LAB ASSIGNMENT 9

AIM: Installation of nagios on ubuntu system.

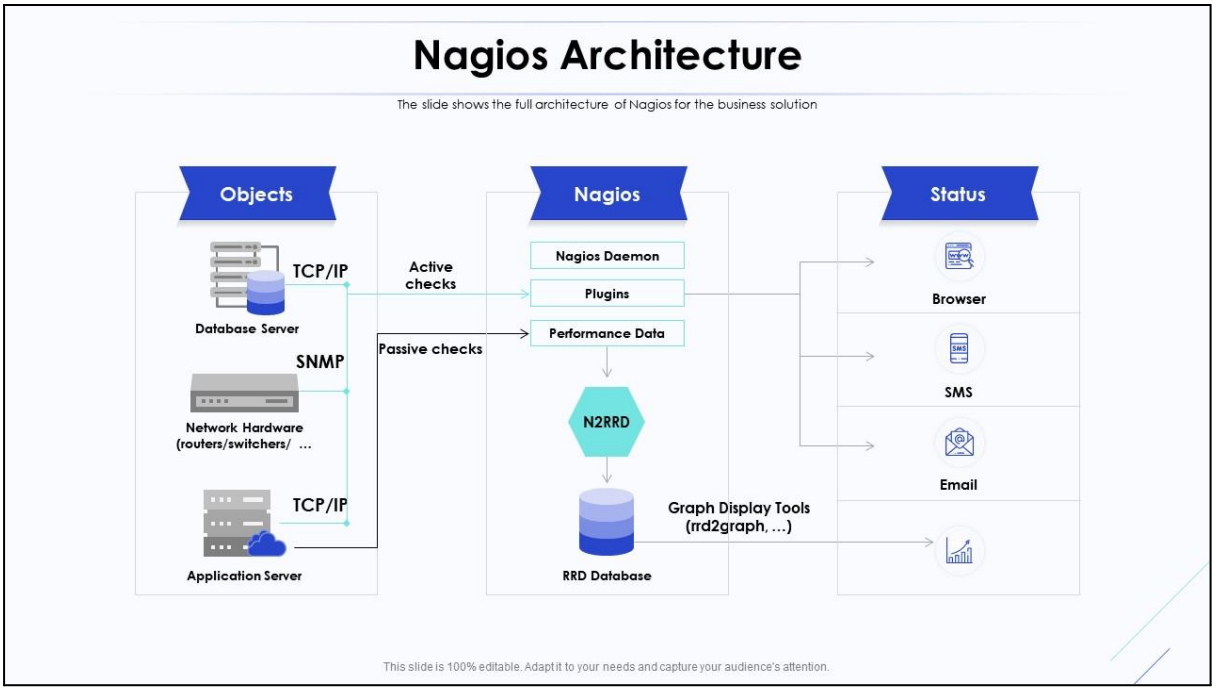
LAB OUTCOME:
LO1, LO5 Mapped.

THEORY:
Nagios is an open-source monitoring and alerting system that is widely used to monitor the availability and health of IT infrastructure components, including servers, network devices, applications, and services. It helps organisations maintain the reliability and stability of their systems by providing real time visibility into the performance and status of various components in their environment.

Key features and components of Nagios includes:

1. Host and Service Monitoring: Nagios allows users to define hosts (e.g., servers, routers) and services (e.g., web services, email servers) to be monitored. It periodically checks these hosts and services to ensure they are functioning correctly.
2. Alerting and Notification: When Nagios detects a problem or a service outage, it can send notifications to designated administrators or teams through various methods, including email, SMS, and custom scripts. This enables timely responses to issues.
3. Flexible Configuration: Nagios is highly configurable and allows users to define custom checks, thresholds, and notification rules. This flexibility makes it suitable for a wide range of monitoring scenarios.
4. Web Interface: Nagios provides a web-based dashboard that offers a real-time view of the monitored infrastructure's status. Administrators can access this dashboard to see which services are up or down and view historical performance data.
5. Plugin Architecture: Nagios uses a plugin system that allows users to extend its monitoring capabilities. Many plugins are available for monitoring specific applications, devices, or protocols, and users can develop custom plugins as needed.
6. Performance Graphs: Nagios can collect performance data and display it in graphs and charts. This helps in analysing historical trends and identifying potential issues before they become critical.
7. Event Logging: Nagios keeps a detailed log of monitoring events and notifications. This log can be useful for troubleshooting and auditing.
8. Scheduled Downtime: Administrators can schedule downtime for planned maintenance or upgrades to prevent unnecessary alerts during maintenance windows.
9. Community and Support: Nagios has an active user community, which provides resources, documentation, and support for users. There are also commercial versions and third-party tools built around Nagios for additional features and support.

Nagios is highly versatile and can be used in various IT environments, from small businesses to large enterprises. It plays a crucial role in ensuring the availability and performance of critical infrastructure components, helping organisations proactively address issues and minimise downtime. Additionally, Nagios can be integrated into larger IT management and monitoring solutions to provide comprehensive visibility into an organisation's technology stack.



WORKING:
Nagios works based on a simple yet effective principle: monitoring and alerting. It continuously checks the status and performance of various hosts (e.g., servers, network devices) and services (e.g., applications, websites) by running predefined checks, known as plugins.

1. Configuration: The Nagios administrator defines what needs to be monitored and how in a configuration file. This includes specifying hosts, services, notification settings, and alert thresholds. Users can configure checks to run at specific intervals, such as every minute, and set warning and critical thresholds for each service (e.g., response time should be under 200ms).

2. Checks and Plugins: Nagios periodically runs these checks based on the defined intervals and uses plugins to perform the checks. A plugin is a small script or executable that carries out a specific monitoring task, such as pinging a server, checking the HTTP response code of a website, or monitoring disk space usage.

3. Status Data: After running the checks, Nagios collects status data, including the results of the checks, timestamps, and performance metrics (if applicable). This data is stored internally.

4. Alerting and Notifications: If a check indicates a problem (e.g., a service is down, a server is unresponsive, a threshold is exceeded), Nagios triggers an alert. Alerts can be notifications sent to administrators or teams through various means, such as email, SMS, or custom scripts. These alerts inform relevant personnel about the issue, enabling them to take action.

5. Dashboard and Reports: Nagios provides a web-based dashboard where administrators can view the real-time status of monitored hosts and services. They can see which services are up, which are down, and view historical performance data in the form of graphs and charts. This dashboard is a central hub for monitoring and managing the infrastructure.

6. Event Logging: Nagios keeps a detailed log of all monitoring events, including check results, alerts, and notifications. This log serves as an audit trail and can be useful for troubleshooting and historical analysis.

7. Scheduled Downtime: Nagios allows administrators to schedule downtime for hosts and services during planned maintenance windows. This prevents Nagios from generating unnecessary alerts during maintenance activities.

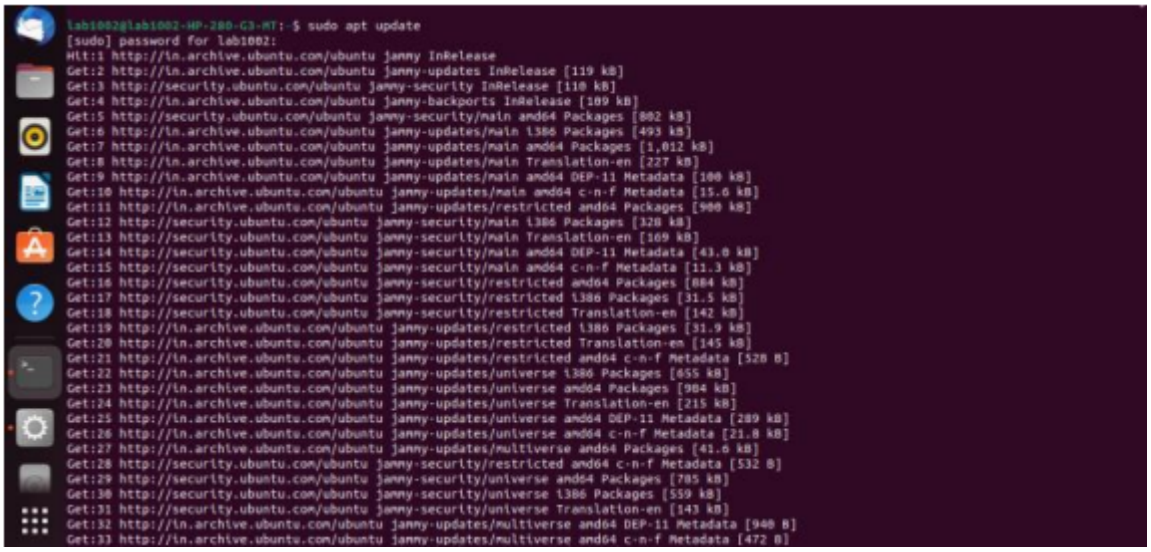
8. Recovery Notifications: When a previously failed service or host returns to a healthy state, Nagios can send recovery notifications to inform administrators that the issue has been resolved.

9. Escalation and Acknowledgements: Nagios supports advanced features like escalation, where alerts can be escalated to higher-level teams if not acknowledged or resolved within a certain timeframe. Administrators can also acknowledge alerts, indicating that they are aware of the issue and are working on it.

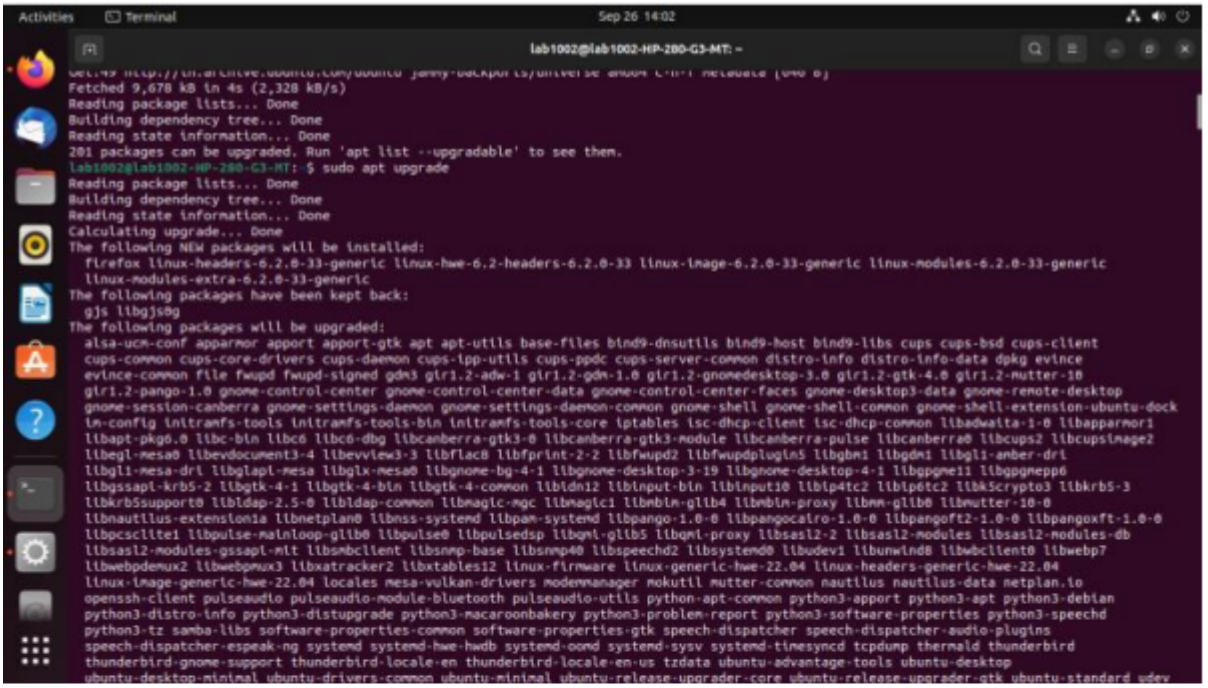
10. Performance Data: Nagios can collect and display performance data, which helps in identifying trends, bottlenecks, and potential issues before they become critical. This data can be used for capacity planning and optimization.

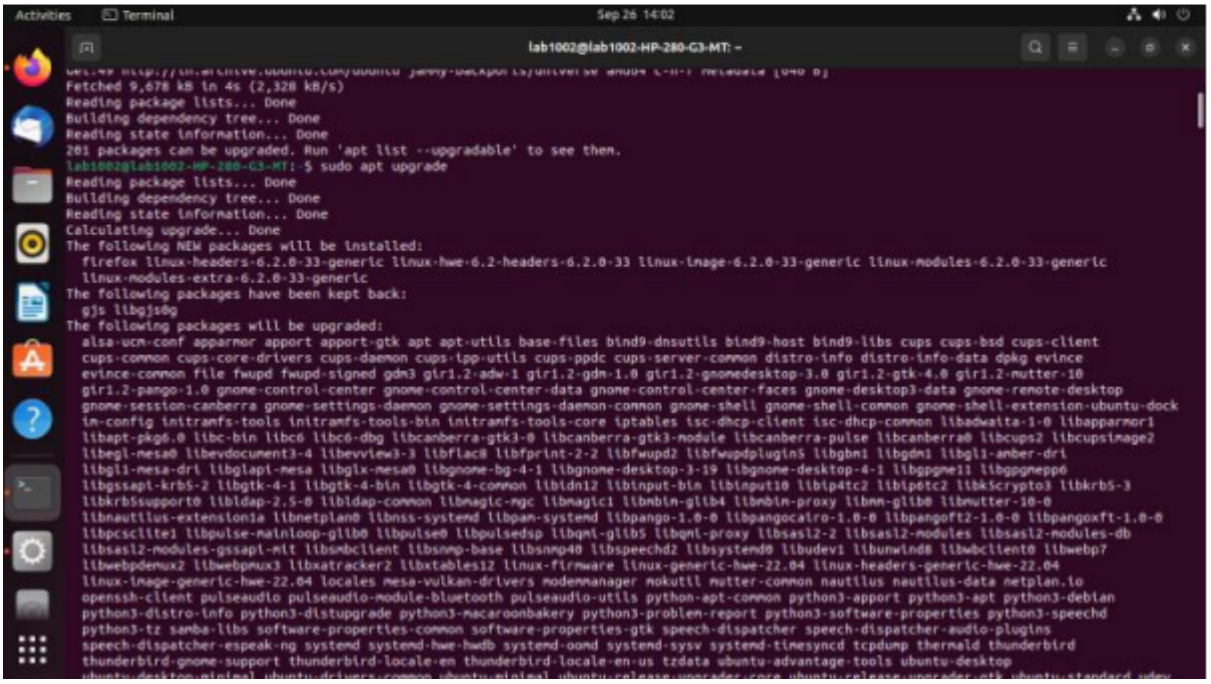
Nagios operates continuously, providing real-time monitoring and alerting for IT infrastructure components. It helps organisations maintain the availability and performance of their systems and applications while enabling prompt responses to issues, ultimately reducing downtime and ensuring a more stable IT environment.

INSTALLATION STEPS:



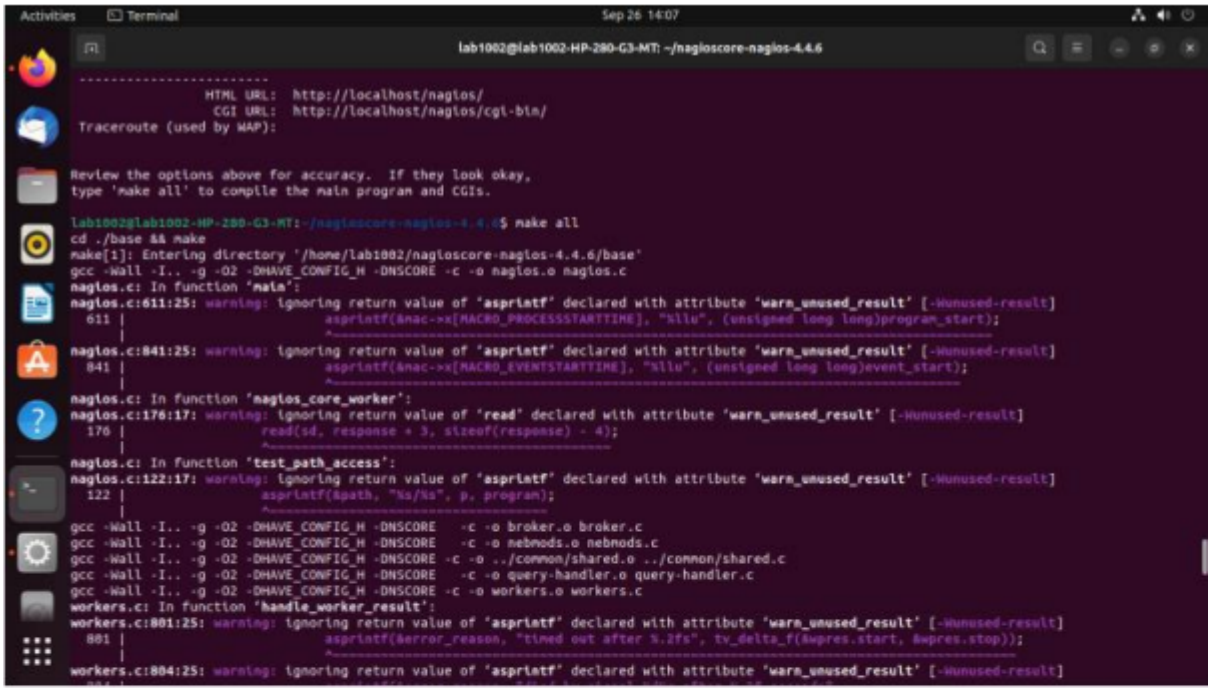
Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

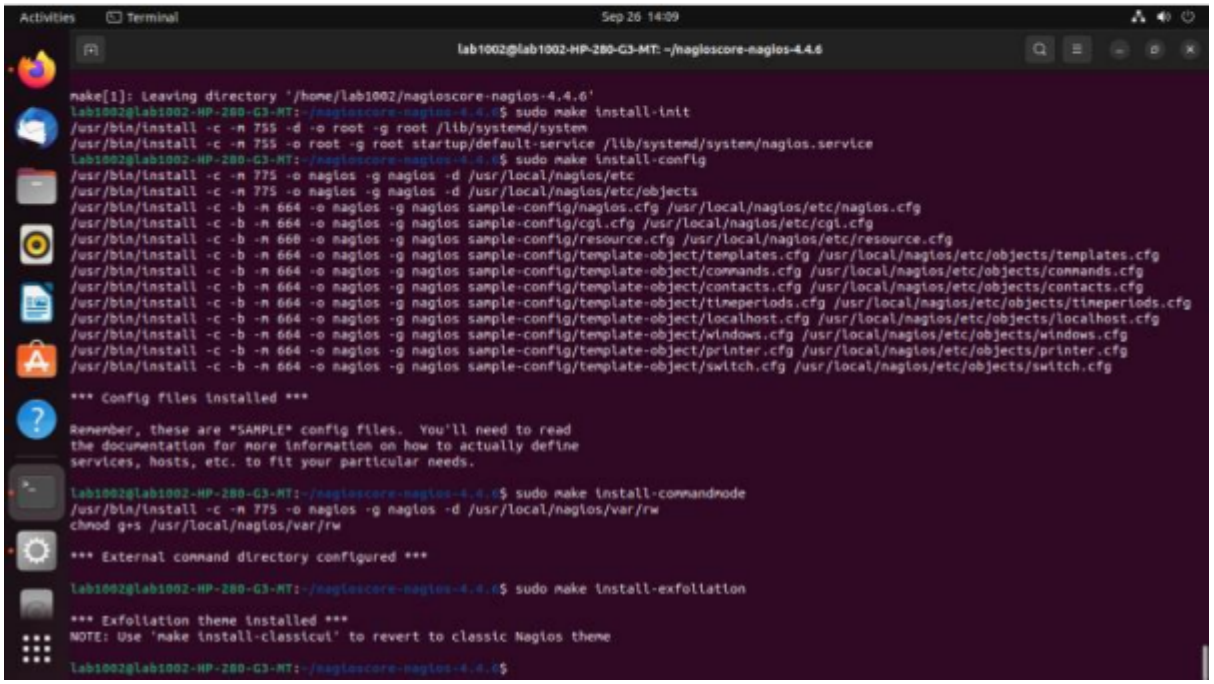


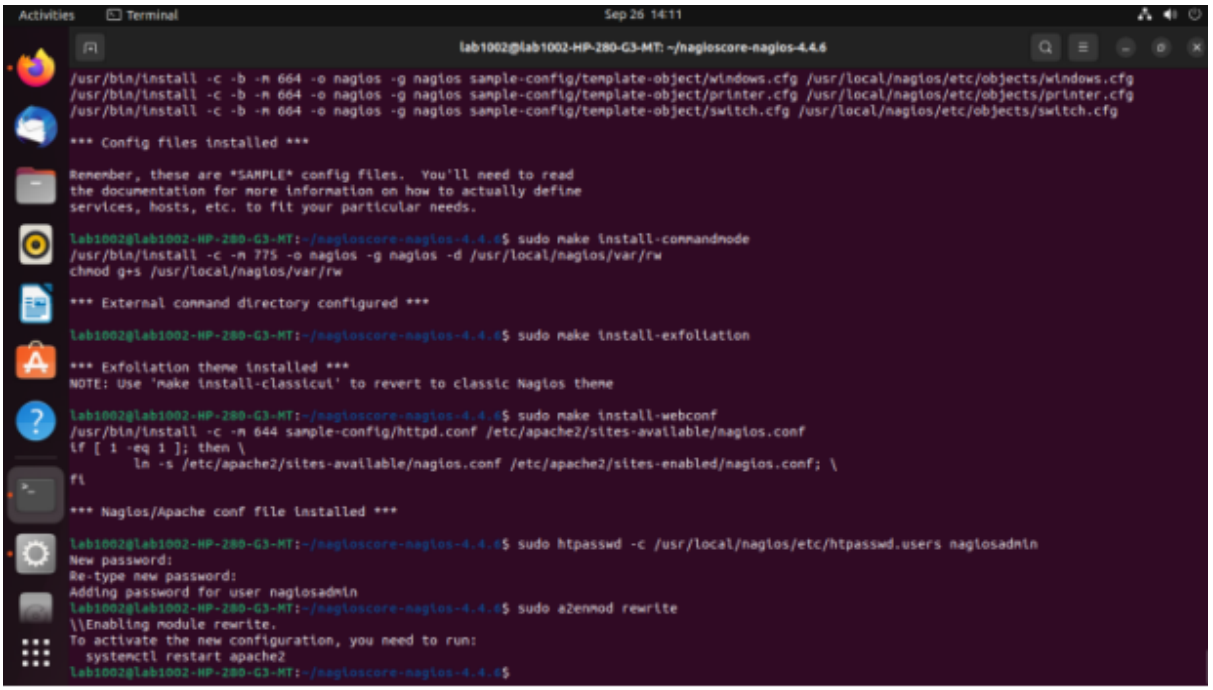


```
Activities   Terminal                    Sep 26 14:05
lab1002@lab1002-HP-280-G3-MT: ~
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.9ubuntu3) ...
Setting up php (2:8.1+92ubuntu1) ...
Processing triggers for ufw (0.36.1-4ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.3) ...
Processing triggers for php8.1-cli (8.1.2-1ubuntu2.14) ...
Processing triggers for libapache2-mod-php8.1 (8.1.2-1ubuntu2.14) ...
lab1002@lab1002-HP-280-G3-MT:~$ sudo useradd nagios
lab1002@lab1002-HP-280-G3-MT:~$ sudo groupadd nagcmd
lab1002@lab1002-HP-280-G3-MT:~$ sudo usermod -a -G nagcmd nagios
lab1002@lab1002-HP-280-G3-MT:~$ sudo usermod -a -G nagcmd www-data
lab1002@lab1002-HP-280-G3-MT:~$ wget https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.6.tar.gz
--2023-09-26 14:04:26--  https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.6.tar.gz
Resolving github.com (github.com)... 20.207.73.82
Connecting to github.com (github.com)|20.207.73.82|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/NagiosEnterprises/nagioscore/tar.gz/refs/tags/nagios-4.4.6 [following]
--2023-09-26 14:04:26--  https://codeload.github.com/NagiosEnterprises/nagioscore/tar.gz/refs/tags/nagios-4.4.6
Resolving codeload.github.com (codeload.github.com)... 20.207.73.88
Connecting to codeload.github.com (codeload.github.com)|20.207.73.88|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: 'nagios-4.4.6.tar.gz'

nagios-4.4.6.tar.gz                     [       <=>                                     ]  10.81M  8.06MB/s    in 1.3s

2023-09-26 14:04:28 (8.06 MB/s) - 'nagios-4.4.6.tar.gz' saved [11333431]

lab1002@lab1002-HP-280-G3-MT:~$ tar -zxvf nagios-4.4.6.tar.gz
nagioscore-nagios-4.4.6/
nagioscore-nagios-4.4.6/.gitignore
nagioscore-nagios-4.4.6/.travis.yml
nagioscore-nagios-4.4.6/CONTRIBUTING.md
nagioscore-nagios-4.4.6/Changelog
nagioscore-nagios-4.4.6/INSTALLING
nagioscore-nagios-4.4.6/LEGAL
nagioscore-nagios-4.4.6/LICENSE
```

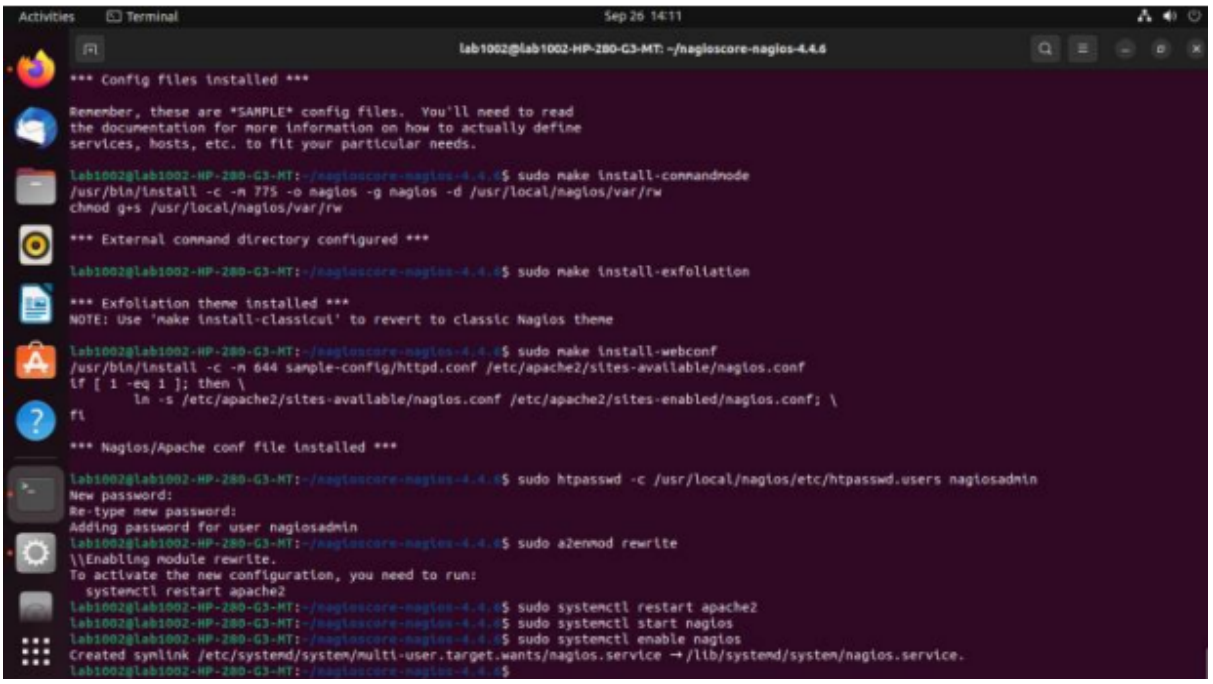
```
Activities   Terminal                    Sep 26 14:07
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
nagioscore-nagios-4.4.6/xdata/xodtemplate.c
nagioscore-nagios-4.4.6/xdata/xodtemplate.h
nagioscore-nagios-4.4.6/xdata/xpddefault.c
nagioscore-nagios-4.4.6/xdata/xpddefault.h
nagioscore-nagios-4.4.6/xdata/xrddefault.c
nagioscore-nagios-4.4.6/xdata/xrddefault.h
nagioscore-nagios-4.4.6/xdata/xsddefault.c
nagioscore-nagios-4.4.6/xdata/xsddefault.h
lab1002@lab1002-HP-280-G3-MT:~$ cd nagioscore -nagios-4.4.6
bash: cd: too many arguments
lab1002@lab1002-HP-280-G3-MT:~$ cd nagioscore-nagios-4.4.6
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ ./configfure --with-http-conf=/etc/apache2/sites-enabled
bash: ./configfure: No such file or directory
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ ./configure --with-http-conf=/etc/apache2/sites-enabled
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking whether time.h and sys/time.h may both be included... yes
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
```


Activities
Terminal
Sep 26 14:07
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
HTML URL: http://localhost/nagios/
CGI URL: http://localhost/nagios/cgi-bin/
Traceroute (used by WAP):
Review the options above for accuracy. If they look okay,
type 'make all' to compile the main program and CGIs.
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ make all
cd ./base && make
make[1]: Entering directory '/home/lab1002/nagioscore-nagios-4.4.6/base'
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c
nagios.c: In function 'main':
nagios.c:611:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]
nagios.c:841:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]
nagios.c: In function 'nagios_core_worker':
nagios.c:176:17: warning: ignoring return value of 'read' declared with attribute 'warn_unused_result' [-Wunused-result]
nagios.c: In function 'test_path_access':
nagios.c:122:17: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
workers.c: In function 'handle_worker_result':
workers.c:801:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]
workers.c:804:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]


Activities
Terminal
Sep 26 14:09
lab1002@lab1002-HP-280-G3-MT: ~/nagioscore-nagios-4.4.6
make[1]: Leaving directory '/home/lab1002/nagioscore-nagios-4.4.6'
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg
*** Config files installed ***
Remember, these are *SAMPLE* config files. You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw
*** External command directory configured ***
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-exfoliation
*** Exfoliation theme installed ***
NOTE: Use 'make install-classicui' to revert to classic Nagios theme
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$



```
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-exfoliation

*** Exfoliation theme installed ***
NOTE: Use 'make install-classicui' to revert to classic Nagios theme

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/apache2/sites-available/nagios.conf
if [ 1 -eq 1 ]; then \
        ln -s /etc/apache2/sites-available/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo a2enmod rewrite
\\Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$
```



```
*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-exfoliation

*** Exfoliation theme installed ***
NOTE: Use 'make install-classicui' to revert to classic Nagios theme

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/apache2/sites-available/nagios.conf
if [ 1 -eq 1 ]; then \
        ln -s /etc/apache2/sites-available/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

*** Nagios/Apache conf file installed ***

lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo a2enmod rewrite
\\Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo systemctl restart apache2
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo systemctl start nagios
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$ sudo systemctl enable nagios
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /lib/systemd/system/nagios.service.
lab1002@lab1002-HP-280-G3-MT:~/nagioscore-nagios-4.4.6$
```

CONCLUSION:

In this assignment, we successfully installed Nagios on an Ubuntu system, setting up a robust monitoring and alerting framework for efficient system management and proactive issue resolution.

LAB ASSIGNMENT 10

AIM: To study Puppet tools in Devops.

LAB OUTCOME:
LO1, LO6 Mapped.

THEORY:

WHAT IS A PUPPET TOOL?
Puppet is an open-source configuration management and automation tool used for deploying, configuring, and managing servers and infrastructure as code. It helps system administrators and DevOps teams automate repetitive tasks, enforce consistent configurations, and ensure that infrastructure is in the desired state, reducing manual effort and minimizing errors.

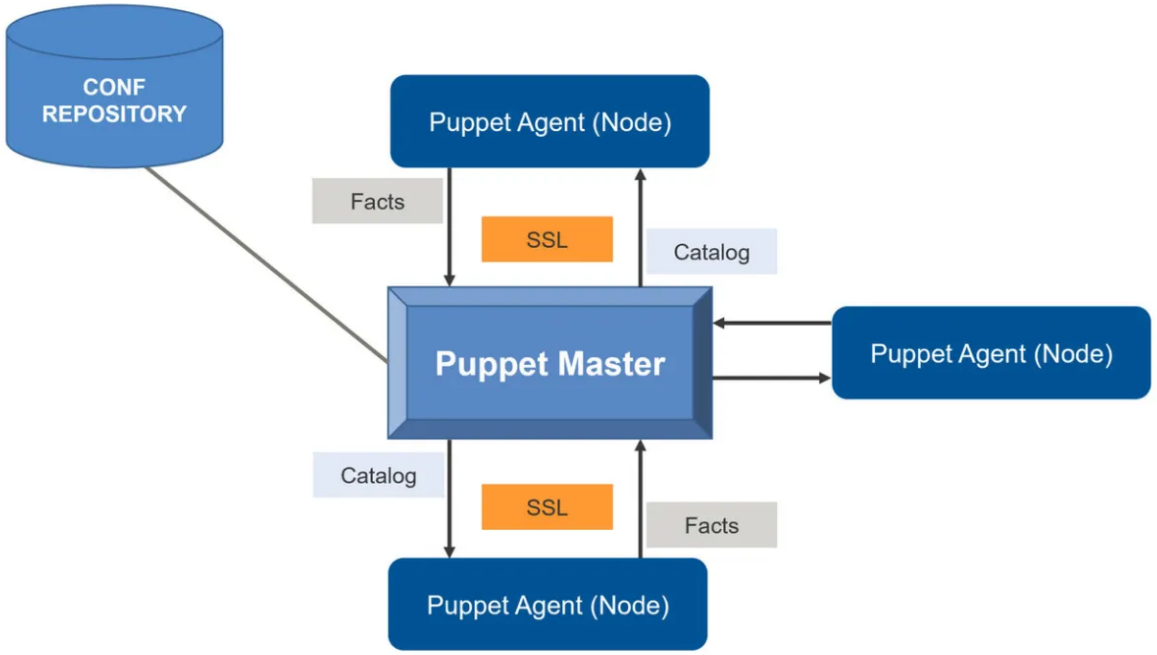
Features of Puppet:
- **Automation**: Puppet automates the configuration and management of servers and infrastructure, reducing manual tasks and errors.
- **Declarative Language**: It uses a declarative language to specify desired infrastructure states, making it easy to define what the system should look like.
- **Infrastructure as Code (IaC)**: Puppet treats infrastructure configurations as code, enabling versioning, testing, and collaboration like software code.
- **Resource Abstraction**: Puppet abstracts system resources into manageable "resources," simplifying configuration management.
- **Module-Based**: It organizes configurations into reusable modules, streamlining the management of common software and services across different parts of your infrastructure.

WHAT CAN A PUPPET DO?
The Puppet Server is responsible for:
- Compiling the Catalog File for hosts, based on system, configuration, manifest file, etc. Puppet prepares a Catalog File based on the manifest file, which is a Puppet program used to control the systems running the Puppet Agent. After processing the manifest file, the Puppet Server prepares the Catalog File based on the target platform.
- Sending the Catalog File to Agents when they query the Server.
- Storing information about the entire environment, such as host information, metadata such as authentication keys.
- Gathering reports from each Agent and then preparing the overall report.

HOW DO PUPPETS WORK?
Puppet uses an agent-server model for configuring systems, with agents and a server known as the Puppet Agent and Puppet Server, respectively. The key points are:

- Puppet Agent must be installed on each system for management.
- Agents connect securely to the Puppet Server to receive instructions in a file referred to as the Catalog.
- Agents execute instructions to reach the desired system state and report back to the server.

- Puppet employs a declarative Domain Specific Language (DSL) for configuration attributes, defined in manifests.
- Agents collect facts about themselves and send them to the Puppet Master.
- The Puppet Master compiles catalogs specifying how each node should be configured.
- Catalogs are sent to agents, which then apply the configurations, ensuring consistency.
- The Puppet Master manages the entire infrastructure, including compiling catalogs, sending reports, and handling file transfers.
- Communication is secured through SSL/TLS protocols, encrypting traffic between nodes and the master.



Puppet Blocks

Puppet Blocks. Resources- Puppet has many built-in resources like file, user, package and service. The puppet language allows administrators to manage the system resources independently and ensure that the system is in the desired state.

Puppet Resources

Resources are one of the key fundamental units of Puppet used to design and build any particular infrastructure or a machine. They are mainly used for modeling and maintaining system configurations. Puppet has multiple types of resources, which can be used to define the system architecture or the user has the leverage to build and define a new resource.

Puppet Classes

Puppet classes are defined as a collection of resources, which are grouped together in order to get a target node or machine in a desired state. These classes are defined inside Puppet manifest files which are located inside Puppet modules.

Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

Puppet Modules
Modules serve as the basic building blocks of Puppet and are reusable and shareable. Modules contain Puppet classes, defined types, tasks, task plans, functions, resource types and providers, and plug-ins such as custom types or facts. Modules must be installed in the Puppet modulepath.

Puppet Manifest File
A manifest is a file containing Puppet configuration language that describes how resources should be configured. The manifest is the closest thing to what one might consider a Puppet program. It declares resources that define state to be enforced on a node.

PUPPET MANIFEST FILES:

- In puppet, all the programs are written in Ruby programming language and added with an extension of .pp is known as manifests. The full form of .pp is the puppet program.
- Manifest files are puppet programs. This is used to manage the target host system. All the puppet programs follow the puppet coding style. We can use a set of different kinds of resources in any manifest, which is grouped by definition and class.
- Puppet manifest also supports the conditional statement. The default manifest file is available in the /etc/puppet/manifests/site.pp location.

SYNTAX OF A MANIFEST FILE:
A manifest file contains the following components:

1. Resource Declaration
2. Comments
3. Variables
4. Conditional Statements
5. Classes and Modules
6. Include Classes
7. Templates
8. Resource Ordering
9. Node Definitions

EXAMPLE OF MANIFEST FILES:

```
# This is a Puppet manifest file
# Define a file resource
file { '/etc/myconfig':
  ensure  => 'file',
  owner   => 'root',
  group   => 'root',
  mode    => '0644',
  content => 'This is the configuration content.',
# Set file content
}
# Define a package resource
package { 'nginx':
  ensure => 'installed',
}
# Define a service resource
service { 'nginx':
```

```
  ensure    => 'running',
  enable    => true,
  subscribe => File['/etc/myconfig'],
}
# Node definition
node 'webserver.example.com' {
  include nginx_config
}
```

WHY DO WE NEED PUPPET MANIFEST FILES?

Puppet manifest files are crucial for:

- Automation: They automate configurations, reducing manual work.
- Consistency: Ensure a uniform state across servers.
- Version Control: Track changes and collaborate using code repositories.
- Modularity: Reuse configurations for efficiency.
- Customization: Adapt configurations using variables.
- Documentation: Clearly document infrastructure setups.
- Scalability: Scale configurations effortlessly.
- Compliance: Enforce security and compliance standards.
- Change Control: Manage changes systematically.

BENEFITS OF PUPPET:

1. Eliminates time consuming, complex and stressful manual configurations of the infrastructure
2. Automates the process of configurations, controlling and managing large numbers (over 100 servers) of servers and other infrastructure
3. Eliminates complex error-prone tasks of automating the infrastructure deployment and configuration
4. It is an inexpensive method of solving the configurations bottlenecks and latency in the speed
5. Puppet is used as a continuous delivery model to the software release cycle by automating the operations and deployment workflow

CONCLUSION:

Through this assignment, I have learnt the concept of Puppet Tools in DevOps, its different components and working. I also understood the fundamentals of Manifest files in Puppet.