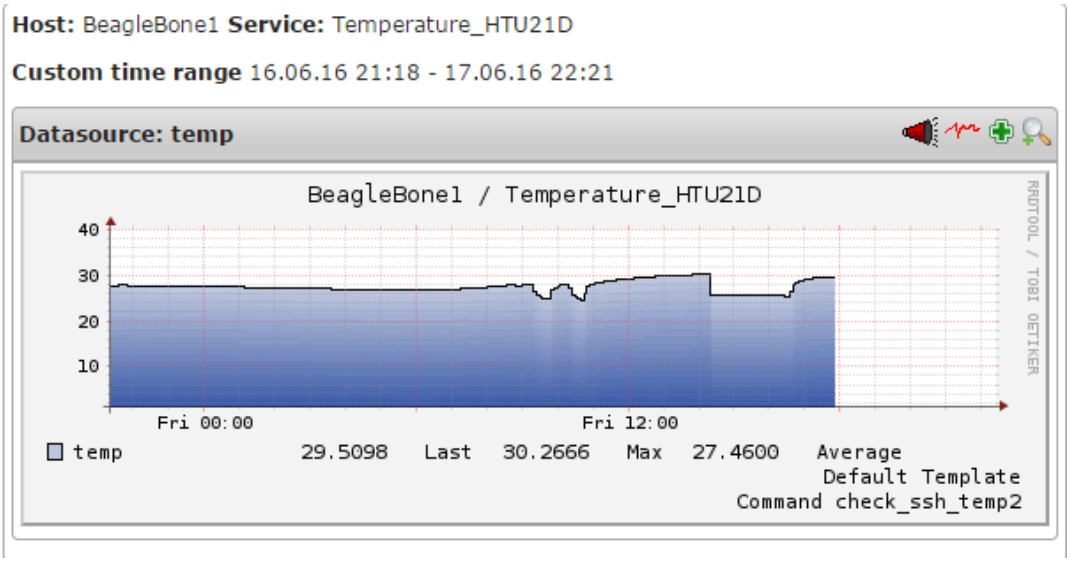


Fig.6.0 Temperature_HTU21D's Graph for one day

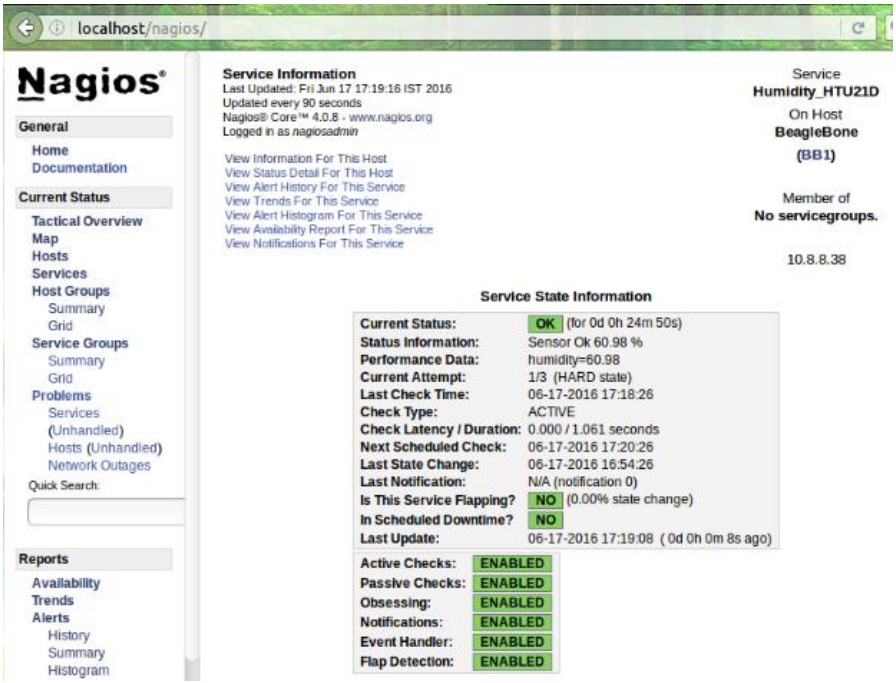


Fig.6.1 Humidity_HTU21D Service's Status

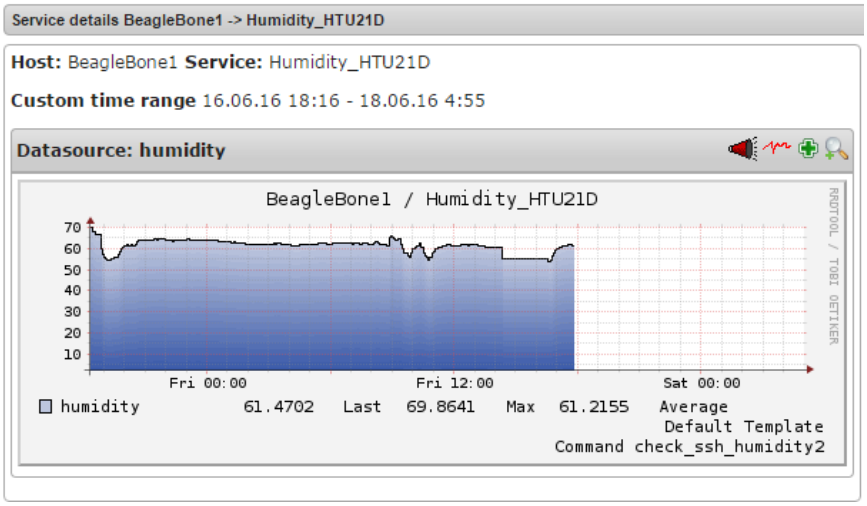


Fig.6.2 Humidity_HTU21D Graph's for two days

4.5 Standalone Sensor Connection

Created a new host "Director's Residence.cfg" and have defined two temperature services (HTU21D, TMP36) and one humidity service (HTU21D).

```
define service{
use                             generic-service-graphed
host_name                       DirectorResidence
check_interval                  3
service_description             TempTMP36
check_command                   check_tempT1
}
define service{
```

```
use                     generic-service-graphed
host_name               DirectorResidence
check_interval          3
service_description     TempHTU21D
check_command           check_tempT2
}
define service{
use                     local-service,srv-pnp
host_name               DirectorResidence
check_interval          3
service_description     HumidHTU21D
check_command           check_HumidH2
}
```

In commands.cfg;

```
define command{
command_name            check_tempT1
command_line            $USER1$/check_sensors -H
$HOSTADDRESS$ -p $ARG1$ $ARG2$
}
define command{
command_name            check_tempT2
command_line            $USER1$/check_sensors -H
$HOSTADDRESS$ -p $ARG1$ $ARG2$
}
define command{
command_name            check_HumidH2
command_line            $USER1$/check_sensors -H
$HOSTADDRESS$ -p $ARG1$ $ARG2$
}
```

I got the readings from the Beagle Bone setup deployed on the server in the CNRG Lab and created all four databases and xml files because check_sensors is actually the script for checking the system's laptop. After getting the readings file, I update the database and hence the graph shows up.

These are the four "UpdateReading.py" files which then update the database because there are four services and hence four rrd databases.

UpdateReadingTMP36.py:

```
#!/usr/bin/env python
import os
f = open("UpdateReadingTMP36.txt", "r")
for line in f:
command = "sudo ./ImportfileTMP36.py " + line
os.system(command)
```

UpdateReadingT2.py:

```
#!/usr/bin/env python
import os
f = open("UpdateReadingT2.txt", "r")
for line in f:
command = "sudo ./ImportfileT2.py " + line
os.system(command)
```

UpdateReadingH2.py:

```
#!/usr/bin/env python
import os
f = open("UpdateReadingH2.txt", "r")
for line in f:
command = "sudo ./ImportfileH2.py " + line
os.system(command)
```

These are the four “import_tempfile.py” files:

ImportfileTMP36.py

```
#!/usr/bin/python
import sys
import os
import rrdtool
file_name = sys.argv [1]
line = []
data = [line.strip() for line in open(file_name,
'r')]
temp = []
for i in data:
temp = i.split()
string = str(temp[1]) + ":" + str(temp[0])
command = "sudo rrdtool update TempTMP36.rrd" +
string
os.system(command)
```

ImportfileT2.py:

```
#!/usr/bin/python
import sys
import os
import rrdtool
file_name = sys.argv [1]
line = []
data = [line.strip() for line in open(file_name,
'r')]
temp = []
for i in data:
temp = i.split()
string = str(temp[1]) + ":" + str(temp[0])
command = "sudo rrdtool update TempHTU21D.rrd" +
```

```
            string
            os.system(command)

ImportfileH2.py

            #!/usr/bin/python
            import sys
            import os
            import rrdtool
            file_name = sys.argv [1]
            line = []
            data = [line.strip() for line in open(file_name,
            'r')]
            temp = []
            for i in data:
            temp = i.split()
            string = str(temp[1]) + ":" + str(temp[0])
            command = "sudo rrdtool update HumidHTU21D.rrd " +
            string
            os.system(command)
```

4.5.1 Screenshots of Standalone Sensor Network

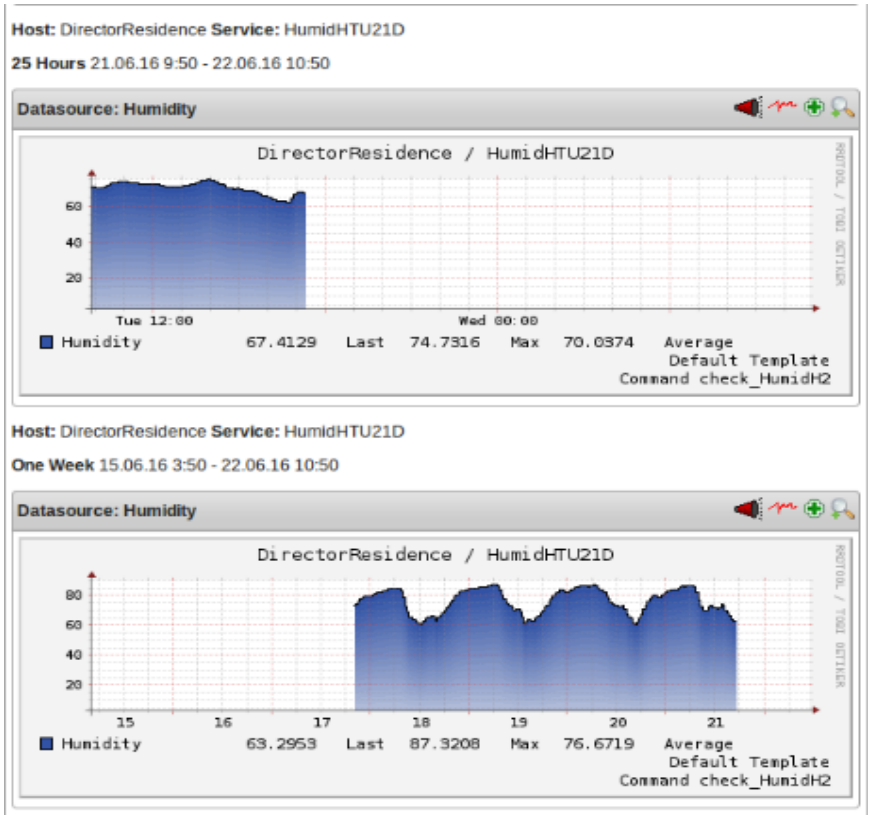


Fig.6.3 Humidity Graph by sensor HTU21D on host DirectorResidence

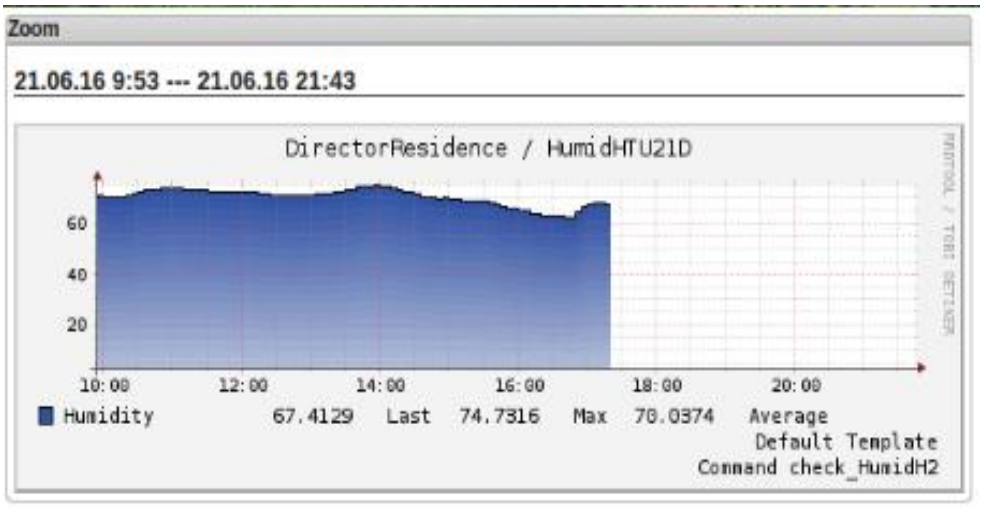


Fig.6.4 Expanded form of Humidity Graph on host DirectorResidence for the last twenty hours

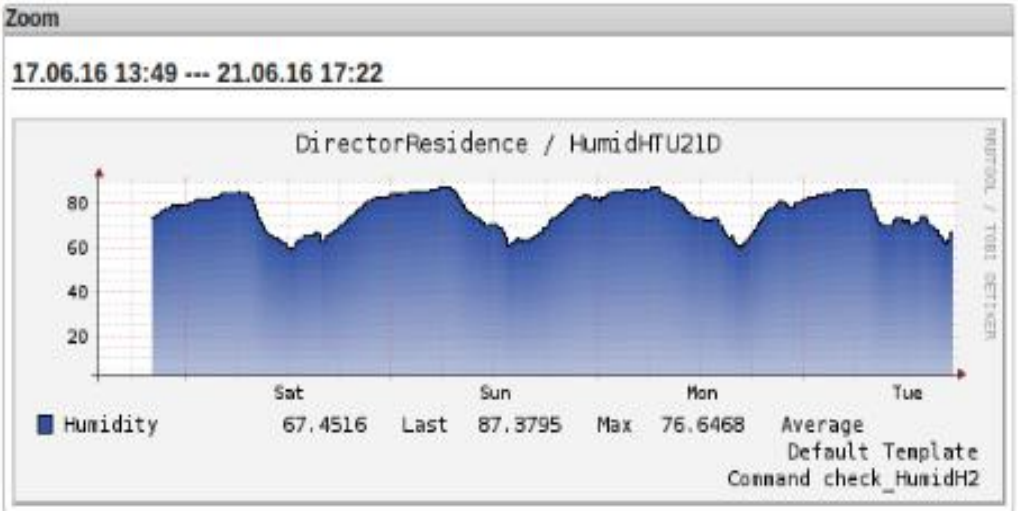


Fig.6.5 Expanded form of Humidity Graph on host DirectorResidence for one week

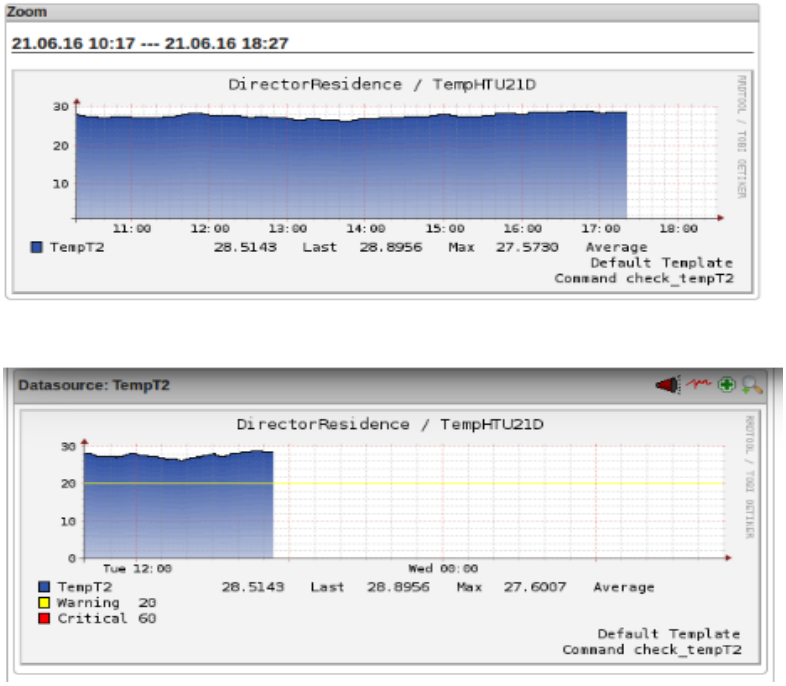


Fig.6.6 Temperature graph by sensor HTU21D on host DirectorResidence

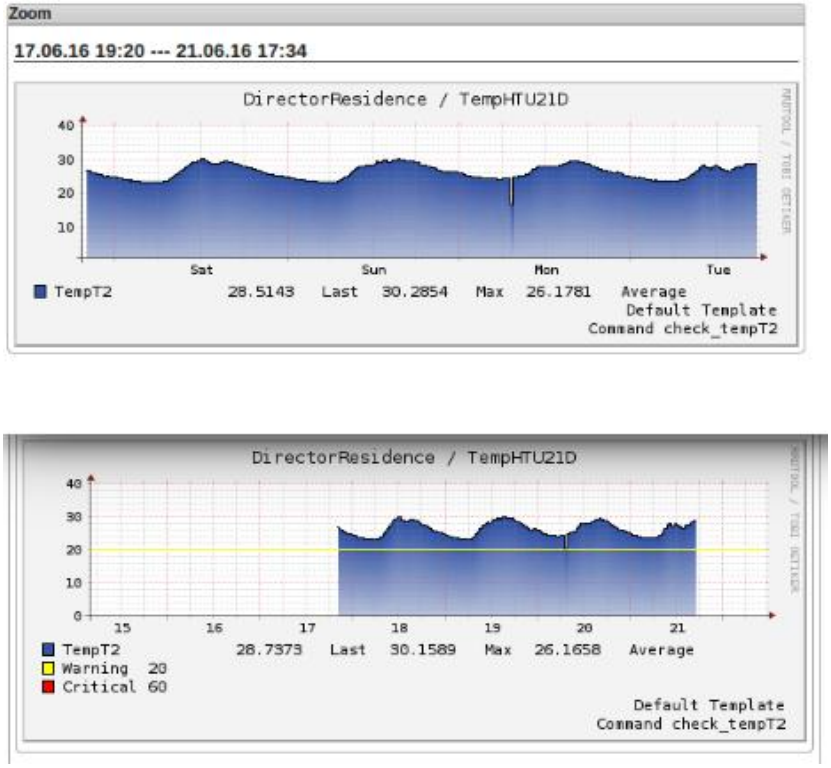


Fig.6.7 Expanded form of Temperature Graph on host DirectorResidence for one week

Fig.6.8 Graph of TempHTU21D on host DirectorResidence for the Last four hours

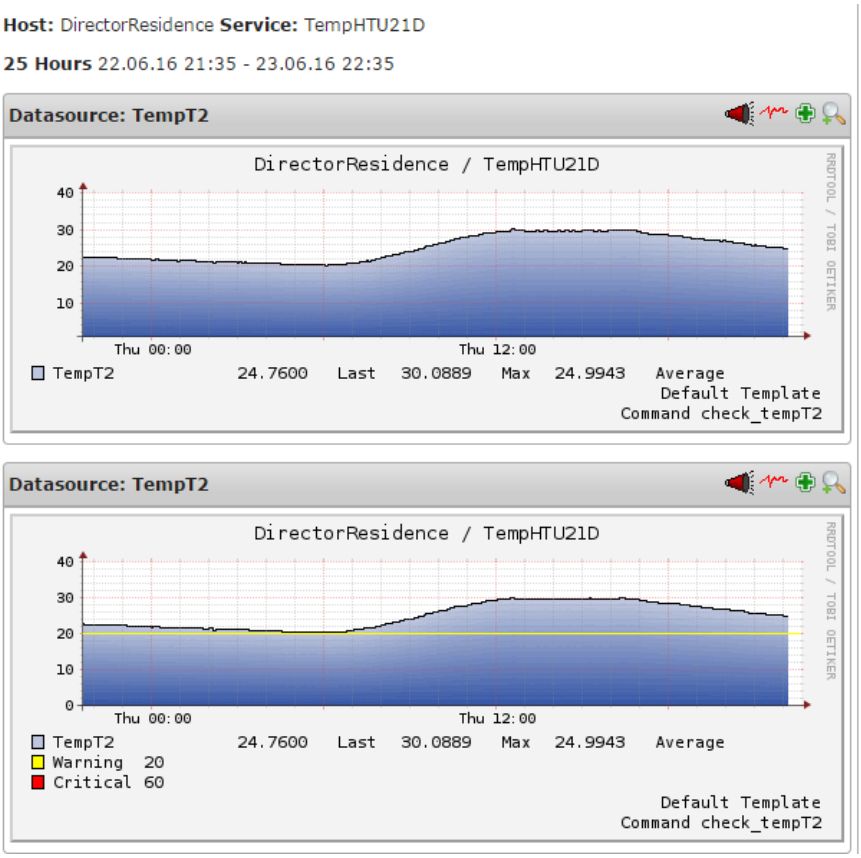


Fig.6.9 Graph of TempHTU21D on host DirectorResidence for the Last twenty five hours

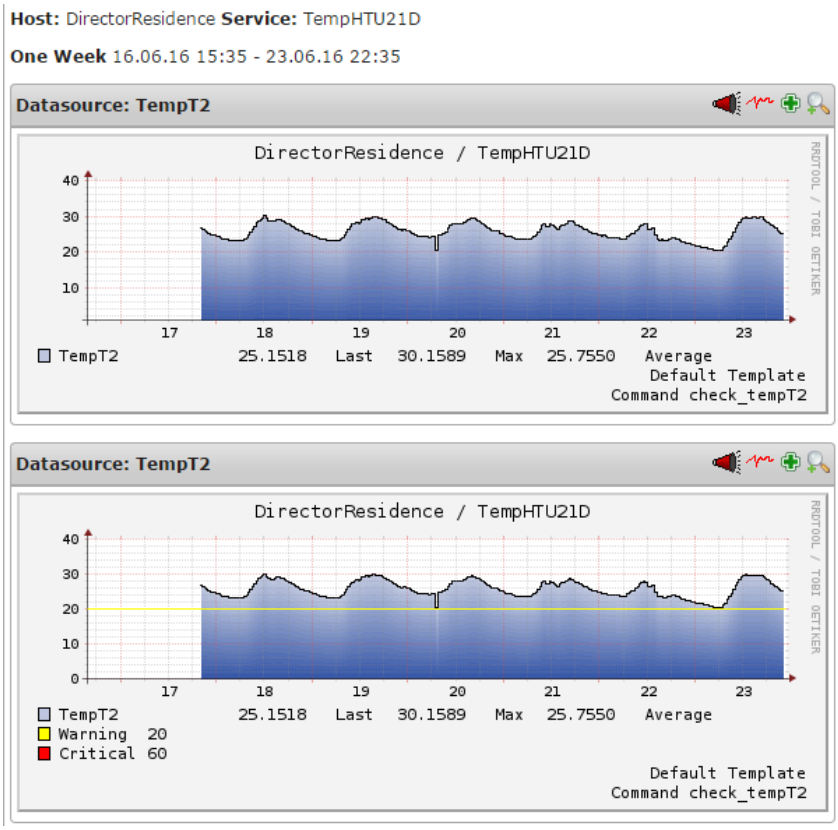


Fig.7.0 Graph of TempHTU21D on host DirectorResidence for the Last One week

Chapter 5 Summary and Conclusions

5.1 Summary

It is quite an interesting project and I got to learn a lot of new things.

- Installed Nagios in my own laptop and on redmine server in the CNRG Lab
- Installed pnp4Nagios and configured it with Nagios and CloudStack
- Set up automatic host discovery setup
- Integrated Nagios with Multiple Sensors

5.2 Conclusions

We understood the need of network management whose basic function is operations, administration, maintenance and provisioning of network and services. It ensure the customers are happy with their service and they get whatever they demand or expect. For this goal, the management establish a policy for contracting an SLA with the users.

Nagios is an excellent monitoring tool which alerts us when something goes wrong or when a host or service becomes OK after recovering from the down state. We can see it is quite easy to integrate Nagios with pnp4Nagios, CloudStack and sensors. We can configure Nagios in our own way provided we write a correct plugin. These plugins are only helpful for making Nagios run. Nagios give us a correct information about the host we monitor. For eg, if we want to know how many instances are running on CloudStack, we write a plugin in Nagios and hence we will get the same answer as in actual. In this way, integration of Nagios and CloudStack is not a difficult task if we do it carefully. Similarly, Nagios was integrated with pnp4Nagios and we get good graphs which actually shows the values e.g., cpu utilization, swap usage or number of users etc.

Integration of Nagios with sensors was too not a difficult task but it took a lot of time for me because of lack of proper understanding. I also learned about rrdtool usage and it basics. I used rrdtool for creating a database and how to update data into it. For logging into beagle bone, I followed ssh passwordless login which helped in getting the values from the beagle bone. We have also used multiple sensor deployment and integrated it with Nagios. There are two types of sensor connection i.e. Networked and Standalone Connection. The standalone sensor setup is kept at Director's Residence which keeps on collecting data from the surroundings while the networked sensor setup was deployed on the server in the CNRG Lab for which I logged into the machine and hence monitored the sensors via Nagios. Screenshots have been added for all the services and hosts' graph in the report.

I would like to conclude that working in this project was a great fun. We can see that Nagios can simultaneously monitor both sensors as well as CloudStack. We can see that sensors are important for giving information about the weather (temperature and humidity) and Nagios perfectly displays that in the form of the graph.

5.3 Future Work

Nagios is very useful for showing the graphs and by seeing graphs, we can get a good idea about the variations in temperature and humidity. Right now, it has been integrated with CloudStack and Sensor but in future it can be configured with actual agricultural network having clouds and sensors. The weather data can be used for the prediction of the future weather as in the case of standalone sensor network. The co-relation of agricultural production with the past weather condition can be easily understood with the help of the weather data. For eg. for growing of apples in the farm, it require high temperature and hence the production is more. Similarly if the production is not that high, we can get to know the reasons for the poor production which can be inferred from the weather data.

References

[1] James F. Kurose, Keith W. Ross. "Network Management" in *Computer Networking: A top-down Approach Featuring the Internet*, 3rd ed, Kurose, Ross, Ed. Dorling Kindersley. Kundli, 2006, pp.729.

[2] *"Simply Easy Learning by tutorialspoint.com: Cloud Computing-an overview",* Available from http://www.tutorialspoint.com/cloud_computing/cloud_computing_tutorial.pdf. (Accessed: 3rd November, 2015).

[3] "Temperature Sensor- TMP 36" [Online]. Available: https://www.sparkfun.com/products/10988 [Accessed: 28-04-2016]

[4] Nate, *"HTU21D Humidity Sensor Hookup Guide".* [Online] Available: *https://learn.sparkfun.com/tutorials/htu21d-humidity-sensor-hookup-guide* [Accessed:15th June,2016]

[5] "*Waterproof DS18B20 Digital temperature sensor + extras".* [Online]. Available: https://www.adafruit.com/product/381 [Accessed:10th June,2016]

[6] S. Shete, T.A. Gonsalves & D. Jalihal "Image Analysis for Network based Agri Advisory System" in National Conference on Communications, IIT Guwahati, 5th March, 2016.

[7] Afeez Abiola Yusuff (May, 2012) "*NETWORK MONITORING: Using Nagios as an Example Tool"*, CENTRAL OSTROBOTHNIA UNIVERSITY OF APPLIED SCIENCES (Accessed: 27th February, 2016)

[8] "*CloudStack Documentation Release 4.5.1: Introduction", Retrieved from* https://media.readthedocs.org/pdf/CloudStack/latest/CloudStack.pdf. (Accessed: 3rd November, 2015)

[9] "CloudStack Documentation Release 4.5.1: Introduction", Retrieved from https://CloudStack.apache.org/ (Accessed: 27th October, 2015).

[10] Jayneil Dalal, "*Beagle Bone- Hands on Tutorial*",2013. [Online]. Available: http://elinux.org/images/b/b7/Beaglebone_-_Hands_on_Tutorial.pdf. [Accessed: 23-Apr-2016]

[11] Brian Proffitt (September 19, 2013), "What APIs Are And Why They're Important", (Accessed: 18th June, 2016)

[12] Ramesh Natarajan (November 20, 2008), "3 Steps to Perform SSH Login Without Password Using ssh-keygen & ssh-copy-id". [Online]. Available: http://www.thegeekstuff.com/2008/11/3-steps-to-perform-ssh-login-without-password-using-ssh-keygen-ssh-copy-id. [Accessed: 23-06-2016]

[13] Ben Rockwood, "Getting Started with RRDtool", 2004. [Online]. Available: http://www.cuddletech.com/articles/rrd. [Accessed: 23-Apr-2016]

[14] Ian Stokes-Rees, "Grid Monitoring using Nagios and RRDtool", in Hepsysman Conference, 29th April, 2003

[15] Chris Burgess. 2005, "The Nagios Book", Online: http://Nagiosbook.org/PRE-RELEASE_The_Nagios_Book-05012006.pdf, Accessed on 22nd June, 2015.

[16] Nagios Configuration. https://geekpeek.net/Nagios-configuration/, Geekpeek.net, 30th April 2013, Accessed on 22nd June, 2015.

[17] Jason Hancock (8 march, 2012) *Nagios-CloudStack*, Available: https://github.com/jasonhancock/Nagios-CloudStack (Accessed: September, 2015)

[18] Jason Hancock (13th Oct, 2012) *CloudStack-php-client*, Available: https://github.com/qpleple/CloudStack-php-client (Accessed: September, 2015)

[19] Zorkian (24th May) *Nagios-api*, Available: https://github.com/zorkian/Nagios-api (Accessed: 30th September, 2015)