

Monitoring and Deployment in Cloud Infrastructure

Project report submitted in partial fulfillment of the requirement for the degree

Of

Bachelor of Technology

In

COMPUTER SCIENCE & ENGINEERING

By

Abhishek Jain (161212)

Under the supervision of

Mr. Anoop Kumar

To



Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Wagnaghat, Solan-173234,
Himachal Pradesh

Certificate

I hereby declare that the work presented in this report entitled “**Monitoring and Deployment in Cloud Infrastructure**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from February 2020 to May 2020 under the supervision of **Mr. Anoop Kumar** (Manager, Cloud Operations).

The matter embodied in the report has not been appeased for the award of any other degree or diploma.



Abhishek Jain (161212)

This is to certify that the above affirmation made by the candidate is true to the best of my knowledge.



Mr. Anoop Kumar

Manager, Cloud Operations

Dated: 23rd May, 2020

Acknowledgement

I would like to take the opportunity to thank and express my deep sense of gratitude to my project guide Mr. Anoop Kumar and my mentors Mr. Ketan Bhardwaj and Ms. Megha Taneja for their immense support and valuable guidance without which it would not have been possible to reach this stage of my final year project.

I am also obliged to all my faculty members for their valuable support in their respective fields which helped me in reaching at this stage of our project. My thanks and appreciations also go to the colleagues who have helped me out with their abilities in developing the project.



Abhishek Jain
161212

Date: 23rd May, 2020

Table of Contents

Certificate	2
Acknowledgement	3
Table of Contents	4
Abstract	9
Chapter 1. Introduction	9
1.1 Introduction.....	10
1.2 Problem Statement	10
1.3 Objectives.....	11
1.4 Methodology.....	11
1.4.1 Network Utility used in management of resources.....	11
1.5 Organization of the thesis	13
Chapter 2. Literature Survey.....	14
2.1 Introduction.....	15
2.2 Monitoring in Cloud Infrastructure.....	15
2.2.1 Nagios for Monitoring.....	15
2.2.1.1 Features of Nagios.....	15
2.2.1.2 Components of Nagios.....	16
2.2.1.3 Nagios Configuration Files.....	16
2.2.1.4 Host Checks in Nagios.....	17
2.2.1.5 Services Checks in Nagios.....	18
2.2.1.6 Different type of checks in Nagios.....	18
2.2.2 Prometheus for Monitoring.....	19
2.2.2.1 Features of Prometheus.....	20
2.2.2.2 Components of Prometheus.....	20
2.2.2.3 Alerting in Prometheus.....	21
2.2.3 Pagerduty for Monitoring.....	22
2.3 Deployment in Cloud Infrastructure.....	22
2.3.1 Ansible for automation and Deployment.....	22
2.3.1.1 Components of Ansible.....	23

2.3.2 Jenkins for automation and Deployment.....	24
Chapter 3. System Development.....	26
3.1 Introduction to PostgreSQL.....	27
3.2 Introduction to Python.....	27
3.3 Introduction to Network Utilities.....	28
3.3.1 Wireshark.....	28
3.3.2 Ping network utility.....	30
3.3.3. Traceroute network utility.....	31
3.3.4 Netcat network utility.....	32
3.4 System Requirements.....	34
3.4.1 Software Requirements	34
3.4.2 Hardware Requirements.....	35
3.5 Conclusion.....	35
Chapter 4. Performance Analysis.....	36
4.1 Introduction.....	37
4.2 Nagios of Monitoring.....	37
4.3 Ansible for Management.....	40
Chapter 5. Conclusion & future work.....	42
5.1 Conclusion.....	43.
5.2 Future work.....	43
References.....	44

List of Figures

Fig 1.1	Working of SSH	12
Fig 2.1	Nagios Architecture	17
Fig 2.2	Nagios Plugin result	18
Fig 2.3	Nagios active checks process	19
Fig 2.4	Nagios Passive checks process	19
Fig 2.5	Prometheus Architecture	20
Fig 2.6	Alertmanager Sample Template	21
Fig 3.1	Wireshark packet capture - I	29
Fig 3.2	Wireshark packet capture - II	30
Fig 3.3	Ping output	31
Fig 3.4	Traceroute output	32
Fig 3.5- 3.6	Netcat server-client results	32
Fig 3.7	Get command using netcat	33
Fig 3.8	Checking open ports using netcat	34
Fig 4.1	Sample host file for Nagios	37
Fig 4.2	Sample services file for Nagios	38
Fig 4.3	Nagios status	38
Fig 4.4	Nagios UI for host checks	39
Fig 4.5	Nagios UI for service checks	39
Fig 4.6	Ping module in Ansible	40
Fig 4.7	Custom playbook in Ansible	40
Fig 4.8	Playbook Execution results	41

ABBREVIATIONS

SFTP	:	Secure File Transfer Protocol
FTP	:	File Transfer Protocol
SSH	:	Secure Shell
SMTP	:	Simple Mail Transfer protocol

ABSTRACT

Cloud Computing has gained a lot of interest in recent time. Now-a-days any resource could be accessed from anywhere in the world. Moreover services can be hosted online as well. The number of users accessing the internet is increasing day by day. With the increase in number of users, the requests per second made by the users for a particular is increasing. To handle those requests, the hosts hosting the particular services are also increasing. But those hosts need to be monitored and configured as well. With the large number of hosts, the monitoring and management process becomes tough and to solve this problem, tools are required which can automate the process up to some extent. In this project we will be discussing some tools which can be used for monitoring and management purposes.

CHAPTER- 1

INTRODUCTION

CHAPTER: 1 INTRODUCTION

1.1 Introduction

Nowadays, everyone is using different resources present at different corners of the world. People are using various services like social networking, online shopping, movies that are hosted by different servers around the world. Moreover people are now moving their local data to cloud platforms for better storage, better accessibility and for various services available online. People can even host their services without even using their own hardware architecture. But the question arises how this is all possible. This is possible just because of the concept of cloud computing. Hence, the concept of cloud computing is quite popular in recent years.

With the increase with users, the servers hosting particular services are increasing as well. But it causes a problem in management of servers. For a big cloud architecture, two things are always critical i.e. monitoring and deployment. The servers need to be consistently monitored because if there is some problem then all the services will go down and it will cause a lot of problems for the company as well as the customers. Moreover the servers present all over the world need to be frequently upgraded, which may be a security patch or a software upgrade.

To monitor and manage the cloud, we need some tools that can perform the task in real time without affecting the services hosted on the cloud. If we have some tool that can notify the problem in the real time, then the proper steps could be taken and the customers using the services will not be affected much. Imagine a situation when a problem going on in the cloud may not be notified in the real time, or there is some problem that the security patch could not be made in the real time, then it can cause a lot of trouble for the customers as well as for the Company.

To solve these kinds of problems we need proper tools that can fulfill our requirements.

1.2 Problem Statement

Users using the services available online are increasing day by day due to which large number of resources are to be made available to fulfill the requirement. But with the increase in expansion of resources, the management of resources has become a tedious task. The resources should be available at real time and to ensure this we need tools that can help in managing the resources that are present at different geographical locations.

With the help of various networking tools, we can manage the resources in real time from anywhere in the world. The aim would be to learn about different tools and technologies that can help us to manage the resources in real time.

1.3 Objectives

The main objective of the project would be to know the different tools and technologies available that can help us to manage the resources in real time and with greater efficiency.

It will be a process of the few steps:

- A. Collecting the information about relevant tools and technologies from online resources.
- B. Implementing the tools to check the features.
- C. Deciding what tools can fulfil our requirement.

1.4 Methodology

A local setup will be made with the help of various operating systems running on virtual machines. This will help us to test different technologies and their uses. With the help of various tools and technologies, we will try to monitor and manage various hosts. We will see how those tools are working and what networking utilities are behind them which is helping us to manage our resources.

1.4.1 Network utility used in management of resources

The SSH (Secure Shell) is a software package that is used to make a connection with the remote host over an insecure channel. It is used almost everywhere to manage the remote devices. With the help of encryption, the SSH secures the channel between the client and the server. It

provides various options for authentication with message security and integrity. It is a better alternative than telnet and FTP which were insecure. SFTP is a secure file transfer protocol that is used to transfer files in an encrypted manner.

It is used for different purposes like:

- A secure channel to access resources for users and automated process
- Automatic file transfer
- executing commands on remote host
- managing network devices

Working of SSH protocol:

SSH works on the client-server architecture. This means that the connection is established by the client to the server. With the help of public key cryptography, the identity of the server is identified. After the identification phase, using strong encryption and hashing algorithms, a secure channel is created between the client and the server. And then the data can be exchanged. The figure below describes the whole process:-

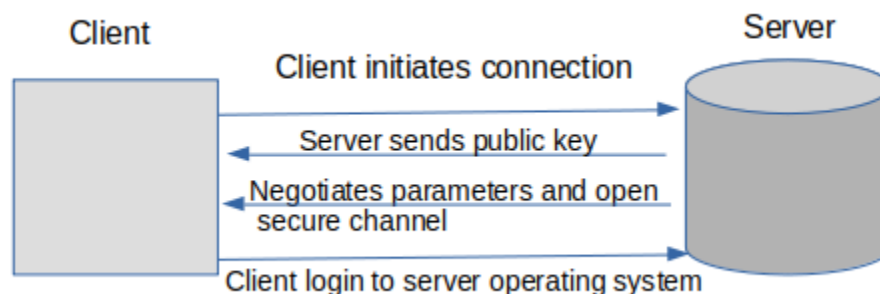


Fig 1.1 represents how SSH works

Authentication with SSH keys

For automation and single-sign on, public key infrastructure method is mainly used. In this there is a public-private key pair which is used for authentication. The public key is present on the server, and anyone having the private key can access the server. This is mainly used for secure automation. The files can be automatically transferred or commands could be executed

on the remote host with the help of this. The whole process could be automated and the systems can be configured automatically.

In large organizations, there are large number of SSH keys, therefore to manage these keys is an important task. SSH keys perform the similar functions as usernames and password do.

After the connection phase, the data can be transmitted over secure channel without the possibility of the man-in-the-middle attacks. The data is encrypted by the parameters negotiated during the setup phase. The data is encrypted using the industry standard encryption algorithms like AES, and it ensures integrity using hash algorithms like SHA-2.

The files can be transferred to the remote host using SFTP. It works using SSH and is most widely used utility around the world.

Examples of SSH clients are:

- OpenSSH for Linux/Unix environment.
- Putty for windows environment.
- CyberDuck for Mac.

1.5 Organization of thesis

The thesis is divided into 5 chapters. The first chapter talks about the problem statement, objectives, and the methodology that is used. The second chapter talks about the literature survey. The third chapter talks about the system development and the fourth chapter talks about the tools that were used for the management of the target hosts. The fifth chapter talks about the conclusion and future work.

CHAPTER- 2

LITERATURE SURVEY

CHAPTER: 2 LITERATURE SURVEY

2.1 Introduction

The cloud computing is in great demand now. The resources can be accessible from anywhere in the world. Moreover the resources are replicated at different geo-locations for better accessibility and low latency. To manage and monitor these resources, there are industry standard tools available with different features. These tools uses the SSH utility to make remote connections and perform the given task. The monitoring and deployment aspects plays an important role in cloud infrastructure.

2.2 Monitoring in Cloud Infrastructure

The Monitoring plays an important role in cloud infrastructure. The servers hosting the resources needs to be continuously monitored. For a big cloud infrastructure, monitoring each and every host is a very difficult task. To ease the monitoring process and to automate this to some extent, there are various tools available i.e. Nagios, Prometheus etc.

2.2.1 Nagios for monitoring

Nagios is an open source tool for monitoring the hosts. It monitors the hosts and services that we specify, and alerts when anything goes bad or even when it comes back to normal state.

2.2.1.1 Features of nagios

- Nagios can monitor various network services like SMTP, POP3, PING, HTTP etc.
- It can monitor host resources as well like memory, CPU usage etc.
- We can design our own plugins as well.
- The task across various hosts are done parallelly.
- Inheritance and hierarchy like features are also available.

- We can also specify contacts to reach when something wrong happens.
- Logs can also be created with log rotation feature.
- A web interface is also available for better visualization.

2.2.1.2 Components of nagios

- Nagios core - It is the primary monitoring and alerting engine.
- Nagios plugins - These are the executable which are responsible for checking the status of the remote host. These are executed on the remote host and alerts the nagios core about the state of the host. There are multiple plugins available for checking services like SSH, SMTP etc.
- Nagios add-ons - These are the additional packaged which extend the functionality of Nagios core.
 - Available add-ons are:
 - The configuration files can be managed from web interface
 - NRPE for monitoring remote host
 - Passive check which are not a part of nagios plugins but provides the states of the remote hosts to the nagios core
 - Notification login could be enhanced

2.2.1.3 Nagios Configuration files

To tell the nagios what hosts are to be monitored, and what should be done if any anomaly happens, the configuration files are needed. Using those file, nagios takes decisions accordingly.

The various files of nagios are:-

- Main configuration file - It contains the directives that affects the operation of nagios core daemon.

- Resource files - It is used to store user defined macros. The sensitive information like passwords are usually stored in resource files.
- Object Definition file - It is used to define services, hosts, contacts, commands etc. It defines what to monitor, whom to contact in anything happens.
- CGI configuration file – It contains the directives for the operations of CGI files.

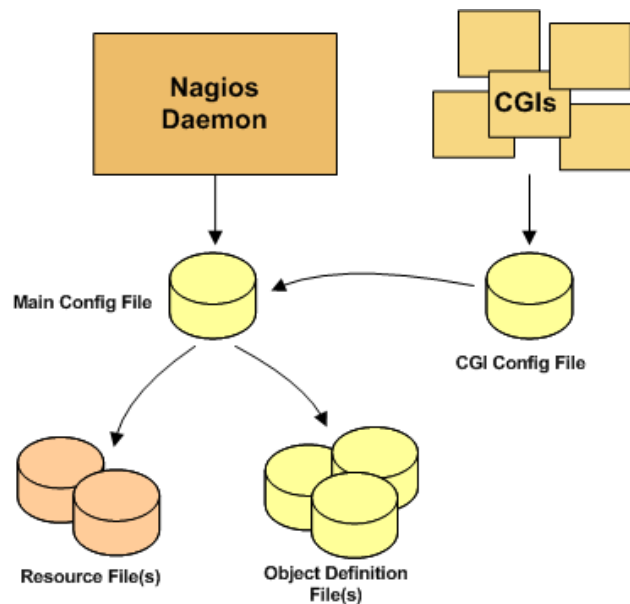


Fig 2.1 describes how the configuration files are used by Nagios daemon.

2.2.1.4 Host Checks in Nagios

The hosts present in the host file are checked at regular interval of time. The interval can be defined by `check_interval` directives present in host definitions. The on-demand checks are scheduled by nagios as well when the service state on the host is changed to determine the reachability of the host. The checks are done parallelly across all the hosts.

There are three host states:-

- Up
- Down

- Unreachable

The plugins that are executed on the hosts determine the state of the host and inform the nagios daemon about the same.

Plugin Result	Preliminary Host State
OK	UP
WARNING	UP or DOWN*
UNKNOWN	DOWN
CRITICAL	DOWN

Fig 2.2 represents the plugin result and the corresponding state of the host.

2.2.1.5 Service Checks

The services hosted on the hosts are checked at regular interval of time. The interval is defined by 'check_interval' and 'retry_directives' in the service definitions. The on-demand service checks are also done by nagios if there is any dependency between the services. The checks are done parallelly. The services that are checked can be any one of the four states mentioned below:-

- OK
- WARNING
- UNKNOWN
- CRITICAL

2.2.1.6 Different types of checks in nagios

- Active checks – The checks which are initiated by nagios core demon are active checks. The plugins are executed by nagios core and it returns the state back to the daemon. If

something wrong is detected, then the daemon will take the required actions and contact the specified receiver.

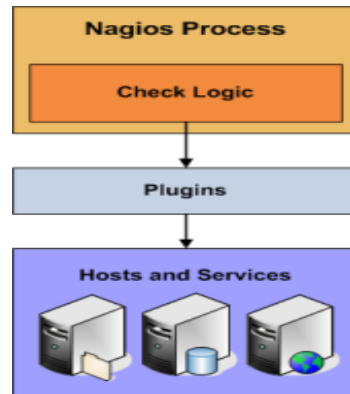


Fig 2.3 represents the process of active checks in nagios.

- **Passive checks** – These checks are initiated and executed by external applications which are usually behind the firewall where nagios is unable to access the hosts. After the checks are done, the application inform the nagios about the state of the host and nagios takes actions accordingly. The only requirement is that the result that are returned should be in the format as specified by nagios.

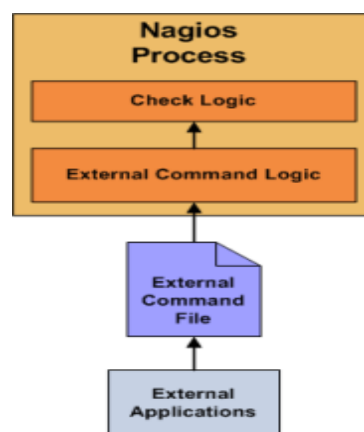


Fig 2.4 represents the passive checks process in Nagios

2.2.2 Prometheus for monitoring

Prometheus is an open source tool that is used for monitoring and alerting purposes. It stores the time series data of target host.

2.2.2.1 The feature of Prometheus's are:-

- a multi-dimensional data model is created with time series data which are identified by key value pairs
- To efficiently use the dimensions, PromQL is there. It is a flexible query language
- There is an intermediary gateway for pushing time series
- With the help of static configurations and service discovery, targets can be identified
- For graphs and dashboard, multiple modes are available which helps in better visualization

2.2.2.2 Components of Prometheus

- Prometheus server – It collects and stores the time series data
- Client libraries – These are responsible for collecting metrics and sending to the server. They can be written in GO, Ruby and python. Other languages are also supported.
- Push Gateway- For short lived jobs a gateway is available for pushing metrics
- Alertmanager – For handling and alerting the alerts
- Various support tools
- Exporters for services like StatsD, Graphite etc.

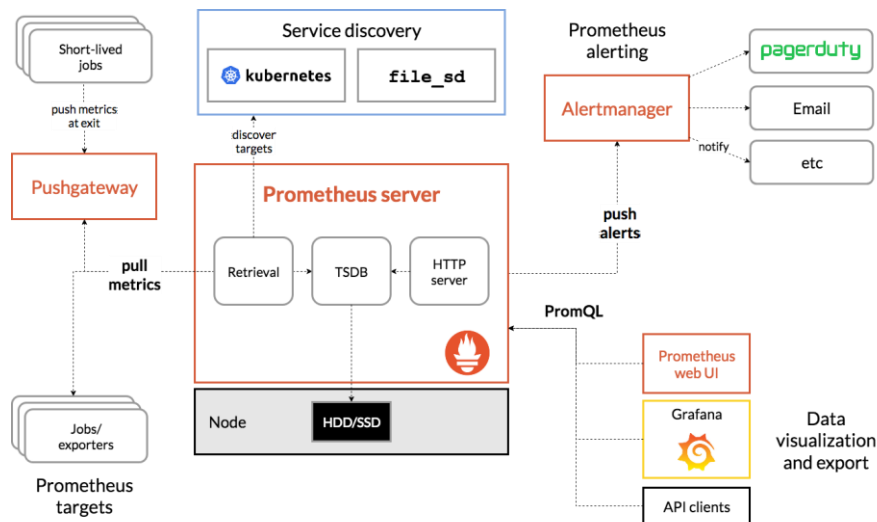


Fig 2.5 represents the architecture of Prometheus

2.2.2.3 Alerting in Prometheus

Alertmanager is a component of Prometheus which is used for handling the alerts. It can group the alerts according to some criteria and can reduce the number of alerts which can further help in visibility and analysis. It also perform other functions like inhibition, silencing and sending the alerts to specified receiver. The receiver can be a chat system, on-call person or mail.

```
global:
  slack_api_url: '<slack_webhook_url>'

route:
  receiver: 'slack-notifications'
  group_by: [alertname, datacenter, app]

receivers:
- name: 'slack-notifications'
  slack_configs:
  - channel: '#alerts'
    text: 'https://internal.myorg.net/wiki/alerts/{{ .GroupLabels.app }}/{{ .GroupLabels.alertname }}'
```

Fig 2.6

Figure 2.6 represents the sample template for the slack notifications. The global configurations are mentioned below the global keyword. The receivers, to which notification has to be sent needs to be mentioned below the receiver's keyword.

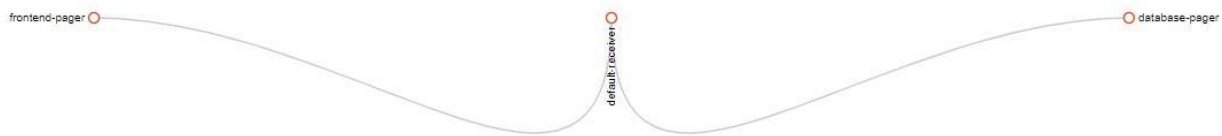


Fig 2.7 Routing tree example

Figure 2.7 represents the routing tree for different receivers. It represents that different routes could be mentioned according to the type of alerts. For e.g. alerts related to frontend application could be routed to frontend pager and the alerts related to database could be routed to backend pager. In this way the alerts could be managed well and resolved faster.

2.2.3 Pagerduty in monitoring

Pagerduty is a tool created to enhance the performance and reliability of businesses by eliminating the chaos across the entire operations lifecycle. It gives powerful capabilities like on-call scheduling, escalation policies, incident monitoring etc. It enables DevOps to deliver better experience to customer. With the help of real-time alerts, DevOps team can quickly act and resolve the problem. It enables users to set up and prioritize the alerts with SMS and voice messages capabilities. For example, if a team member doesn't respond to the email in a timely manner, the alert notification would automatically be sent to alternative team member.

Features of pagerduty are:-

- Reliable alerting
- System and user reporting
- Automated escalations
- Service grouping
- Mobile incident management

2.3 Deployment in cloud infrastructure

The deployment plays an important role in cloud infrastructure because every day there is a change, which may be a security patch or a software update. But when services are already running, deployment of a software package on the servers becomes critical. There are industry standard tools available for deployment and management of hosts like Jenkins, Ansible, bamboo etc.

2.3.1 Ansible for automation and deployment

Ansible is an open source tools that is used for automation. It is a powerful tool and is easy to install. It can be used in configuration management, deployments and automation of tasks. It can also do IT orchestration where tasks are to be executed in sequence on several servers. It uses SSH agent to make connection to remote host for doing the desired task.

2.3.1.1 Components of Ansible

It is a simple tool for automating cloud provisioning, deployments and configuration management. The components of Ansible are:-

- Modules – Ansible make connections to host and executes scripts on them which are called “Ansible modules”. The modules can also accept parameters. Ansible modules could be customized according to the need.
- Module utilities- When different modules uses the same piece of code, these functions can be stored as module utilities to reduce duplication and maintenance.
- Plugins- While modules are executed on the remote host, plugins executes on the control node. It offers various options and extension for the existing features of Ansible. It can help in transforming output to more human readable format, logging etc.
- Inventory- By default Ansible picks the remote host information from the file written in YAML, INI etc. But it can also picks host from sources like Openstack, Rackspace etc.

- Playbooks- To enable orchestration, play books plays an important role. In this you can defines various steps in order to achieve a desired state. Multiple host can be defined for which the tasks are to be executed.

2.3.2 Jenkins for automation and deployment

Jenkins is an open source automation tool written in java with plugins built for continuous Integration purpose. It is used for build and testing software projects continuously that makes easier for developers to integrate the changes to the project. Due to this users always get a fresh build. With the help of Jenkins, the development and testing process could be accelerated through automation. With the help of plugins, Jenkins provides continuous integration. Several tools could be integrated with the help of plugins like HTML publisher, Git etc.

Advantages of Jenkins:

- Open source tool with great community support
- Easy to install
- Large number of plugins available
- Free of cost
- Developed in java, therefore can be used on major platforms

Continuous integration is a development practice in which the developers are required to make changed to code in a shared repository frequently. When a change is made, a new build is made available. It can make testing process easier and efficient.

How Jenkins works?

- Suppose a developer commits a change in the shred repository. At regular interval, Jenkins server checks the repository.

- As soon as a change occurs, the Jenkins detects the change and make a fresh build of the project
- If the build process fails, the concerned team is notified
- If the build process is successful, the builds is installed on the test server
- After testing Jenkins generates the report and notifies the developer
- It will continue the process if any change is notified

CHAPTER- 3

SYSTEM DEVELOPMENT

CHAPTER: 3 SYSTEM DEVELOPMENT

3.1 Introduction to PostgreSQL

PostgreSQL is an open source object-relational database system that uses SQL language and extends its existing features including safely store and it scales the most complicated data workloads.

Features of PostgreSQL:

- User defined data types are available
- Table inheritance
- Locking mechanism
- Nested transactions
- Asynchronous replication
- Views, rules, sub-query

It allows to add custom functions using programming languages like C/C++, java etc. User defined data-types, index types are possible. The user requirements could be filled by using customized plugins.

For a connection to a database:

```
psql -d databse -U user -W
```

If database resides on other hosts then

```
psql -h host -d databse -U user -W
```

3.2 Introduction to python

Python is an open source, high-level and object-oriented language. It has built in data structures, with dynamic binding which makes easy for writing codes in few line as possible. Python syntax is easy to understand and use. Because there is no compilation step involved, the edit-debug-test cycle is fast. Debugging in python is easy because there is a built in debugger.

Features of python:

- Easy to learn and use
- Understandable and readable
- Interpreted language
- Open-source language
- Cross-platform language
- object oriented language
- GUI can be built is well

3.3. Introduction to Network Utilities

There are several network utilities available for troubleshooting the network problems like ping, traceroute, Wireshark etc. These tools can be used to monitor the network connectivity and the problems could be detected easily.

3.3.1 Wireshark

Wireshark is network packet analyzer. It captures the packet and presents in as detail as possible. With the help of details of packet, the network problems could be easily detected. It gives details about all the packets moving from particular interface. It can also be used for learning purposes. By looking at the captured packet, one can have an idea about how the data is transferred between the two devices.

Purpose of Wireshark:

- It is used to troubleshoot the network problems
- It is used to examine security problems
- It can be used to verify network applications
- It can be used to debug network protocols
- It can be used to learn network protocols

Features of Wireshark:

- It is available for both Unix and Windows
- It captures live packet transmission of any network interface
- It can be used for analyzing saved packet captures
- Packet captures can be saved as well
- Packets can be filtered according to some criteria
- Statistics can be developed as well

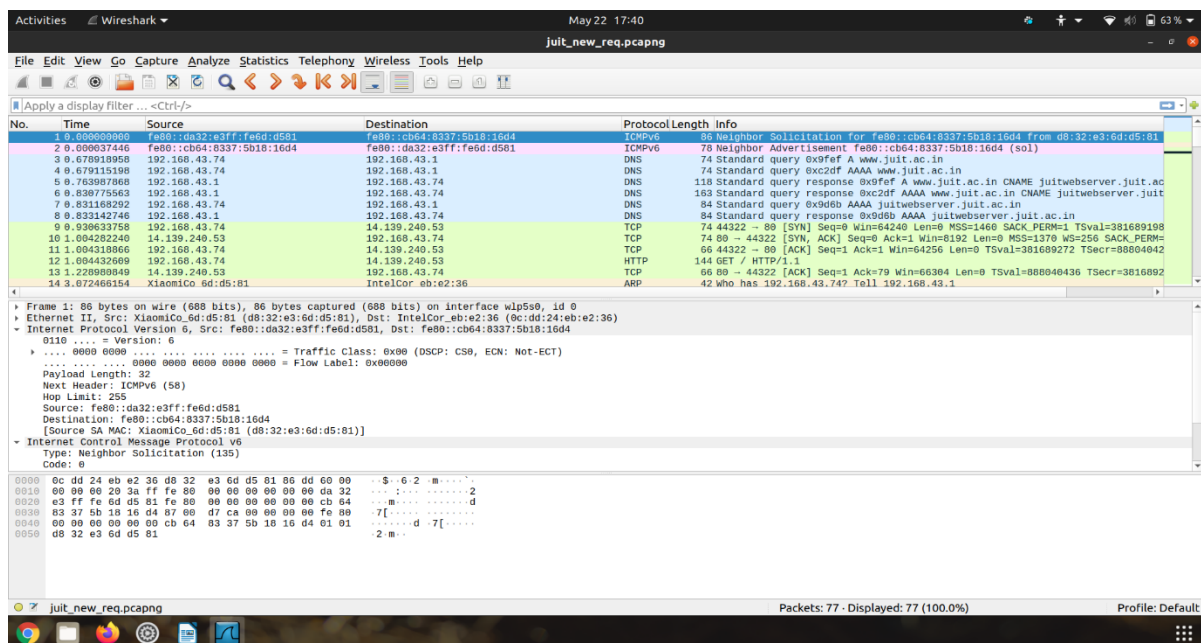


Fig 3.1

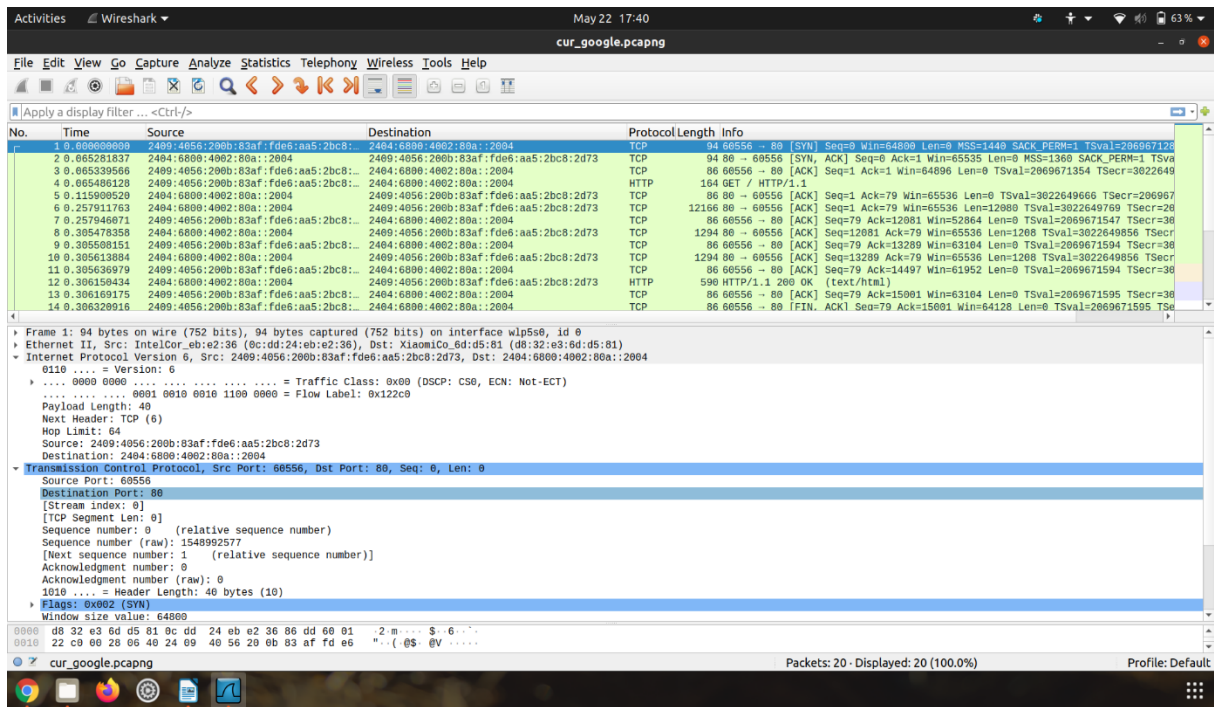


Fig 3.2

Figure 3.1 and 3.1 represents the packet captures of the curl request to www.juit.ac.in and www.google.com respectively. It represents how the request is sent to server and how the data is fetched.

3.3.2 Ping network utility

Ping is a network utility which identifies if the host is reachable or not, certain IP address exists or not. It can be used for troubleshooting network connectivity and calculating response time. It uses ICMP request packet and waits for ICMP response packet which further helps in calculating round trip time.

```

abhishek@abhishek-VirtualBox:~$ ping www.juit.ac.in
PING juitwebserver.juit.ac.in (14.139.240.53) 56(84) bytes of data.
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=1 ttl=46 time=75.8 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=2 ttl=46 time=76.1 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=3 ttl=46 time=85.8 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=4 ttl=46 time=82.5 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=5 ttl=46 time=82.8 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=6 ttl=46 time=88.7 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=7 ttl=46 time=66.1 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=8 ttl=46 time=75.5 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=9 ttl=46 time=84.7 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=10 ttl=46 time=82.7 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=11 ttl=46 time=73.8 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=12 ttl=46 time=76.1 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=13 ttl=46 time=80.5 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=14 ttl=46 time=77.4 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=15 ttl=46 time=72.5 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=16 ttl=46 time=84.2 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=17 ttl=46 time=72.1 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=18 ttl=46 time=99.4 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=19 ttl=46 time=78.7 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=20 ttl=46 time=85.4 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=21 ttl=46 time=85.0 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=22 ttl=46 time=56.7 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=23 ttl=46 time=81.0 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=24 ttl=46 time=80.5 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=25 ttl=46 time=68.7 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=26 ttl=46 time=76.2 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=27 ttl=46 time=76.6 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=28 ttl=46 time=74.8 ms
64 bytes from 14.139.240.53 (14.139.240.53): icmp_seq=29 ttl=46 time=87.5 ms
^C
--- juitwebserver.juit.ac.in ping statistics ---
29 packets transmitted, 29 received, 0% packet loss, time 28043ms
rtt min/avg/max/mdev = 56.786/78.944/99.465/7.817 ms

```

Fig 3.3 represents the output of ping utility

3.3.3 Traceroute network utility

Traceroute is a network utility which is used to trace the path taken by the IP packet. It works on the same principle as of ping, but in addition to that it also shows the connectivity problems between different routers. It can help in identifying the areas where connectivity is poor i.e. latency is high.

```

abhishek@abhishek-VirtualBox:~$ traceroute www.google.com
traceroute to www.google.com (172.217.167.228), 30 hops max, 60 byte packets
 1  _gateway (192.168.43.1)  2.330 ms  2.224 ms  2.651 ms
 2  * * *
 3  10.72.124.26 (10.72.124.26)  51.409 ms  10.72.124.18 (10.72.124.18)  47.367 ms  58.322 ms
 4  172.25.95.17 (172.25.95.17)  47.284 ms  51.298 ms  58.215 ms
 5  172.25.117.75 (172.25.117.75)  47.185 ms  57.965 ms  51.134 ms
 6  172.26.8.247 (172.26.8.247)  58.114 ms  50.866 ms  60.548 ms
 7  172.26.127.34 (172.26.127.34)  60.360 ms  39.947 ms  172.26.127.50 (172.26.127.50)  50.734 ms
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  74.125.32.32 (74.125.32.32)  91.750 ms  91.710 ms  93.447 ms
16  * * *
17  142.250.60.134 (142.250.60.134)  101.258 ms  209.85.248.26 (209.85.248.26)  111.584 ms  108.170.248.177 (108.170.248.177)  112.579 ms
18  108.170.248.187 (108.170.248.187)  119.971 ms  108.170.248.179 (108.170.248.179)  131.838 ms  108.170.248.186 (108.170.248.186)  123.859 ms
19  216.239.54.93 (216.239.54.93)  116.082 ms  108.170.238.147 (108.170.238.147)  118.651 ms  209.85.242.105 (209.85.242.105)  85.057 ms
20  74.125.243.97 (74.125.243.97)  74.016 ms  79.999 ms  172.253.66.107 (172.253.66.107)  90.150 ms
21  172.253.67.91 (172.253.67.91)  86.284 ms  172.253.67.89 (172.253.67.89)  99.611 ms  81.294 ms
22  172.253.67.91 (172.253.67.91)  97.204 ms  172.253.67.89 (172.253.67.89)  102.192 ms  81.066 ms
23  del11s04-in-f4.1e100.net (172.217.167.228)  103.110 ms  172.253.67.91 (172.253.67.91)  80.413 ms  del11s04-in-f4.1e100.net (172.217.167.228)  90.632 ms

```

Fig 3.4 Represents output of traceroute utility

3.3.4 Netcat network utility

Netcat is a network utility which is used for making TCP and UDP connections to the target host. With the help of netcat utility, messages could be shared among hosts. It could be used to check open ports on the target host and the data could be fetched with the netcat as well. For e.g. with Wireshark webpages could be retrieved using http request.

```

abhishek@abhishek-VirtualBox:~$ nc 192.168.43.252 5000
hi abhishek
hi!

```

Fig 3.5

```

abhi@abhi-VirtualBox:~$ nc -l 5000
hi abhishek
hi!

```

Fig 3.6

Figure 3.5 and 3.6 represents the message sharing between two hosts. It is based on a client server architecture. Server waits for the request from the client, once the connection is established, data transfer can take place.

```
abhishek@abhishek-VirtualBox:~$ printf "GET /index.html HTTP/1.1\r\nHost: localhost\r\n\r\n" | nc localhost 80
HTTP/1.1 200 OK
Date: Fri, 22 May 2020 13:01:41 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Fri, 13 Mar 2020 08:01:51 GMT
ETag: "2aa6-5a0b7e1d2de81"
Accept-Ranges: bytes
Content-Length: 10918
Vary: Accept-Encoding
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2016-11-16
    See: https://launchpad.net/bugs/1288690
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">

* {
  margin: 0px 0px 0px 0px;
  padding: 0px 0px 0px 0px;
}

body, html {
  padding: 3px 3px 3px 3px;

  background-color: #D8DBE2;

  font-family: Verdana, sans-serif;
  font-size: 11pt;
  text-align: center;
}

```

Fig 3.7

Figure 3.7 represents how the netcat can be used for fetching web pages using GET request. Netcat can also be used to check open ports on the target host as shown in figure 3.8, which can be used for troubleshooting connection problems as well as can be used for security purposes also.

```

abhishek@abhishek-VirtualBox:~$ nc -z -v 192.168.43.252 60-90
nc: connect to 192.168.43.252 port 60 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 61 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 62 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 63 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 64 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 65 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 66 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 67 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 68 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 69 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 70 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 71 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 72 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 73 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 74 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 75 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 76 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 77 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 78 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 79 (tcp) failed: Connection refused
Connection to 192.168.43.252 80 port [tcp/http] succeeded!
nc: connect to 192.168.43.252 port 81 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 82 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 83 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 84 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 85 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 86 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 87 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 88 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 89 (tcp) failed: Connection refused
nc: connect to 192.168.43.252 port 90 (tcp) failed: Connection refused
abhishek@abhishek-VirtualBox:~$

```

Fig 3.8

3.3 System Requirement

With advancement in technologies, high end processors are needed for applications to run. The servers which host the applications need high hardware specifications so that maximum client requests could be handled without any latency.

3.3.1 Software Requirements

Python is a programming languages which has large number of packages including data structures which makes programming easier. It is supported by most of the tools now a days. I have used python 2.7 to support custom nagios plugins and Ansible modules.

Linux operating system is required for hosting nagios and Ansible. Virtual machines are hosted on Virtual Box. To access Nagios UI, any web browser can be used, and in this experiment Firefox is used.

3.3.2 Hardware Requirements

As mentioned above, hardware plays an important part. For hosting the applications mentioned above, a good memory space and RAM of 4 GB is advised. As hosting an application requires multiprogramming, a processor with 4 or more cores is required.

3.4 Conclusion

Two virtual machines are required, on one monitoring and management tools would be hosted and the other host would be managed by the first node. Nagios and Ansible would be used for monitoring and management purposes. The results are mentioned in the next chapter.

CHAPTER- 4

PERFORMANCE ANALYSIS

CHAPTER: 4 PERFORMANCE ANALYSIS

4.1 Introduction

In this chapter we will see how Nagios and Ansible can be used for monitoring and management purposes.

4.2 Nagios for monitoring

Nagios is a tool that is used for monitoring the hosts and various services running on them. Two virtual machines were setup running on Ubuntu, a Linux distribution. One machine is used for hosting Nagios and the other machine is used for monitoring purpose. Nagios gathers the information about the hosts from the file 'host.cfg' as shown in figure 4.1.



```
abhishek@abhishek-VirtualBox:~$ cat nano /usr/local/nagios/etc/hosts.cfg
cat: nano: No such file or directory
## Default Linux Host Template ##
define host{
name                linux-box                ; Name of this template
use                 generic-host             ; Inherit default values
check_period        24x7
check_interval      1
retry_interval      1
max_check_attempts  10
check_command       check-host-alive
notification_period 24x7
notification_interval 30
notification_options d,r
contact_groups      admins
register            0                        ; DONT REGISTER THIS - ITS A TEMPLATE
}

## Default
define host{
use                 linux-box                ; Inherit default values from a template
host_name           abhishek                 ; The name we're giving to this server
alias               ubuntu-a                 ; A longer name for the server
address             192.168.43.252           ; IP address of Remote Linux host
}
```

Fig 4.1 shows sample host configuration file

The services to be monitored for a particular host should be defined in the file 'services.cfg' as shown in figure 4.2. Custom services could be defined as well if the custom plugin is available.

```

abhishek@abhishek-VirtualBox:~$ cat /usr/local/nagios/etc/services.cfg
define service{
    use                     generic-service
    host_name               abhijain
    service_description     CPU Load
    check_interval          0.1
    check_command            check_nrpe!check_load
}

define service{
    use                     generic-service
    host_name               abhijain
    service_description     Total Processes
    check_command            check_nrpe!check_total_procs
}

define service{
    use                     generic-service
    host_name               abhijain
    service_description     Current Users
    check_command            check_nrpe!check_users
}

define service{
    use                     generic-service
    host_name               abhijain
    service_description     SSH Monitoring
    check_command            check_nrpe!check_ssh
}

define service{
    use                     generic-service
    host_name               abhijain
    service_description     FTP Monitoring
    check_command            check_nrpe!check_ftp
}

```

Fig 4.2 shows sample service configuration file

Once the services are hosts are configured, nagios service could be started. It will load the host and services information from the configuration files and will monitor the hosts and services accordingly.

```

abhishek@abhishek-VirtualBox:~$ systemctl status nagios
● nagios.service - Nagios Core 4.4.5
   Loaded: loaded (/lib/systemd/system/nagios.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-05-04 14:20:07 IST; 2 weeks 4 days ago
     Docs: https://www.nagios.org/documentation
  Main PID: 636 (nagios)
    Tasks: 8 (limit: 4675)
   CGroup: /system.slice/nagios.service
           └─ 636 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
              637 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
              638 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
              639 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
              640 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
              658 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
              12465 /bin/ping -n -U 30 -c 5 10.38.150.45
              12466 /usr/local/nagios/libexec/check_nrpe -H 10.38.150.45 -c check_load

May 19 15:27:50 abhishek-VirtualBox nagios[636]: wproc: stderr line 01: /bin/sh: 1: /bin/mail: not found
May 19 15:27:50 abhishek-VirtualBox nagios[636]: wproc: stderr line 02: /usr/bin/printf: write error: Broken pipe
May 22 17:54:47 abhishek-VirtualBox nagios[636]: Warning: A system time change of 266342 seconds (3d 1h 59m 2s forwards in time) has been detected. Compensating...
May 22 17:56:37 abhishek-VirtualBox nagios[636]: Auto-save of retention data completed successfully.
May 22 17:56:58 abhishek-VirtualBox nagios[636]: HOST NOTIFICATION: nagiosadmin;tecmint;DOWN;notify-host-by-email;CRITICAL - Time to live exceeded (10.38.150.45)
May 22 17:56:58 abhishek-VirtualBox nagios[636]: wproc: NOTIFY Job 838 from worker Core Worker 637 is a non-check helper but exited with return code 127
May 22 17:56:58 abhishek-VirtualBox nagios[636]: wproc: hosttecmint; service:(none); contact:nagiosadmin
May 22 17:56:58 abhishek-VirtualBox nagios[636]: wproc: early_timeout=0; exited 0ks; wait status=32512; error_code=0;
May 22 17:56:58 abhishek-VirtualBox nagios[636]: wproc: stderr line 01: /bin/sh: 1: /bin/mail: not found
May 22 17:56:58 abhishek-VirtualBox nagios[636]: wproc: stderr line 02: /usr/bin/printf: write error: Broken pipe

```

Fig 4.3 shows Nagios service status

Nagios performs two types of checks i.e. host checks and services checks. Host checks are the one which checks if the host is reachable or not whereas services are the once which checks whether the concerned services are working well or not. Nagios gives a UI as well for better visibility and accessibility.

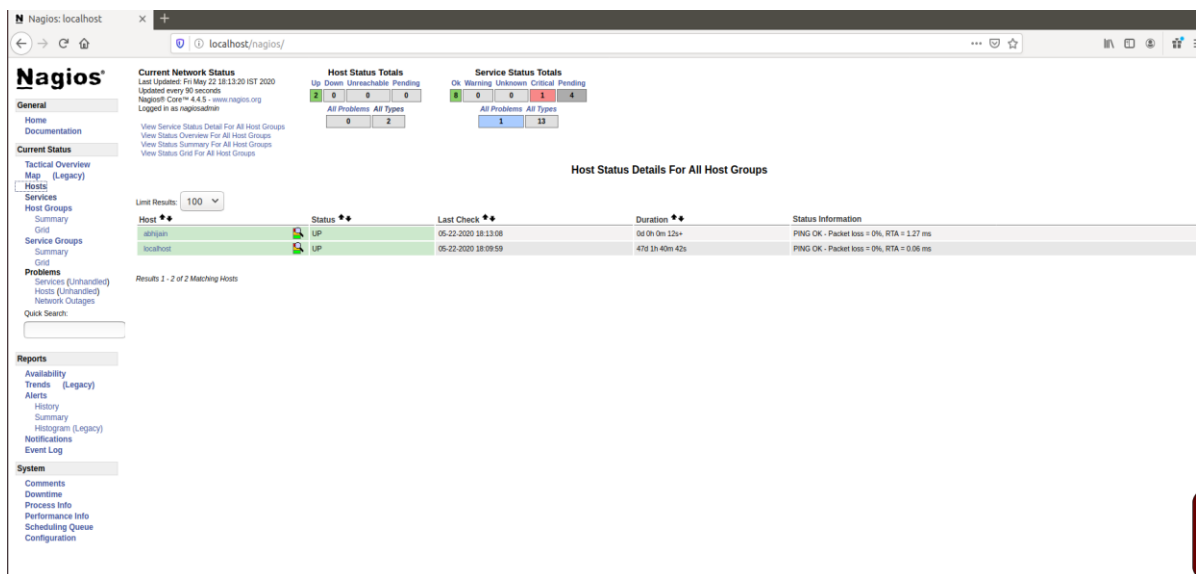


Fig 4.4 shows Nagios UI for host checks

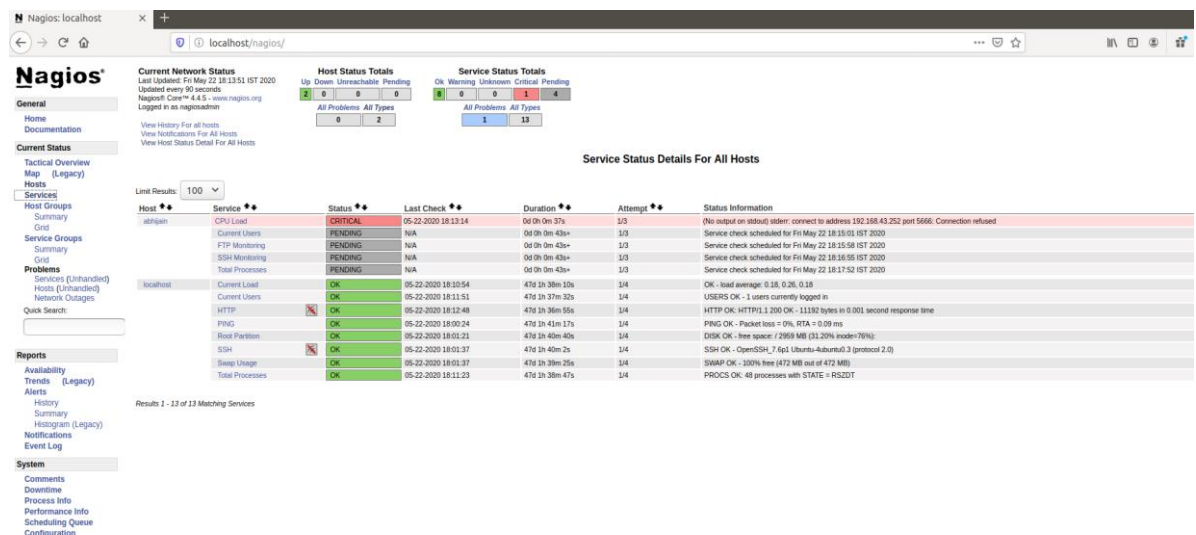
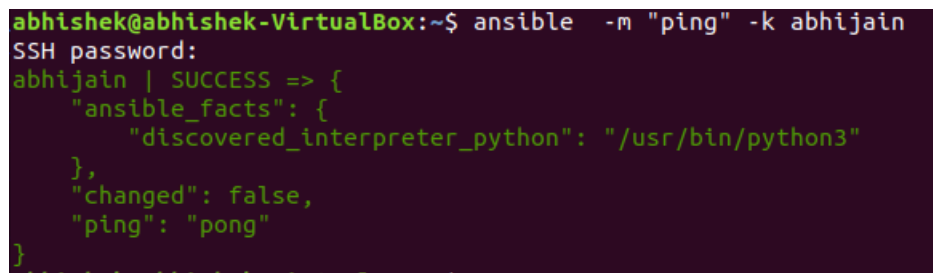


Fig 4.5 show Nagios UI for service checks

4.3 Ansible for Management

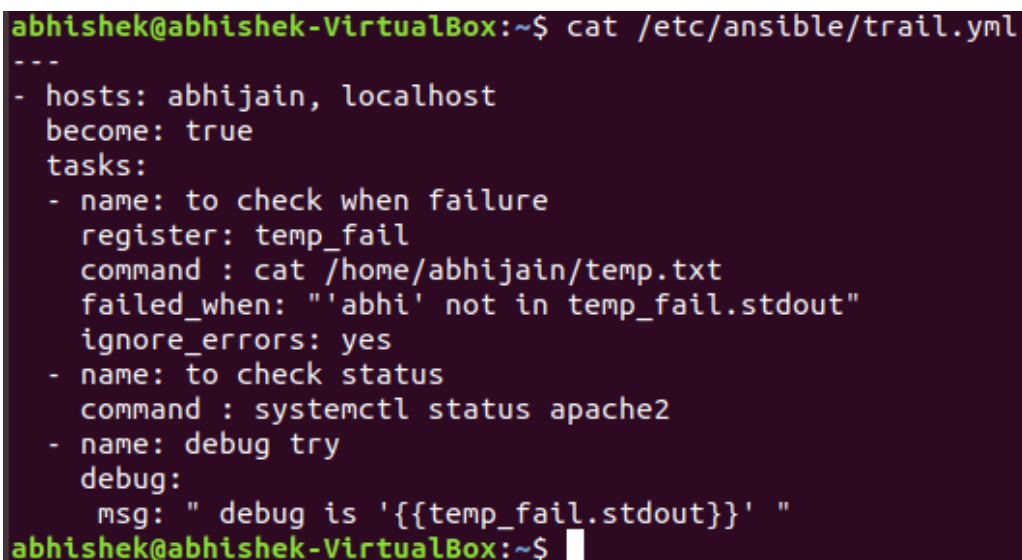
Ansible is the tool which is used to automate the process of management and configuration on the target host. Ansible has custom modules which can be used to manage the target host.



```
abhishhek@abhishhek-VirtualBox:~$ ansible -m "ping" -k abhijain
SSH password:
abhijain | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

Fig 4.6 shows result of PING module

With the help of Ansible playbooks multiple commands could be executed on multiple hosts. In the playbooks, tasks and hosts are defined. Ansible also provides features like debugging and logging.



```
abhishhek@abhishhek-VirtualBox:~$ cat /etc/ansible/trail.yml
---
- hosts: abhijain, localhost
  become: true
  tasks:
    - name: to check when failure
      register: temp_fail
      command : cat /home/abhijain/temp.txt
      failed_when: "'abhi' not in temp_fail.stdout"
      ignore_errors: yes
    - name: to check status
      command : systemctl status apache2
    - name: debug try
      debug:
        msg: " debug is '{{temp_fail.stdout}}' "
```

Fig 4.7

Figure 4.7 shows a sample Ansible playbook. It consists of three tasks:

- First task is to find if the pattern ‘abhi’ in the file contents or not. It also uses ‘ignore_errors’ module to continue the execution of the script if the task is marked as failed.
- Second task is to check the status of Apache server on the remote host.
- Third task is to print the output of the first task. The ‘debug’ module is used for debugging purposes.

```

abhi@abhishek-VirtualBox:~$ ansible-playbook /etc/ansible/trail.yml -k
SSH password:

PLAY [abhi@localhost] *****

TASK [Gathering Facts] *****
ok: [abhi@localhost]
[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host localhost should use
/usr/bin/python3, but is using /usr/bin/python for backward compatibility with
prior Ansible releases. A future Ansible release will default to using the
discovered platform python for this host. See https://docs.ansible.com/ansible/
2.9/reference_appendices/interpreter_discovery.html for more information. This
feature will be removed in version 2.12. Deprecation warnings can be disabled
by setting deprecation_warnings=False in ansible.cfg.
ok: [localhost]

TASK [to check when failure] *****
ok: [abhi@localhost] => { "changed": true, "cmd": ["cat", "/home/abhi@localhost/temp.txt"], "delta": "0:00:00.001833", "end": "2020-05-22 18:17:52.288983", "failed_when_result": true, "msg": "non-zero ret
urn code", "rc": 1, "start": "2020-05-22 18:17:52.286250", "stderr": "cat: /home/abhi@localhost/temp.txt: No such file or directory", "stderr_lines": ["cat: /home/abhi@localhost/temp.txt: No such file or directory"],
"stdout": "", "stdout_lines": []}
...ignoring
cmd: cat /home/abhi@localhost/temp.txt
stderr: cat: /home/abhi@localhost/temp.txt: No such file or directory
ok: [abhi@localhost] => { "changed": false, "cmd": ["cat", "/home/abhi@localhost/temp.txt"], "delta": "0:00:00.002855", "end": "2020-05-22 18:17:52.290833", "failed_when_result": true, "rc": 0, "start": "20
20-05-22 18:17:52.287978", "stderr": "", "stderr_lines": [], "stdout": "ajain", "stdout_lines": ["ajain"]}
...ignoring
cmd: cat /home/abhi@localhost/temp.txt
stdout: ajain

TASK [to check status] *****
changed: [localhost]

cmd: systemctl status apache2
[WARNING]: Failure using method (v2_runner_on_ok) in callback plugin
(<ansible.plugins.callback_human_log.CallbackModule object at 0x7ff6683f3b90>):
'ascii' codec can't encode character 'u'\u25cf' in position 0: ordinal not in
range(128)
changed: [abhi@localhost]
cmd: systemctl status apache2

TASK [debug try] *****
ok: [abhi@localhost] => {
  "msg": " debug is 'ajain' "
}
ok: [localhost] => {
  "msg": " debug is ' ' "
}

PLAY RECAP *****
abhi@localhost      : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=1
localhost           : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=1

Playbook run took 0 days, 0 hours, 0 minutes, 3 seconds

```

Fig 4.8

The output of playbook mentioned in figure 4.7 is shown in figure 4.8. It also uses the custom ‘timer’ plugin which tells how much time is used by the Ansible for executing tasks mentioned in the playbook on the mentioned hosts. With the help of Ansible plugins, the output could be modified as per requirements.

CHAPTER- 5

CONCLUSIONS AND FUTURE WORK

CHAPTER: 5 CONCLUSIONS

5.1 Conclusion

After studying about various tools for monitoring and management of hosts, we came to know that these tools can make the process easier. With the help of automation, the hosts can be managed and monitored in a better way. Consider a cloud consisting of around thousands of nodes. To monitor and manage those large number of tasks is a tough task. Moreover it can lead to errors and degradation of services. With the help of Nagios and Ansible, the tasks could be automated. Custom plugins could be developed according to the requirements. The tools like Alertmanager, Pagerduty and VictorOps could be used for alerting purposes. There are various network utility tools which could be used to trace connection problems

5.2 Future Work

The information gathering process of the hosts that are to monitored could be automated with the help of various scripts. Custom Nagios and Ansible scripts could be developed to enhance the existing capability of the tools. Our own tools could be developed as well to automate the whole management process.

References

1. www.python.org
2. www.pagerduty.com
3. www.alertmanager.io
4. www.nagios.org
5. www.prometheus.io
6. www.jenkins.io
7. www.postgresql.org
8. www.ssh.com