Faculdade de Ciências da Universidade de Lisboa

Departamento de Informática

Mestrado em Engenharia Informática e Segurança Informática

RELATÓRIO – GRUPO 10

Configuração e Gestão de Sistemas

Knowledge Base - IT infrastructure

Alunos: **Rodrigo Rodrigues (fc64370)**
Diogo Agostinho (fc64416)
João Videira (fc64415)

Professor: **Doutor Hugo Miranda**

2º Semestre Letivo 2024/2025

maio 2025

Índice

1. Network Architecture & IP Assignment

1.1 Network Topology

The infrastructure consists of three virtual servers on the 10.101.150.0/24 network segment. The servers form a logical triangle with specific communication paths dictated by service dependencies and security considerations. This topology creates natural security boundaries while enabling necessary service interactions.

1.2 IP Address Calculation & Assignment

IP addresses are systematically assigned using the formula:

```
Base IP = 51 + ((Group Number - 1) × 3)
```

For Group 10, this calculation yields:

```
Base = 51 + ((10 - 1) × 3) = 51 + 27 = 78
```

Therefore:

Server	Function	IP Address
Server 0	ownCloud & MariaDB	10.101.150.78
Server 1	Web Server & Nagios	10.101.150.79
Server 2	DNS & Samba	10.101.150.80

2.3 Network Verification

Before beginning service deployment, verify network configuration and connectivity:

```
# Test basic connectivity
ping -c 3 10.101.150.78
ping -c 3 10.101.150.79
ping -c 3 10.101.150.80

# Verify SSH access
ssh cgsadmin@10.101.150.78
ssh cgsadmin@10.101.150.79
ssh cgsadmin@10.101.150.80
# Check network interface configuration
ip addr show
```

Note: To access the web contents of the server, in your PC browser check this section

2. Initial Access & Security Guidelines

2.1 VPN Connection

Secure remote access requires establishing a VPN connection before attempting to access any of the servers. This provides an additional security layer by ensuring all administrative traffic is encrypted.

Setup Instructions:

1. Follow FCUL's official VPN installation guide: https://ciencias.ulisboa.pt/pt/vpn
2. Select the appropriate client for your operating system
3. Configure the client using the credentials provided by the FCUL IT department
4. Verify connection status before attempting server access
5. Troubleshoot connection issues by checking client logs and network configurations

Security Considerations:

- Never attempt to access the servers without an active VPN connection
- Keep the VPN client updated to the latest version
- Disconnect the VPN when administrative tasks are complete
- Do not share VPN credentials or allow session sharing

2.2 SSH Access Configuration

Server	IP Address	Default Password	Changed Password
Server 0	10.101.150.78	cgsadmin	cgsg10s78admin
Server 1	10.101.150.79	cgsadmin	cgsg10s79admin
Server 2	10.101.150.80	cgsadmin	cgsg10s80admin

Immediate Security Hardening Steps:

1. Change default passwords using strong password guidelines:

```
passwd
```

2. Restart SSH service to apply changes:

```
sudo systemctl restart sshd
```

Important: *Always verify SSH access with key authentication before disabling password authentication to prevent lockouts. Maintain a backup access method in case of key issues.*

3. Firewall Configuration (UFW)

3.1 Global Policy Configuration

The firewall implementation follows the principle of "deny all by default, allow by exception."

Before changing anything Backup Current Rules for each server:

```
sudo ufw status numbered > ~/ufw-backup-$(date +%F_%H%M).rules
```

On each server, implement the global deny policies first:

```
sudo ufw default deny incoming
sudo ufw default deny outgoing
```

3.3 Server-Specific Rules

Note: *The implementation of rules in the firewall (ufw) automatically processes IPV6 rules in principle. However, the rules for IPv6 are presented in case you need them.*

3.3.1 Server 0 (10.101.150.78) — ownCloud & MariaDB

Allow Inbound Traffic:

```
# Administrative access
sudo ufw allow in proto tcp to any port 5666 from 10.101.150.79
comment 'Monitoring from 10.101.150.79'
sudo ufw allow in proto tcp to any port 80 comment 'HTTP access'
sudo ufw allow in proto tcp to any port 443 comment 'HTTPS access'
sudo ufw allow in proto tcp to any port 22 comment 'SSH access'
sudo ufw allow in proto udp to any port 53 comment 'DNS (UDP)'
sudo ufw allow in proto tcp to any port 53 comment 'DNS (TCP)'

# IPv6 rules
sudo ufw allow in proto tcp to any port 22 from Anywhere \(v6\)
comment 'SSH access (IPv6)'
sudo ufw allow in proto udp to any port 53 from Anywhere \(v6\)
comment 'DNS UDP (IPv6)'
sudo ufw allow in proto tcp to any port 53 from Anywhere \(v6\)
comment 'DNS TCP (IPv6)'
sudo ufw allow in proto tcp to any port 80 from Anywhere \(v6\)
comment 'HTTP access (IPv6)'
sudo ufw allow in proto tcp to any port 443 from Anywhere \(v6\)
comment 'HTTPS access (IPv6)'
```

Allow Outbound Traffic:

```
# DNS and other services
sudo ufw allow out proto udp to 10.101.150.80 port 53 comment 'DNS
queries to 10.101.150.80 (UDP)'
sudo ufw allow out proto tcp to 10.101.150.80 port 53 comment 'DNS
queries to 10.101.150.80 (TCP)'
sudo ufw allow out proto tcp to 10.101.150.80 port 445 comment
'SMB/CIFS to 10.101.150.80'
sudo ufw allow out proto tcp to any port 22 comment 'SSH outbound'

# IPv6 outbound
sudo ufw allow out proto tcp to any port 22 from Anywhere \(v6\)
comment 'SSH outbound (IPv6)'
```

Summary:

1. It is being monitored from address 10.101.150.79 (port 5666, possibly Nagios NRPE
2. Runs web services (HTTP/HTTPS on ports 80/443)
3. Works as a DNS server (TCP/UDP port 53)
4. Communicates with server 10.101.150.80 for DNS queries and access to SMB/CIFS shares
5. It allows both inbound and outbound SSH access.
6. It has configurations for both IPv4 and IPv6

3.3.2 Server 1 (10.101.150.79) — Web & Nagios

Allow Inbound Traffic:

```
# Web interface access
sudo ufw allow in proto tcp to any port 80 comment 'HTTP for
Web/Nagios'
sudo ufw allow in proto tcp to any port 443 comment 'HTTPS for
Web/Nagios'

# Administrative access
sudo ufw allow in proto tcp to any port 22 comment 'SSH Access'

# DNS service
sudo ufw allow in proto udp to from any to any port 53 comment 'DNS
queries (UDP)'
sudo ufw allow in proto tcp to from any to any port 53 comment 'DNS
queries (TCP)'

# Monitoring service (Nagios)
sudo ufw allow in proto tcp to any port 5666 comment 'Nagios NRPE'
sudo ufw allow in proto tcp to any port 111 comment 'RPC services'

# Specific access from other servers
sudo ufw allow in from 10.101.150.78 comment 'Access from Server 0'
sudo ufw allow in from 10.101.150.80 comment 'Access from Server 2'

# IPv6 rules
sudo ufw allow in proto tcp to any port 80 from Anywhere \(v6\)
comment 'HTTP (IPv6)'
sudo ufw allow in proto tcp to any port 443 from Anywhere \(v6\)
```

```
comment 'HTTPS (IPv6)'
sudo ufw allow in proto tcp to any port 5666 from Anywhere \(v6\)
comment 'Nagios NRPE (IPv6)'
sudo ufw allow in proto tcp to any port 22 from Anywhere \(v6\)
comment 'SSH (IPv6)'
sudo ufw allow in proto udp to any port 53 from Anywhere \(v6\)
comment 'DNS UDP (IPv6)'
sudo ufw allow in proto tcp to any port 53 from Anywhere \(v6\)
comment 'DNS TCP (IPv6)'
sudo ufw allow in proto tcp to any port 111 from Anywhere \(v6\)
comment 'RPC (IPv6)'
```

Allow Outbound Traffic:

```
# DNS resolution
sudo ufw allow out proto udp to 10.101.150.80 port 53 comment 'DNS
queries to Server 2 (UDP)'
sudo ufw allow out proto tcp to 10.101.150.80 port 53 comment 'DNS
queries to Server 2 (TCP)'

sudo ufw allow out to 8.8.8.8 port 53 proto udp

sudo ufw allow out to 8.8.4.4 port 53 proto udp

# Service monitoring
sudo ufw allow out proto tcp to 10.101.150.78 port 80 comment 'HTTP
monitoring to Server 0'
sudo ufw allow out proto tcp to 10.101.150.80 port 445 comment 'SMB
monitoring to Server 2'
sudo ufw allow out proto tcp to any port 5666 comment 'NRPE
monitoring'
sudo ufw allow out proto udp to any port 53 comment 'DNS queries
(UDP)'
sudo ufw allow out proto tcp to any port 53 comment 'DNS queries
(TCP)'
sudo ufw allow out proto tcp to any port 22 comment 'SSH outbound'

# IPv6 outbound
sudo ufw allow out proto tcp to any port 5666 from Anywhere \(v6\)
comment 'NRPE monitoring (IPv6)'
sudo ufw allow out proto udp to any port 53 from Anywhere \(v6\)
comment 'DNS queries UDP (IPv6)'
```

```
sudo ufw allow out proto tcp to any port 53 from Anywhere \(v6\)
comment 'DNS queries TCP (IPv6)'
```

```
sudo ufw allow out to 8.8.8.8 port 53 proto udp
```

sudo ufw allow out to 8.8.4.4 port 53 proto udp

```
sudo ufw allow out proto tcp to any port 22 from Anywhere \(v6\)
comment 'SSH outbound (IPv6)'
```

Summary:

1. It works as a web server (ports 80 and 443) hosting the Nagios interface.
2. Running Nagios monitoring services (port 5666 for NRPE)
3. It also has DNS services (TCP/UDP port 53)
4. Communicates with servers 10.101.150.78 and 10.101.150.80
5. You have RPC rules (port 111) suggesting you may be running NFS or other RPC-based services
6. Has complete configuration for IPv4 and IPv6

3.3.3 Server 2 (10.101.150.80) — DNS & Samba

Allow Inbound Traffic:

```
# DNS service
sudo ufw allow in proto tcp to any port 53 comment 'DNS queries
(TCP)'
sudo ufw allow in proto udp to any port 53 comment 'DNS queries
(UDP)'

# File sharing
sudo ufw allow in proto tcp to any port 445 comment 'SMB/CIFS'

# Administrative access
sudo ufw allow in proto tcp to any port 22 comment 'SSH Access'

# DNS service - restricted to local network
sudo ufw allow in proto udp from 10.101.150.0/24 to any port 53
comment 'DNS queries (UDP) from LAN'
sudo ufw allow in proto tcp from 10.101.150.0/24 to any port 53
comment 'DNS queries (TCP) from LAN'
```

```
# Web services
sudo ufw allow in proto tcp to any port 80 comment 'Allow HTTP'
sudo ufw allow in proto tcp to any port 443 comment 'Allow HTTPS'

# Monitoring
sudo ufw allow in proto tcp from 10.101.150.79 to any port 5666
comment 'NRPE from Nagios'

# IPv6 rules
sudo ufw allow in proto tcp to any port 53 from Anywhere \(v6\)
comment 'DNS TCP (IPv6)'
sudo ufw allow in proto udp to any port 53 from Anywhere \(v6\)
comment 'DNS UDP (IPv6)'
sudo ufw allow in proto tcp to any port 445 from Anywhere \(v6\)
comment 'SMB/CIFS (IPv6)'
sudo ufw allow in proto tcp to any port 22 from Anywhere \(v6\)
comment 'SSH (IPv6)'
sudo ufw allow in proto tcp to any port 80 from Anywhere \(v6\)
comment 'Allow HTTP (IPv6)'
sudo ufw allow in proto tcp to any port 443 from Anywhere \(v6\)
comment 'Allow HTTPS (IPv6)'
```

Allow Outbound Traffic:

```
# Outbound access
sudo ufw allow out proto tcp to any port 22 comment 'SSH outbound'
sudo ufw allow out proto tcp to any port 53 comment 'DNS TCP
outbound'
sudo ufw allow out proto udp to any port 53 comment 'DNS UDP
outbound'
sudo ufw allow out proto tcp to 10.101.150.78 port 80 comment 'HTTP
to Server 0'
sudo ufw allow out proto udp to 8.8.8.8 port 53 comment 'DNS queries
to Google DNS'
sudo ufw allow out proto udp to 8.8.4.4 port 53 comment 'DNS queries
to Google DNS'

# IPv6 outbound
sudo ufw allow out proto tcp to any port 22 from Anywhere \(v6\)
comment 'SSH outbound (IPv6)'
sudo ufw allow out proto tcp to any port 53 from Anywhere \(v6\)
```

```
comment 'DNS TCP outbound (IPv6)'
```

Summary:

1. It works as a DNS server (port 53 TCP/UDP), with specific permissions for the local network 10.101.150.0/24
2. Provides Samba file sharing services (TCP port 445)
3. Runs web services (HTTP/HTTPS on ports 80/443)
4. It is being monitored by Nagios from machine 10.101.150.79 (port 5666)
5. It has configuration to route DNS queries to Google public servers (8.8.8.8 and 8.8.4.4)
6. Communicates with server 10.101.150.78 via HTTP (port 80)
7. Has complete configuration for IPv4 and IPv6

3.4 Activation and Verification

After configuring all rules, enable and verify the firewall on each server:

```
# Enable the firewall
sudo ufw enable

# Verify the configuration
sudo ufw status numbered


# Disable the firewall
sudo ufw disable
```

Remove Incorrect Rules:

If there is any rule implemented in the firewall that blocks services or is unnecessary, it can be removed with:

```
#list the rules

sudo ufw status numbered
```

```
# Identify rule numbers for DNS inbound traffic
sudo ufw delete [rule_number]
```

Important: *Always maintain an active SSH session when making firewall changes. If a configuration error blocks SSH access, you can use the active session to correct it without being locked out.*

3.5 Connectivity tests

```
# SSH (where applicable)
ssh cgsadmin@<Server_IP >

# HTTP/S

curl -k http://owncloud.group10.private/status.php
(https://<IP DO SERVIDOR>/)

# DNS
dig @10.101.150.80 owncloud.group10.private +short

# Samba
smbclient -L //samba.group10.private/ -U cgsadmin
```

4. Service Deployment & Configuration

4.1 ownCloud & MariaDB (Server 0)

4.1.1 Prerequisites and System Preparation

Before installing ownCloud and MariaDB, prepare the system with necessary dependencies and optimizations:

- **Operating System:** A stable Linux distribution such as Ubuntu Server.
- **Dependencies:**
    - **Web Server:** Apache2
    - **PHP modules:** php-gd, php-json, php-curl, php-xml, php-mbstring, etc.

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:ondrej/php
sudo apt update
sudo apt install apache2 libapache2-mod-php7.4 php7.4 php7.4-
cli php7.4-common php7.4-curl php7.4-gd php7.4-json php7.4-
mbstring php7.4-mysql php7.4-xml php7.4-zip php7.4-intl
php7.4-bcmath php7.4-gmp php7.4-imagick php7.4-ldap php7.4-
redis php-apcu -y
sudo a2enmod php7.4 rewrite headers env dir mime
sudo systemctl restart apache2
```

Obs.: Currently the system is using a different PHP version (php8.0+), in which it would be necessary to replace the package names.

4.1.2 MariaDB Database Install and Configuration

The MariaDB server will support ownCloud by providing its database backend.

```
# Install MariaDB

sudo apt-get install mariadb-server

# Secure MariaDB installation

sudo mysql_secure_installation
```

Next, create the ownCloud database and user:

```
# Access MariaDB as root
sudo mysql -u root –p
```

Password: ‘cgsg10s78admin’

```
# In the MariaDB prompt, execute:
CREATE DATABASE owncloud;
CREATE USER 'owncloud user'@'localhost' IDENTIFIED BY
'your_secure_password';
GRANT ALL PRIVILEGES ON owncloud.* TO 'owncloud user'@'localhost';
```

```
FLUSH PRIVILEGES;
EXIT;
```

Restart services to ensure that all configuration changes take effect:

```
sudo systemctl restart apache2
sudo systemctl restart mariadb
```

Since MariaDB was installed along with the Apache/PHP package installation above, verify it is running:

```
sudo systemctl status mariadb
```

sudo systemctl status apache2

4.1.3 ownCloud Installation

Download the latest ownCloud package from the official repository:

```
# Download the latest stable release
wget https://download.owncloud.com/server/stable/owncloud-complete-latest.zip
```

#Unzip the downloaded archive:

```
unzip owncloud-complete-latest.zip
```

Move the extracted ownCloud folder into Apache's web directory:

```
sudo mv owncloud /var/www/html/

# Set correct ownership and permissions
sudo chown -R www-data:www-data /var/www/html/owncloud
sudo chmod -R 755 /var/www/html/owncloud

# Create data directory outside web root for security
sudo mkdir -p /var/owncloud/data
sudo chown -R www-data:www-data /var/owncloud/data
sudo chmod -R 750 /var/owncloud/data
```

4.1.4 Apache Configuration for ownCloud

Create a dedicated Apache virtual host for ownCloud:

```
sudo nano /etc/apache2/sites-available/owncloud.conf
```

Add the following configuration:

```
<VirtualHost *:80>
    ServerName owncloud.group10.private
    ServerAlias 10.101.150.78
    DocumentRoot /var/www/html/owncloud/

    <Directory /var/www/html/owncloud/>
        Options +FollowSymlinks
        AllowOverride All
        Require all granted

        <IfModule mod_dav.c>
            Dav off
        </IfModule>

        SetEnv HOME /var/www/html/owncloud
        SetEnv HTTP_HOME /var/www/html/owncloud
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/owncloud_error.log
    CustomLog ${APACHE_LOG_DIR}/owncloud_access.log combined
</VirtualHost>
```

Access Owncloud

```
ssh cgsadmin@10.101.150.78 -L 80:localhost:80
User:'owncloud user'

Password:'your_secure_password'
```

Enable necessary Apache modules and the ownCloud site:

```
sudo a2ensite owncloud.conf
```

```
sudo a2enmod rewrite
curl http://localhost/owncloud
sudo systemctl restart apache2
```

4.2 DNS Server (Server 2)

4.2.1 BIND9 Installation

```
# Install BIND9 packages
sudo apt update
sudo apt install bind9 bind9utils bind9-doc dnsutils -y
```

4.2.2 Basic BIND9 Configuration

Configure BIND9 options:

```
sudo nano /etc/bind/named.conf.options
```

Replace the content with:

```
options {
        directory "/var/cache/bind";

        listen-on { 10.101.150.80; 127.0.0.1; };  // Different IP to
avoid conflict
        allow-query { localhost; };

    allow-recursion {
        localhost;
        10.101.150.0/24;
    };

        recursion yes;
        forward only;
```

```
        forwarders {
        8.8.8.8;
        8.8.4.4;
    };
```

- In the `listen-on` parameter, make sure that the field <`127.0.0.1`> is different from 127.0.0.53, as to not conflict with the default DNS provider.
- When BIND can't resolve a DNS query it forwards it to the default one at 127.0.0.53

4.2.3 Zone Configuration

Create the zone configuration:

```
sudo nano /etc/bind/named.conf.local
```

Add the following:

```
zone "group10.private" {
    type master;
    file "/etc/bind/db.group10.private";
    allow-query { any; };
};
```

- Do not use the suffix "`.local`" in order to avoid conflicts with the default settings. Hence why we used the "`.private`".
- Be careful when specifying a file path, watch out for spaces, since `"/etc/bind/db.group10.private"` is not the same as `"/etc/bind/db.group10.private "`.

4.2.4 Create Zone Files

Create the forward zone file:

```
sudo nano /etc/bind/db.group10.private
```

Add the following content:

```
$TTL    604800
@       IN      SOA     ns1.group10.private. admin.group10.private.
(
                        2025041801 ; Serial YYYYMMDDNN - NN é um
incrementador de versao
                        604800     ; Refresh
                        86400      ; Retry
                        2419200    ; Expire
                        604800 )   ; Negative Cache TTL
;
@       IN      NS      ns1.group10.private.
ns1     IN      A       10.101.150.80

; Host records
owncloud        IN      A       10.101.150.78
web             IN      A       10.101.150.79
samba           IN      A       10.101.150.80
mariadb         IN      A       10.101.150.78
```

4.2.5 Other Configs

Bind Server

Change the `resolved.conf` file:

```
Sudo nano /etc/systemd/resolved.conf
```

Replace the content at the end with the following:

```
DNS=127.0.0.53  # Default resolver
Domains=~group10.private  # Route this domain to BIND
DNSStubListener=yes
```

Other VM's of the cluster

Change the `resolved.conf` file:

```
Sudo nano /etc/systemd/resolved.conf
```

Replace the content at the end with the following:

```
DNS=127.0.0.53  # Default resolver
Domains=~group10.private  # Route this domain to BIND
```

Change the `resolv.conf` file:

```
Sudo nano /etc/resolv.conf
```

And replace its contents with:

```
nameserver 10.101.150.80  # Your BIND server IP
options edns0
search group10.private
```

- The file /etc/resolv.conf keeps getting overwritten, so you need to lock it with the following commands:
  `sudo ls -l /etc/resolv.conf  #check the actual path of the symlink`

  `sudo chattr +i /run/systemd/resolve/stub-resolv.conf  #lock the real file`

4.2.6 Validate and Restart BIND9

Check configuration syntax:

```
sudo named-checkconf
sudo named-checkzone group10.private
```

Restart BIND9:

```
sudo systemctl restart bind9
sudo systemctl status bind9
```

Test DNS resolution:

```
# Test forward lookup
dig @10.101.150.80 owncloud.group10.private
```

4.3 Web Server (Server 1)

4.3.1 Apache Installation

```
sudo apt update
sudo apt install apache2 -y
```

4.3.2 Basic Apache Configuration

```
# Enable necessary modules
sudo a2enmod rewrite
sudo a2enmod ssl
sudo a2enmod headers

# Configure main Apache settings
sudo nano /etc/apache2/conf-available/security.conf
```

Add or modify these settings:

```
ServerTokens Prod
ServerSignature Off
```

```
TraceEnable Off
```

4.3.3 Virtual Host Configuration

Create a virtual host for the web server:

```
sudo nano /etc/apache2/sites-available/web.group10.private.conf
```

Add this configuration:

```
<VirtualHost *:80>
    ServerName web.group10.private
    ServerAlias 10.101.150.79
    DocumentRoot /var/www/html/web.group10.private

    ErrorLog ${APACHE_LOG_DIR}/web_error.log
    CustomLog ${APACHE_LOG_DIR}/web_access.log combined

    <Directory /var/www/html/web.group10.private>
        Options -Indexes +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Create the document root directory:

```
sudo mkdir -p /var/www/html/web.group10.private
sudo chown -R www-data:www-data /var/www/html/web.group10.private
```

Create a simple index page:

```
echo "<html><body><h1>Group 10 Web Server</h1><p>This is the main
web server for Group 10 FCUL infrastructure.</p></body></html>" |
sudo tee /var/www/html/web.group10.private/index.html
```

Enable the site and restart Apache:

```
sudo a2ensite web.group10.private.conf
sudo systemctl restart apache2
```

4.4 Samba File Server (Server 2)

Provides file sharing across Windows and Linux clients using the SMB/CIFS protocol.

4.4.1 Samba Installation

```
sudo apt update
sudo apt install samba smbclient -y
```

4.4.2 Samba Configuration

Back up the original configuration for precaution:

```
sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.orig
```

Create a new configuration:

```
sudo nano /etc/samba/smb.conf
```

Add this configuration:

```
[SharedDocs]

comment = Shared Documents for Group 10

path = /srv/samba/shared_docs

browsable = yes

writable = yes

valid users = @group10

create mask = 0640

directory mask = 0750
```

4.4.3 User and Group Configuration (Linux)

Create a dedicated group and configure users:

```
# Create group
sudo groupadd group10

# Add existing user to group
sudo usermod -aG group10 cgsadmin

# Create directory structure
sudo mkdir -p /srv/samba/shared_docs
sudo chown -R cgsadmin:group10 /srv/samba/shared_docs
sudo chmod -R 0750 /srv/samba/shared_docs
```

4.4.4 Create Samba Users

Add the existing system user to Samba:

```
sudo smbpasswd -a cgsadmin
```

Enter a password when prompted:

```
Samba10admin
```

4.4.5 Some operations in Samba

To open an interactive:

```
smbclient -L //10.101.150.80/SharedDocs -U cgsadmin
```

```
These are some of the basic commands available:
put <local_file.txt>                # Upload file
get <remote_file.txt>               # Download file
mkdir <new_folder>                  # Create directory
del <file_to_dellete.txt>           # Delete file
```

4.4.6 Restart and Test Samba

```
sudo systemctl restart smbd
sudo systemctl restart nmbd

# Verify the configuration
testparm

# Test access from another server
# (From Server 0 or Server 1)
smbclient -L //10.101.150.80/ -U cgsadmin
```

4.5 MariaDB Advanced Configuration (Server 0)

MariaDB was initially set up in Section 5.1.2, but additional configuration is needed for optimal performance and security:

4.5.1 Enhanced Security Configuration

Edit the MariaDB configuration file:

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

Add or modify these settings under the `[mysqld]` section:

```
# Network settings
bind-address = 127.0.0.1  # Only allow local connections

# Security settings
local-infile = 0
symbolic-links = 0

# Performance optimizations
key_buffer_size = 64M
max_allowed_packet = 64M
table_open_cache = 400
sort_buffer_size = 4M
net_buffer_length = 8K
read_buffer_size = 2M
```

```
read_rnd_buffer_size = 2M
myisam_sort_buffer_size = 64M
```

4.5.2 Automated Backup Configuration

Create a backup script:

```
sudo nano /usr/local/bin/mariadb-backup.sh
```

Add the following content:

```
#!/bin/bash

TIMESTAMP=$(date +"%Y%m%d-%H%M%S")
BACKUP_DIR="/var/backups/mariadb"
MYSQL_USER="root"
MYSQL_PASSWORD="YourRootPasswordHere"

# Create backup directory if it doesn't exist
mkdir -p $BACKUP_DIR

# Dump all databases
mysqldump --user=$MYSQL_USER --password=$MYSQL_PASSWORD --all-
databases --single-transaction \
  --quick --lock-tables=false > $BACKUP_DIR/full-backup-
$TIMESTAMP.sql

# Dump owncloud database separately (for quicker restores)
mysqldump --user=$MYSQL_USER --password=$MYSQL_PASSWORD --databases
owncloud --single-transaction \
  --quick --lock-tables=false > $BACKUP_DIR/owncloud-backup-
$TIMESTAMP.sql

# Compress the backups
gzip $BACKUP_DIR/full-backup-$TIMESTAMP.sql
gzip $BACKUP_DIR/owncloud-backup-$TIMESTAMP.sql

# Remove backups older than 7 days
find $BACKUP_DIR -type f -name "*.gz" -mtime +7 -delete
```

Make the script executable:

```
sudo chmod +x /usr/local/bin/mariadb-backup.sh
```

Create a cron job for daily backups:

```
sudo nano /etc/cron.d/mariadb-backup
```

Add this line:

```
0 2 * * * root /usr/local/bin/mariadb-backup.sh > /var/log/mariadb-backup.log 2>&1
```

5. Centralized Monitoring with Nagios (Server 1)

5.1 Nagios Core Installation

5.1.1 Prerequisites

Install dependencies:

```
sudo apt update
sudo apt install wget unzip curl openssl build-essential libgd-dev libssl-dev libapache2-mod-php php-gd php apache2 -y
```

5.1.2 Nagios Instalation

Note: adapted from https://docs.vultr.com/install-nagios-on-ubuntu-20-04

Download Nagios Core Setup files. To download the latest version, visit the official releases site.

```
wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.9.tar.gz
```

Extract the downloaded files.

```
sudo tar -zxvf nagios-4.5.9.tar.gz
```

Navigate to the setup directory.

```
cd nagios-4.5.9
```

Run the Nagios Core configure script.

```
sudo ./configure
```

Compile the main program and CGIs.

```
sudo make all
```

Make and install group and user.

```
sudo make install-groups-users
```

Add www-data directories user to the nagios group.

```
sudo usermod -a -G nagios www-data
```

Install Nagios.

```
sudo make install
```

Initialize all the installation configuration scripts.

```
sudo make install-init
```

Install and configure permissions on the configs' directory.

```
sudo make install-commandmode
```

Install sample config files.

```
sudo make install-config
```

Install apache files.

```
sudo make install-webconf
```

Enable apache rewrite mode.

```
sudo a2enmod rewrite
```

Enable CGI config.

```
sudo a2enmod cgi
```

Restart the Apache service.

```
sudo systemctl restart apache2
```

Create a user and set the password when prompted.

```
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users admin
```

Initiate Nagios on startup

```
sudo crontab –e
```

Add the line

```
@reboot systemctl start nagios
```

5.1.3 Nagios Core plugins

Download the Nagios Core plugin. To download the latest plugins, visit the plugins download page.

```
cd ~/
```

```
wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz
```

Extract the downloaded plugin.

```
sudo tar -zxvf nagios-plugins-2.3.3.tar.gz
```

Navigate to the plugins' directory.

```
cd nagios-plugins-2.3.3/
```

Run the plugin configure script.

```
sudo ./configure --with-nagios-user=nagios --with-nagios-
group=nagios
```

Compile Nagios Core plugins.

```
sudo make
```

Install the plugins.

```
sudo make install
```

To verify the Nagios Core configuration.

```
sudo /usr/local/nagios/bin/nagios -v
/usr/local/nagios/etc/nagios.cfg
```

5.1.4 Nagios Add remote host to be monitored

On the host side:

Install NRPE and Plugins

```
sudo apt update
sudo apt install nagios-nrpe-server monitoring-plugins
```

Open the NRPE config:

```
sudo nano /etc/nagios/nrpe.cfg
```

Make sure to add the Nagios server IP to the allowed_hosts :

```
allowed_hosts=127.0.0.1,<NAGIOS_SERVER_IP>
```

Ensure NRPE is listening externally:

```
server_address= <HOST_IP>
```

Restart the NRPE service:

```
sudo systemctl restart nagios-nrpe-server
sudo systemctl enable nagios-nrpe-server
```

On the NAGIOS side:

Add Host Configuration (an example of a file can be found in the following link)

```
sudo nano /usr/local/nagios/etc/servers/<HOSTNAME>.cfg
```

Verify and Restart Nagios

```
sudo /usr/local/nagios/bin/nagios -v
/usr/local/nagios/etc/nagios.cfg

sudo systemctl restart nagios
```

5.1.5 Nagios systemctl commands

Enable and start the Nagios service:

```
sudo systemctl enable nagios
sudo systemctl start nagios
sudo systemctl status nagios
```

5.1.6 Monitor the localhost machine (where nagios is installed)

Uncomment the line:

`cfg_file=/usr/local/nagios/etc/objects/localhost.cfg`

On the file `/usr/local/nagios/etc/nagios.cfg`

Define the services to be checked:

```
sudo nano /usr/local/nagios/etc/objects/localhost.cfg


define host {


    use                     linux-server            ; Name of host
template to use

                                                    ; This host definition
will inherit all variables that are defined

                                                    ; in (or inherited by)
the linux-server host template definition.
    host_name               localhost

    alias                   Nagios + Apache

    address                 127.0.0.1

    max_check_attempts      5

}
```

```
##############################################################################
#####
#
# HOST GROUP DEFINITION
#
##############################################################################
#####


# Define an optional hostgroup for Linux machines


define hostgroup {


    hostgroup_name          linux-servers           ; The name of the
hostgroup
    alias                   VMs FCUL                ; Long name of the
group
    members                 localhost,host80,host78                ; Comma
separated list of hosts that belong to this group
}




##############################################################################
#####
#
# SERVICE DEFINITIONS
#
##############################################################################
#####


# Define a service to "ping" the local machine
```

```
define service {

    use                     local-service           ; Name of service
template to use
    host_name               localhost
    service_description     PING
    check_command           check_ping!100.0,20%!500.0,60%
}

define service {
      use local-service
      host_name localhost
      service_description UFW Integrity Check
      check_command check_ufw_integrity
}


# Define a service to check the disk space of the root partition
# on the local machine.  Warning if < 20% free, critical if
# < 10% free space on partition.

define service {

    use                     local-service           ; Name of service
template to use
    host_name               localhost
    service_description     Root Partition
    check_command           check_local_disk!20%!10%!/
}
```

```
# Define a service to check the number of currently logged in
# users on the local machine.  Warning if > 20 users, critical
# if > 50 users.

define service {

    use                     local-service           ; Name of service
template to use
    host_name               localhost
    service_description     Current Users
    check_command           check_local_users!20!50
}



# Define a service to check the number of currently running procs
# on the local machine.  Warning if > 250 processes, critical if
# > 400 processes.

define service {

    use                     local-service           ; Name of service
template to use
    host_name               localhost
    service_description     Total Processes
    check_command           check_local_procs!250!400!RSZDT
}



# Define a service to check the load on the local machine.
```

```
define service {

    use                     local-service           ; Name of service
template to use
    host_name               localhost
    service_description     Current Load
    check_command           check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
}


# Define a service to check the swap usage the local machine.
# Critical if less than 10% of swap is free, warning if less than 20% is
free

define service {

    use                     local-service           ; Name of service
template to use
    host_name               localhost
    service_description     Swap Usage
    check_command           check_local_swap!20%!10%
}


# Define a service to check SSH on the local machine.
# Disable notifications for this service by default, as not all users may
have SSH enabled.

define service {

    use                     local-service           ; Name of service
template to use
```

```
    host_name                   localhost
    service_description         SSH Availability
    check_command               check_ssh
    notifications_enabled       0
}




# Define a service to check HTTP on the local machine.
# Disable notifications for this service by default, as not all users may
have HTTP enabled.


define service {


    use                         local-service           ; Name of service
template to use
    host_name                   localhost
    service_description         HTTP Apache
    check_command               check_http
    notifications_enabled       0
}
```

5.2 NRPE (Nagios Remote Plugin Executor) Configuration

NRPE allows Nagios to execute monitoring commands on remote hosts. It must be installed on all three servers.

5.2.1 NRPE Installation on Monitored Servers (Servers 0 & 2)

On each server to be monitored (Servers 0 and 2), install NRPE by following the steps described on the section 6.1.4.

5.2.2 Configure Nagios Server to Monitor Remote Hosts

Pre-requisites

Uncomment the line:

`cfg_file=/usr/local/nagios/etc/servers`

On the file `/usr/local/nagios/etc/nagios.cfg`

Define the commands that are going to be used to perform the checks

```
sudo cat /usr/local/nagios/etc/objects/commands.cfg
```

```
define command {

    command_name    check_nrpe

    command_line    /usr/local/nagios/libexec/check_nrpe -H
$HOSTADDRESS$ -c $ARG1$

}


define command {

      command_name check_ufw_integrity

      command_line /usr/lib/nagios/plugins/check_ufw_integrity.sh

}




define command {

    command_name    check_smb_share

    command_line    /usr/lib/nagios/plugins/check_disk_smb -H
$HOSTADDRESS$ -s $ARG1$ -u $ARG2$ -p $ARG3$
```

```
}


define command {

    command_name    check_dns

    command_line    $USER1$/check_dns -H $ARG1$

}


define command{

        command_name    check_mariadb_sql

        command_line    /usr/lib/nagios/plugins/check_mysql_query.pl
-H 10.101.150.78 $ARG1$

}


 define command {


    command_name    notify-host-by-email

    command_line    /usr/bin/printf "%b" "***** Nagios
*****\n\nNotification Type: $NOTIFICATIONTYPE$\nHost:
$HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo:
$HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" | /bin/mail -s "**
$NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ **"
$CONTACTEMAIL$

}


define command {


    command_name    notify-service-by-email
```

```
    command_line    /usr/bin/printf "%b" "***** Nagios
*****\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService:
$SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState:
$SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional
Info:\n\n$SERVICEOUTPUT$\n" | /bin/mail -s "** $NOTIFICATIONTYPE$
Service Alert: $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **"
$CONTACTEMAIL$

}


define command {


    command_name    check-host-alive

    command_line    $USER1$/check_ping -H $HOSTADDRESS$ -w
3000.0,80% -c 5000.0,100% -p 5

}


define command {


    command_name    check_local_disk

    command_line    $USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$

}


define command {


    command_name    check_local_load

    command_line    $USER1$/check_load -w $ARG1$ -c $ARG2$

}
```

```
define command {

    command_name    check_local_procs
    command_line    $USER1$/check_procs -w $ARG1$ -c $ARG2$ -s
$ARG3$
}

define command {

    command_name    check_local_users
    command_line    $USER1$/check_users -w $ARG1$ -c $ARG2$
}

define command {

    command_name    check_local_swap
    command_line    $USER1$/check_swap -w $ARG1$ -c $ARG2$
}

define command {

    command_name    check_local_mrtgtraf
    command_line    $USER1$/check_mrtgtraf -F $ARG1$ -a $ARG2$ -w
$ARG3$ -c $ARG4$ -e $ARG5$
}
```

```
 define command {

    command_name    check_ftp
    command_line    $USER1$/check_ftp -H $HOSTADDRESS$ $ARG1$
}

define command {

    command_name    check_hpjd
    command_line    $USER1$/check_hpjd -H $HOSTADDRESS$ $ARG1$
}

define command {

    command_name    check_snmp
    command_line    $USER1$/check_snmp -H $HOSTADDRESS$ $ARG1$
}

define command {

    command_name    check_http
    command_line    $USER1$/check_http -I $HOSTADDRESS$ $ARG1$
}

define command {
```

```
    command_name    check_ssh
    command_line    $USER1$/check_ssh $ARG1$ $HOSTADDRESS$
}


define command {

    command_name    check_dhcp
    command_line    $USER1$/check_dhcp $ARG1$
}

define command {

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

define command {

    command_name    check_pop
    command_line    $USER1$/check_pop -H $HOSTADDRESS$ $ARG1$
}
```

```
define command {

    command_name    check_imap
    command_line    $USER1$/check_imap -H $HOSTADDRESS$ $ARG1$
}

define command {

    command_name    check_smtp
    command_line    $USER1$/check_smtp -H $HOSTADDRESS$ $ARG1$
}

define command {

    command_name    check_tcp
    command_line    $USER1$/check_tcp -H $HOSTADDRESS$ -p $ARG1$
$ARG2$
}

define command {

    command_name    check_udp
    command_line    $USER1$/check_udp -H $HOSTADDRESS$ -p $ARG1$
$ARG2$
}
```

```
define command {

    command_name    check_nt

    command_line    $USER1$/check_nt -H $HOSTADDRESS$ -p 12489 -v
$ARG1$ $ARG2$

}

define command {

    command_name    process-host-perfdata

    command_line    /usr/bin/printf "%b"
"$LASTHOSTCHECK$\t$HOSTNAME$\t$HOSTSTATE$\t$HOSTATTEMPT$\t$HOSTSTATE
TYPE$\t$HOSTEXECUTIONTIME$\t$HOSTOUTPUT$\t$HOSTPERFDATA$\n" >>
/usr/local/nagios/var/host-perfdata.out

}

define command {

    command_name    process-service-perfdata

    command_line    /usr/bin/printf "%b"
"$LASTSERVICECHECK$\t$HOSTNAME$\t$SERVICEDESC$\t$SERVICESTATE$\t$SER
VICEATTEMPT$\t$SERVICESTATETYPE$\t$SERVICEEXECUTIONTIME$\t$SERVICELA
TENCY$\t$SERVICEOUTPUT$\t$SERVICEPERFDATA$\n" >>
/usr/local/nagios/var/service-perfdata.out

}
```

For the custom custom plugin also create the file /usr/local/nagios/libexec/check_ufw_integrity.sh:

```
#!/bin/bash
```

```
#Expected hash value

EXPECTED_HASH="<PASTE_THE_HASH_HERE>"


#Get current UFW rule status and compute its hash

CURRENT_HASH=$(sudo ufw status numbered | sha256sum | awk '{print
$1}')


#Compare hashes

if [ "$CURRENT_HASH" == "$EXPECTED_HASH" ]; then

     echo "Match"

     exit 0

else

     echo "Error"

     echo "Expected: $EXPECTED_HASH"

     echo "Current: $CURRENT_HASH"

     exit 2

fi
```

Calculate the hash with the command

```
sudo ufw status numbered | sha256sum | awk '{print $1}
```

Give the necessary permissions

```
sudo chown nagios:nagios
/usr/lib/nagios/plugins/check_ufw_integrity.sh

sudo chmod 755 /usr/lib/nagios/plugins/check_ufw_integrity.sh
```

Add the following line to the sudoers file (sudo visudo)

```
nagios ALL=(ALL) NOPASSWD: /usr/bin/ufw
```

Restart Nrpe plugin and Nagios

For Server 0 (ownCloud & MariaDB):

To monitor the mariaDB we had to download a custom plugin that the community made:

On the Nagios machine.

Download the perl script https://raw.githubusercontent.com/harisekhon/nagios-plugins/master/check_mysql_query.pl

Allow execution

```
chmod +x check_mysql_query.pl
```

Install dependencies

```
sudo apt install libjson-perl libjson-pp-perl
```

Download tlds file

```
sudo wget -O /usr/lib/nagios/plugins/resources/tlds-alpha-by-domain.txt https://data.iana.org/TLD/tlds-alpha-by-domain.txt
sudo chown nagios:nagios /usr/lib/nagios/plugins/resources/tlds-alpha-by-domain.txt
sudo chmod 644 /usr/lib/nagios/plugins/resources/tlds-alpha-by-domain.txt
```

Create the config file

```
sudo nano /usr/local/nagios/etc/servers/10.101.150.78.cfg
```

```
define host {
    use                     linux-server
    host_name               host78
    alias                   Owncloud + MariaDB
    address                 10.101.150.78
    max_check_attempts      5
}
```

```
#######################################################################
#
# SERVICE DEFINITIONS
#
#######################################################################

# NRPE-based service checks for remotehost

define service {
    use                     local-service
    host_name               host78
    service_description     Current Load
```

```
    check_command           check_nrpe!check_load
}

define service {
      use local-service
      host_name host78
      service_description UFW Integrity Check
      check_command check_nrpe!check_ufw_integrity
}



define service {
    use                     local-service
    host_name               host78
    service_description     Disk Usage
    check_command           check_nrpe!check_disk
}

define service {
    use                     local-service
    host_name               host78
    service_description     Current Users
    check_command           check_nrpe!check_users
}

define service {
    use                     local-service
    host_name               host78
```

```
    service_description     Total Processes
    check_command           check_nrpe!check_procs
}

define service {
    use                     local-service
    host_name               host78
    service_description     Swap Usage
    check_command           check_nrpe!check_swap
}

define service {
    use                     local-service
    host_name               host78
    service_description     SSH Availability
    check_command           check_ssh
    notifications_enabled   0
}

define service {
    use                     local-service
    host_name               host78
    service_description     HTTP OwnCloud Availability
    check_command           check_http
    notifications_enabled   0
}

define service {
    use                 generic-service
```

```
    host_name           host78

    service_description MariaDB Check

    check_command       check_mariadb_sql!-u owncloud user -p
your_secure_password -d information_schema -q 'SELECT
ROUND(VARIABLE_VALUE/1024/1024) FROM
information_schema.GLOBAL_STATUS WHERE VARIABLE_NAME =
"Memory_used"' -w 512 -c 1024

}
```

For the custom custom plugin also create the file /usr/local/nagios/libexec/check_ufw_integrity.sh:

```
#!/bin/bash

#Expected hash value

EXPECTED_HASH="<PASTE_THE_HASH_HERE>"



#Get current UFW rule status and compute its hash

CURRENT_HASH=$(sudo ufw status numbered | sha256sum | awk '{print
$1}')



#Compare hashes

if [ "$CURRENT_HASH" == "$EXPECTED_HASH" ]; then

     echo "Match"

     exit 0

else

     echo "Error"

     echo "Expected: $EXPECTED_HASH"

     echo "Current: $CURRENT_HASH"

     exit 2
```

```
fi
```

Calculate the hash with the command

```
sudo ufw status numbered | sha256sum | awk '{print $1}
```

Give the necessary permissions

```
sudo chown nagios:nagios
/usr/lib/nagios/plugins/check_ufw_integrity.sh
```

```
sudo chmod 755 /usr/lib/nagios/plugins/check_ufw_integrity.sh
```

Add the following line to the sudoers file (sudo visudo)

```
nagios ALL=(ALL) NOPASSWD: /usr/bin/ufw
```

Add the following line to /etc/nagios/nrpe.cfg

```
command[check_ufw_integrity]=/usr/lib/nagios/plugins/check_ufw_integ
rity.sh
```

For Server 2 (DNS & Samba):

Create the config file

```
sudo nano /usr/local/nagios/etc/servers/10.101.150.80.cfg
```

```
define host {
    use                     linux-server
    host_name               host80
    alias                   DNS Server(BIND9) + Samba Server
    address                 10.101.150.80
```

```
    max_check_attempts      5
}




################################################################################
#####
#
# SERVICE DEFINITIONS
#
################################################################################
#####


# NRPE-based service checks for remotehost


define service {
    use                     local-service
    host_name               host80
    service_description     Current Load
    check_command           check_nrpe!check_load
}


define service {
      use local-service
      host_name host80
      service_description UFW Integrity Check
      check_command check_nrpe!check_ufw_integrity
}



define service {
```

```
    use                     local-service
    host_name               host80
    service_description     Disk Usage
    check_command           check_nrpe!check_disk
}

define service {
    use                     local-service
    host_name               host80
    service_description     Current Users
    check_command           check_nrpe!check_users
}

define service {
    use                     local-service
    host_name               host80
    service_description     Total Processes
    check_command           check_nrpe!check_procs
}

define service {
    use                     local-service
    host_name               host80
    service_description     Swap Usage
    check_command           check_nrpe!check_swap
}

define service {
    use                     local-service
    host_name               host80
    service_description     SSH Availability
    check_command           check_ssh
```

```
    notifications_enabled   0
}

define service {
    use                 generic-service
    host_name           host80
    service_description DNS Resolution Test External - Google
    check_command       check_dns!google.com
}

define service {
    use                 generic-service
    host_name           host80
    service_description DNS Resolution Test Internal - Owncloud
    check_command       check_dns!owncloud.group10.private
}
define service {
    use                 generic-service
    host_name           host80
    service_description Samba Share Availability
    check_command       check_smb_share!SharedDocs!cgsadmin!Samba10admin
}
```

For the custom custom plugin also create the file /usr/local/nagios/libexec/check_ufw_integrity.sh:

```
#!/bin/bash

#Expected hash value

EXPECTED_HASH="<PASTE_THE_HASH_HERE>"


#Get current UFW rule status and compute its hash
```

```
CURRENT_HASH=$(sudo ufw status numbered | sha256sum | awk '{print
$1}')


#Compare hashes

if [ "$CURRENT_HASH" == "$EXPECTED_HASH" ]; then

     echo "Match"

     exit 0

else

     echo "Error"

     echo "Expected: $EXPECTED_HASH"

     echo "Current: $CURRENT_HASH"

     exit 2

fi
```

Calculate the hash with the command

```
sudo ufw status numbered | sha256sum | awk '{print $1}
```

Give the necessary permissions

```
sudo chown nagios:nagios
/usr/lib/nagios/plugins/check_ufw_integrity.sh

sudo chmod 755 /usr/lib/nagios/plugins/check_ufw_integrity.sh
```

Add the following line to the sudoers file (sudo visudo)

```
nagios ALL=(ALL) NOPASSWD: /usr/bin/ufw
```

Add the following line to /etc/nagios/nrpe.cfg

```
command[check_ufw_integrity]=/usr/lib/nagios/plugins/check_ufw_integrity.sh
```

Restart Nrpe plugin and Nagios

5.3 Acess Nagius Dashboard

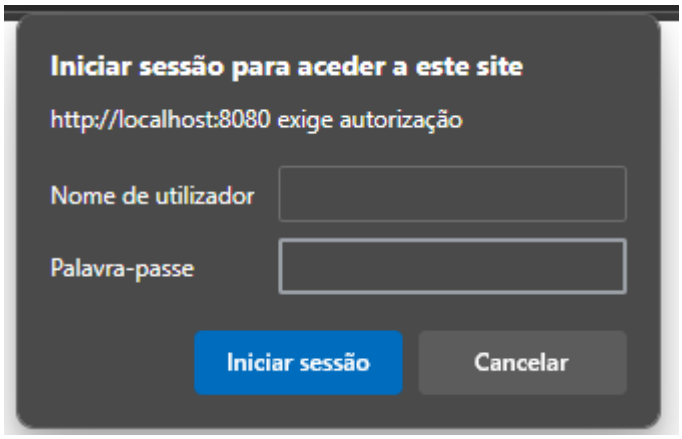
To access the **Nagius dashboard** (GUI) we need to run the ssh on server2 (10.101.150.79) with a special flag:

```
ssh cgsadmin@10.101.150.79 -L 8080:localhost:80
```

Now in the PC used to connect through ssh go to a browser and access:

```
http://127.0.0.1:8080/nagios/
```

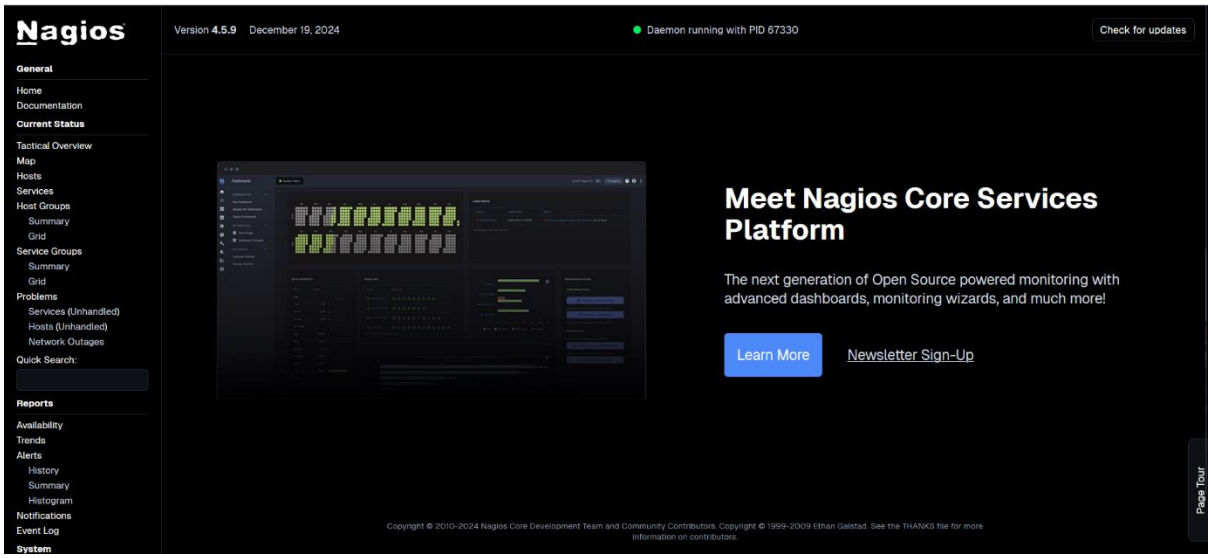
It will prompt a User and Password:



User: `admin`

Password: `admin79MGHTMIWGY`

Now the dashboard should appear:



5.4. Groups and other Info

5.4.1. Names, Aliases and Parents

The information about the hosts displayed in Nagios can be changed in these files:

```
sudo nano /usr/local/nagios/etc/servers/<10.101.150.80.cfg>
```

For the host running Nagios go to this file instead:

```
sudo nano /usr/local/nagios/etc/objects/localhost.cfg
```

5.4.2. Host Groups

To create a **Host Group** edit the following file:

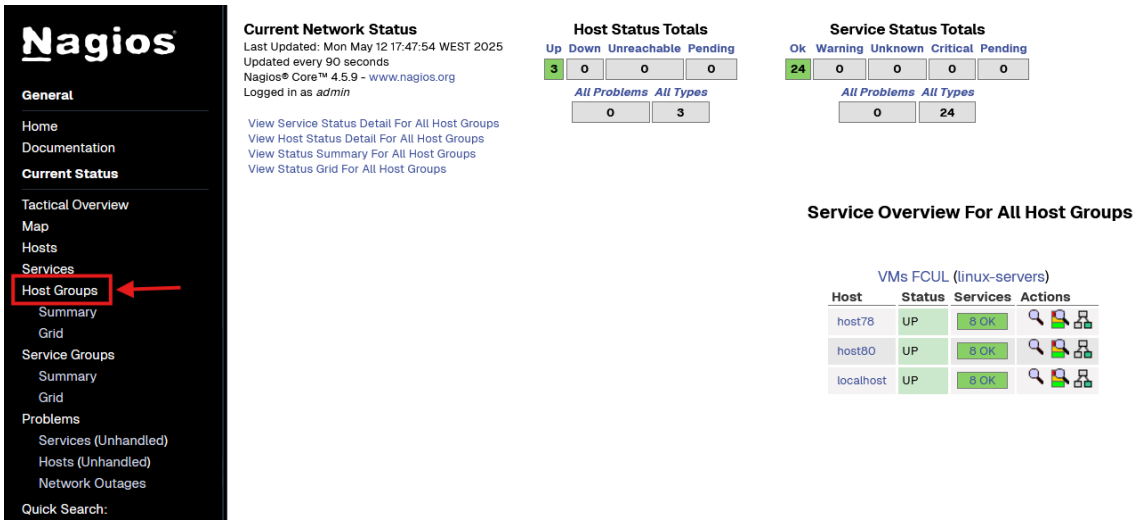
```
sudo nano /usr/local/nagios/etc/objects/localhost.cfg
```

And add an entry below **Host Group Definition**, like in the image:

```
###############################################################################
#
# HOST GROUP DEFINITION
#
###############################################################################

# Define an optional hostgroup for Linux machines

define hostgroup {

    hostgroup_name          linux-servers           ; The name of the hostgroup
    alias                   VMs FCUL                ; Long name of the group
    members                 localhost,host80,host78                ; Comma separated list of hosts tha
t belong to this group
}
```

Be careful when changing the `hostgoup name` because that's the name used to refer to the group. For human identification purposes the `alias` can be freely changed.

Now all these Hosts will appear grouped in Nagios Dashboard under the tab **Host Groups**



5.4.3. Service Groups

The services can also be grouped.

First create the file:

```
sudo nano /usr/local/nagios/etc/objects/servicegroups.cfg
```

And edit its contents:

```
define servicegroup {

    servicegroup_name    web_services
```

```
    alias               All Web Services
    members             host78, HTTP OwnCloud
Availability,localhost, HTTP Apache
}

define servicegroup {
    servicegroup_name   ssh_test
    alias               SSH General Check
    members             host78, SSH Availability,localhost, SSH
Availability, host80, SSH Availability
}

define servicegroup {
    servicegroup_name   DNS_services
    alias               All DNS Services
    members             host80, DNS Resolution Test External -
Google, host80, DNS Resolution Test Internal - Owncloud
}

define servicegroup {
    servicegroup_name   disk_Storage
    alias               General Root Storage Check
    members             host78, Disk Usage, localhost, Root
Partition, host80, Disk Usage
}

define servicegroup {
    servicegroup_name   users
    alias               General Current Users Check
```

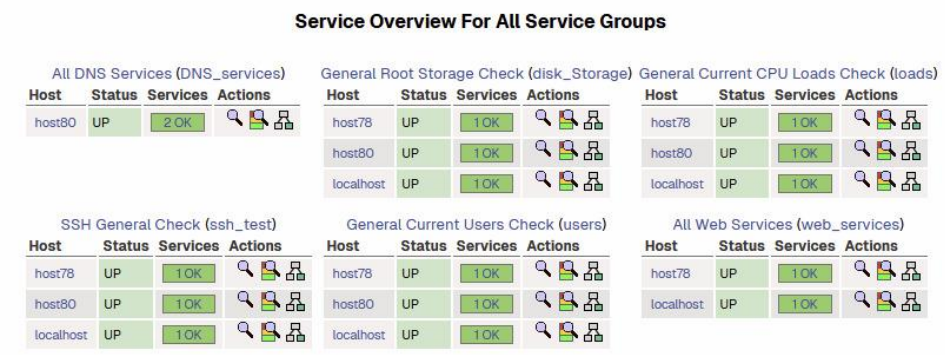
```
    members                 host78, Current Users, localhost, Current
Users, host80, Current Users
}


define servicegroup {
    servicegroup_name       loads
    alias                   General Current CPU Loads Check
    members                 host78, Current Load, localhost, Current
Load, host80, Current Load
}
```

Then add its path to the main Nagios config file for the settings to take effect:

```
sudo nano /usr/local/nagios/etc/nagios.cfg
```

```
# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg

# Definitions for monitoring services in the Hosts
cfg_file=/usr/local/nagios/etc/objects/servicegroups.cfg
```

Just like the Hosts, the services will now appear in groups in the tab **Service Groups:**



6. Final Testing & Demo Checklists

6.1. Testing Procedures

- **Service Functionality:**
  - Test SSH access, ownCloud file operations, DNS resolution, web server content delivery, Samba share availability, and MariaDB connectivity.
- **Nagios Monitoring:**
  - Verify all host and service checks are active. Simulate service failures to observe alerting (including custom lock counts).
  - Use this script to check the if nagios is monitoring the webservs correctly:

```
#!/bin/bash

# Target web server (change if needed)
URL="http://localhost"

# Number of requests to simulate
TOTAL_REQUESTS=100
DELAY=0.1  # Delay in seconds between requests

echo "Sending $TOTAL_REQUESTS requests to $URL..."

for i in $(seq 1 $TOTAL_REQUESTS); do
  curl -s -o /dev/null "$URL"
  echo "Request $i sent"
  sleep $DELAY
done

echo "Done!"
```

- **Automation and Reconfiguration Tests:**

    - Run your Ansible playbooks or scripts in a test environment to validate reproducibility of the configuration.
- **Backup and Restore:**
    - Confirm that database backups and configuration file backups are created successfully and can be restored when needed.

6.2. Final Demo Checklist

- Verified SSH access with updated and secured credentials.
- ownCloud is fully operational and connects to MariaDB.
- DNS resolves all required service names (ownCloud, web, Samba, MariaDB).
- The web server serves content appropriately.
- Samba shares are accessible and correctly permissioned.
- MariaDB is running, and automated backups are scheduled.
- Nagios monitors every component.
- The repository is up to date with all configuration files, scripts, and documentation.
- Full documentation is available and organized in this Wiki with a clear version history.