Name: Kshitij Hundre
Div: D15C
Roll No:18

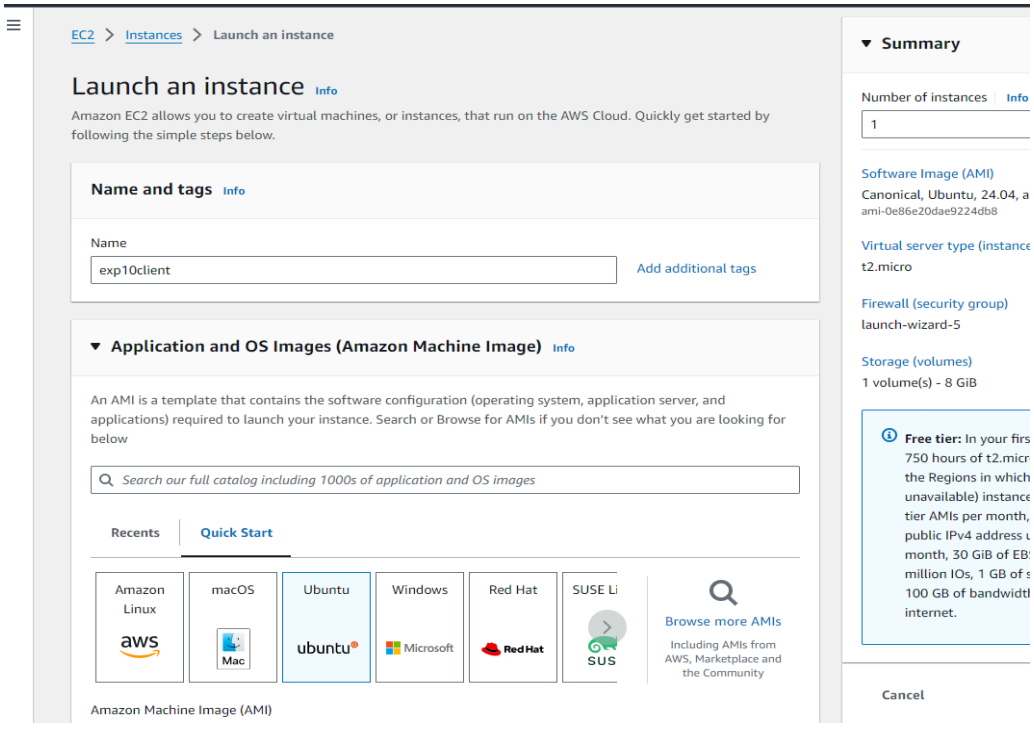
Adv Devops Exp:10

Aim :
The aim of this experiment is to set up and configure Nagios for comprehensive monitoring of ports, services, and both Windows and Linux servers. The objective is to ensure real-time monitoring, detect potential issues, and provide timely alerts for system administrators to take preventive or corrective actions, ensuring optimal system performance and uptime.

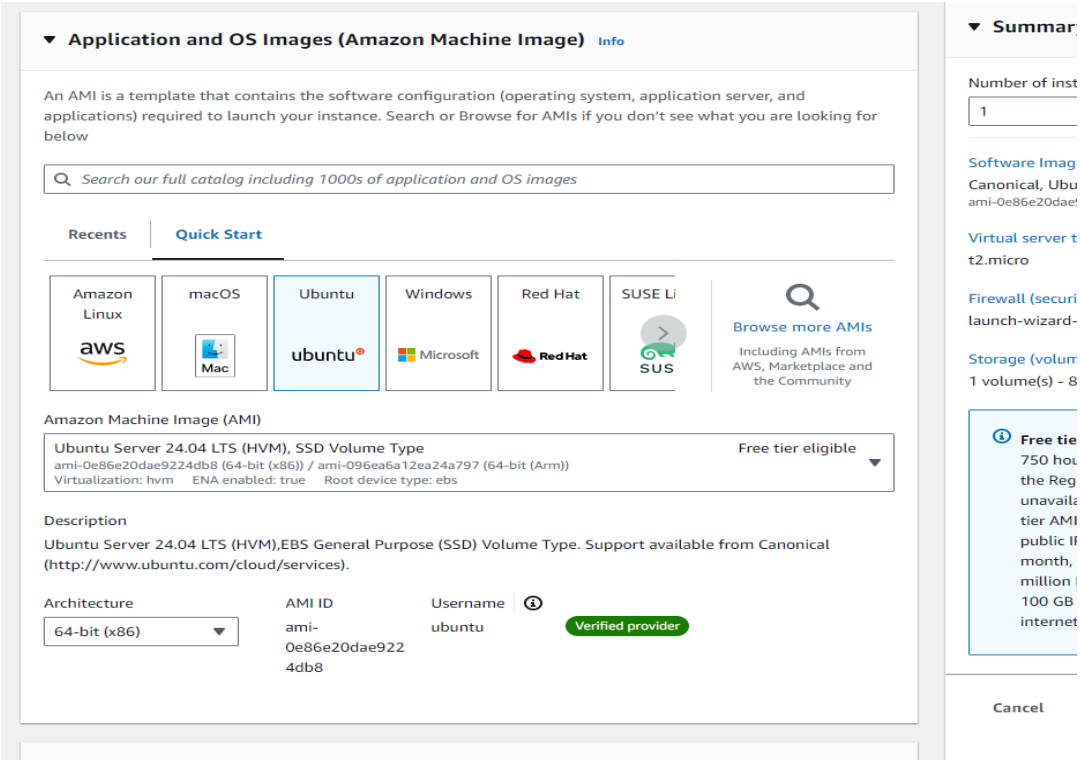
Steps

1) Launch an instance

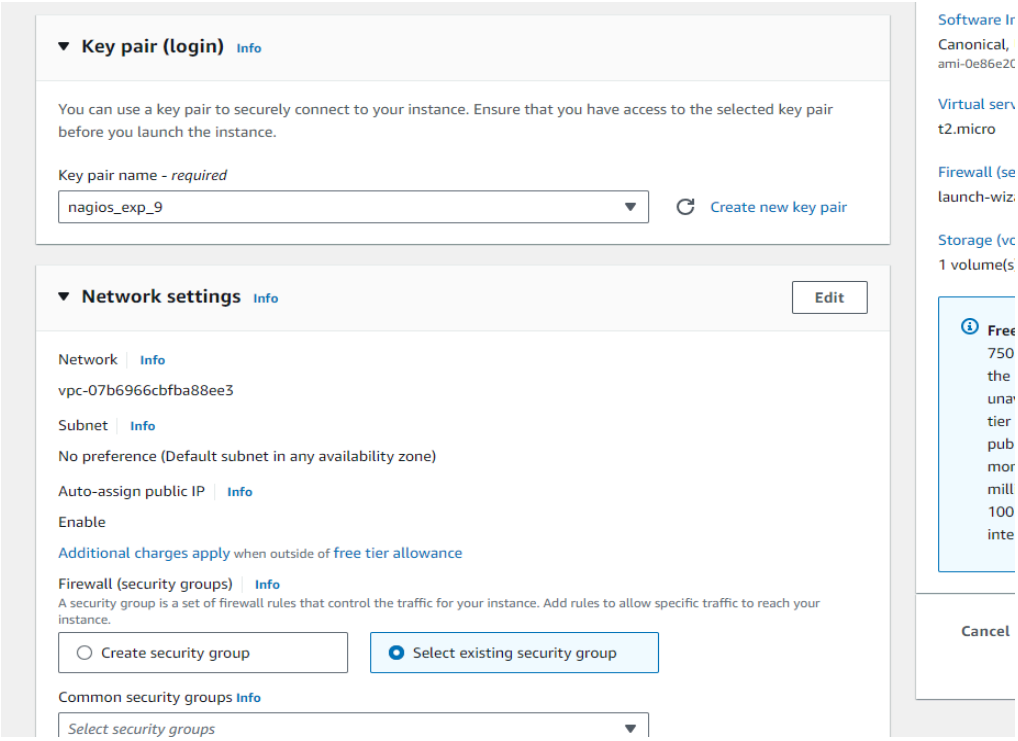
Launch an ec2 instance.
Select Ubuntu as the OS to give a meaningful name of the instance.



Select the same security group as given to the exp9 machine.

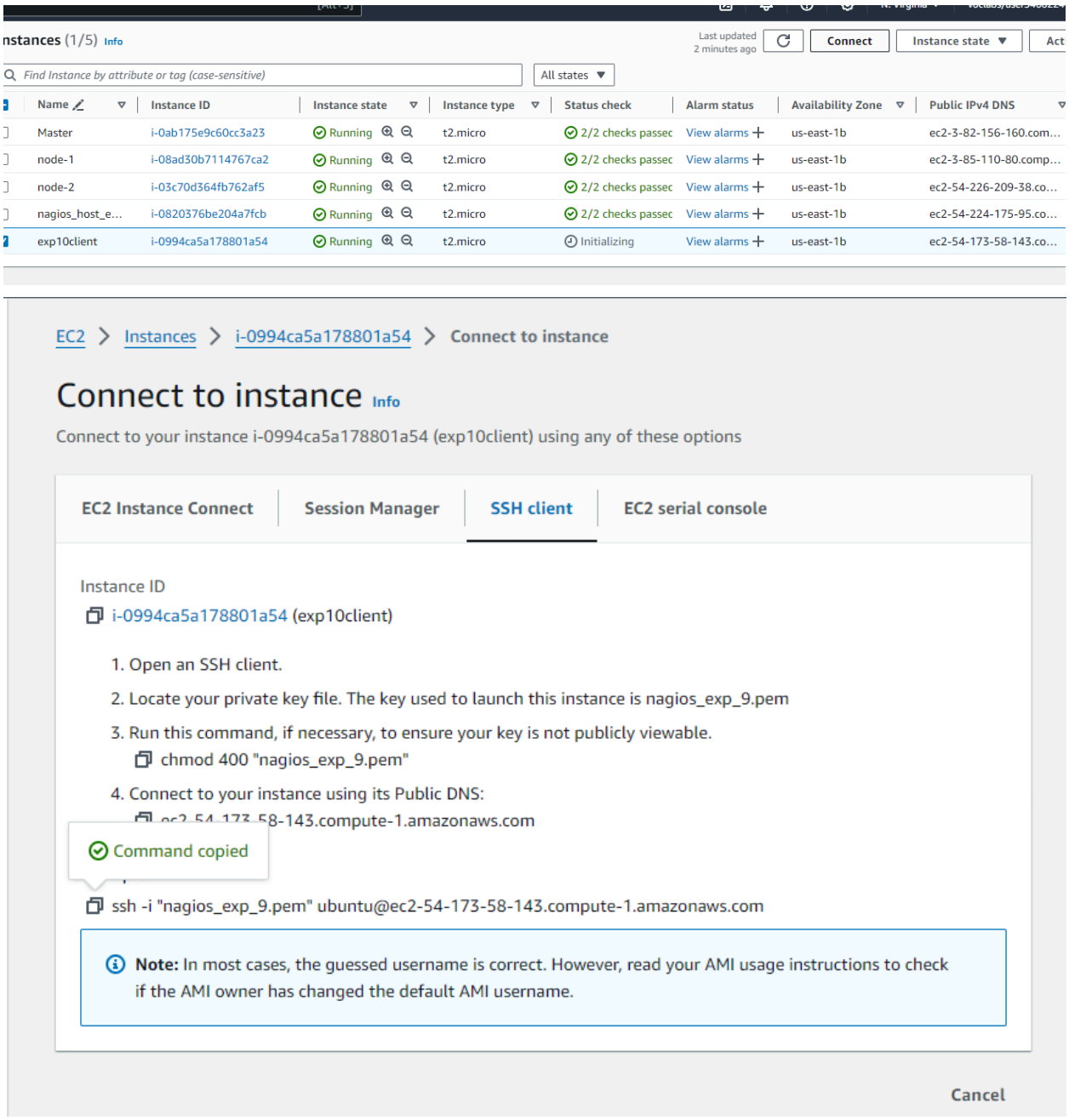


Make sure to select the same key-pair login used in the exp9 machine.

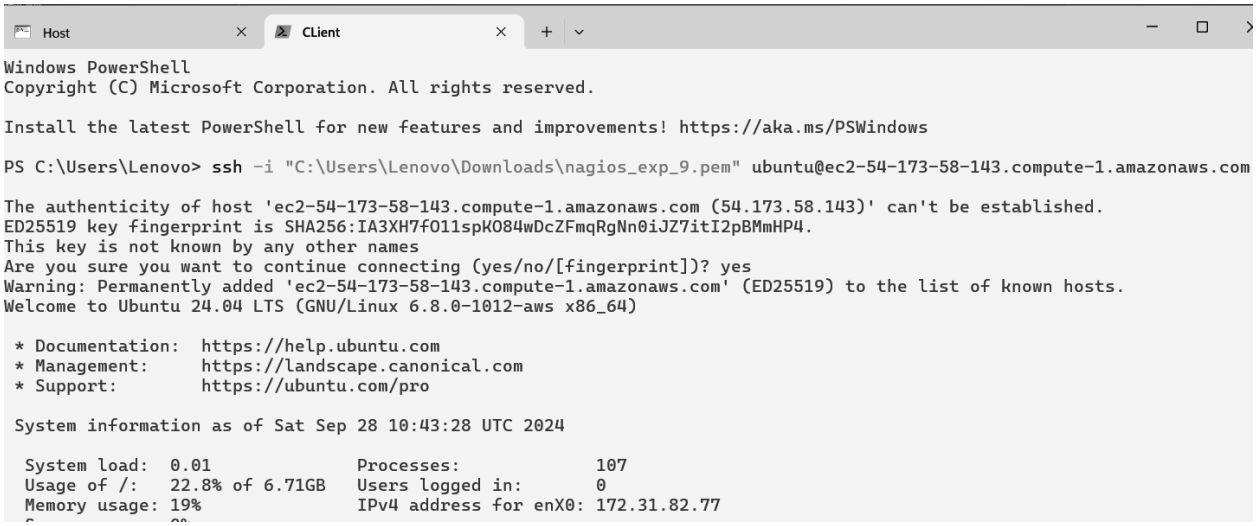


click on launch instance.

Now connect with this client machine using the ssh through your terminal(open a new terminal in your local machine and we will need both of the terminals open)



Note to change the path of the .pem file.



2) Go to nagios host machine (Host machine)

Perform the following commands

ps -ef | grep nagios

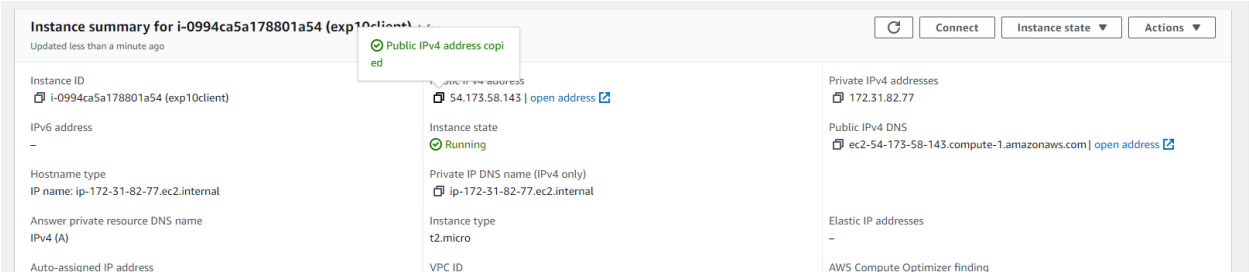
sudo su
mkdir -p /usr/local/nagios/etc/objects/monitorhosts/linuxhosts

cp /usr/local/nagios/etc/objects/localhost.cfg
/usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

nano /usr/local/nagios/etc/objects/monitorhosts/linuxhosts/linuxserver.cfg

Change hostname and alias to linuxserver
Change address to public ip address of client instance (Ubuntu instance) you can get the

ip address by clicking on the instance id on the instances section there you will get the public ipv4 address




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

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will in>
                                                    ; in (or inherited by) the lin>
    host_name               linuxserver
    alias                   linuxserver
    address                 54.173.58.143
}
```

Change hostgroup_name to linux-servers1

```
# Define an optional hostgroup for Linux machines

define hostgroup {

    hostgroup_name          linux-servers1          ; The name of the hostgroup
    alias                   Linux Servers           ; Long name of the group
    members                 localhost               ; Comma separated list of host>
}
```

Change the occurrences of hostname further in the document from localhost to linuxserver
example like:

```
    host_name                       localhost
```

changed to

```
define service {

    use                     local-service           ; Name of service templat
    host_name               linuxserver
    service_description     PING
    check_command           check_ping!100.0,20%!500.0,60%
}
```

This is the last one

```
define service {

    use                     local-service           ; Name of service template to >
    host_name               linuxserver
    service_description     HTTP
    check_command           check_http
    notifications_enabled   0
```

now ctrl+O and enter to save and then ctrl+X for exiting.

Open nagios configuration file and add the line shown below
nano /usr/local/nagios/etc/nagios.cfg

```
[root@ip-172-31-80-137 ec2-user]# nano /usr/local/nagios/etc/nagios.cfg
```

##Add this line below the opened nano interface where similar lines are commented.
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/

```
 GNU nano 5.8                                                  /usr/local/nagios/etc/nagios.cfg
# 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 the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/localhost.cfg

# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg

# Definitions for monitoring a router/switch
#cfg_file=/usr/local/nagios/etc/objects/switch.cfg

# Definitions for monitoring a network printer
#cfg_file=/usr/local/nagios/etc/objects/printer.cfg


# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:

#cfg_dir=/usr/local/nagios/etc/servers
#cfg_dir=/usr/local/nagios/etc/printers
#cfg_dir=/usr/local/nagios/etc/switches
#cfg_dir=/usr/local/nagios/etc/routers
cfg_dir=/usr/local/nagios/etc/objects/monitorhosts/



# OBJECT CACHE FILE
# This option determines where object definitions are cached when
# Nagios starts/restarts.  The CGIs read object definitions from
```

ctrl+o and enter for saving and ctrl+x to exit nano editor.

Verify configuration files

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

```
[root@ip-172-31-80-137 ec2-user]# /usr/local/nagios/bin/nagios -v /usr/local/nagios
/etc/nagios.cfg

Nagios Core 4.5.5
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-09-17
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects
```

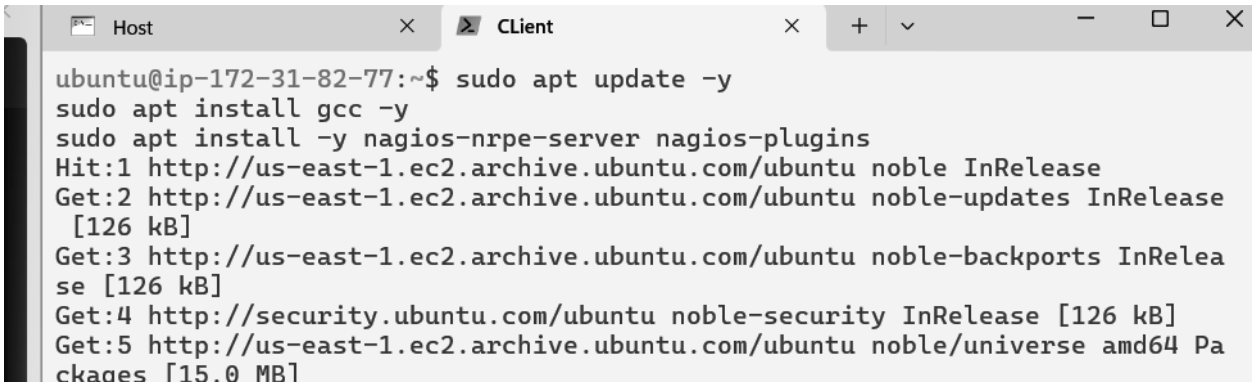
```
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-80-137 ec2-user]#
```

Restart nagios service.
service nagios restart

```
Things look okay - No serious problems were detected during the pre-flight check
[root@ip-172-31-80-137 ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@ip-172-31-80-137 ec2-user]#
```

3) Go to client machine (ubuntu machine)

Perform the following commands
sudo apt update -y
sudo apt install gcc -y
sudo apt install -y nagios-nrpe-server nagios-plugins



```
Running kernel seems to be up-to-date.

Restarting services...

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
 ubuntu @ session #1: sshd[990,1101]
 ubuntu @ user manager service: systemd[996]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-82-77:~$
```

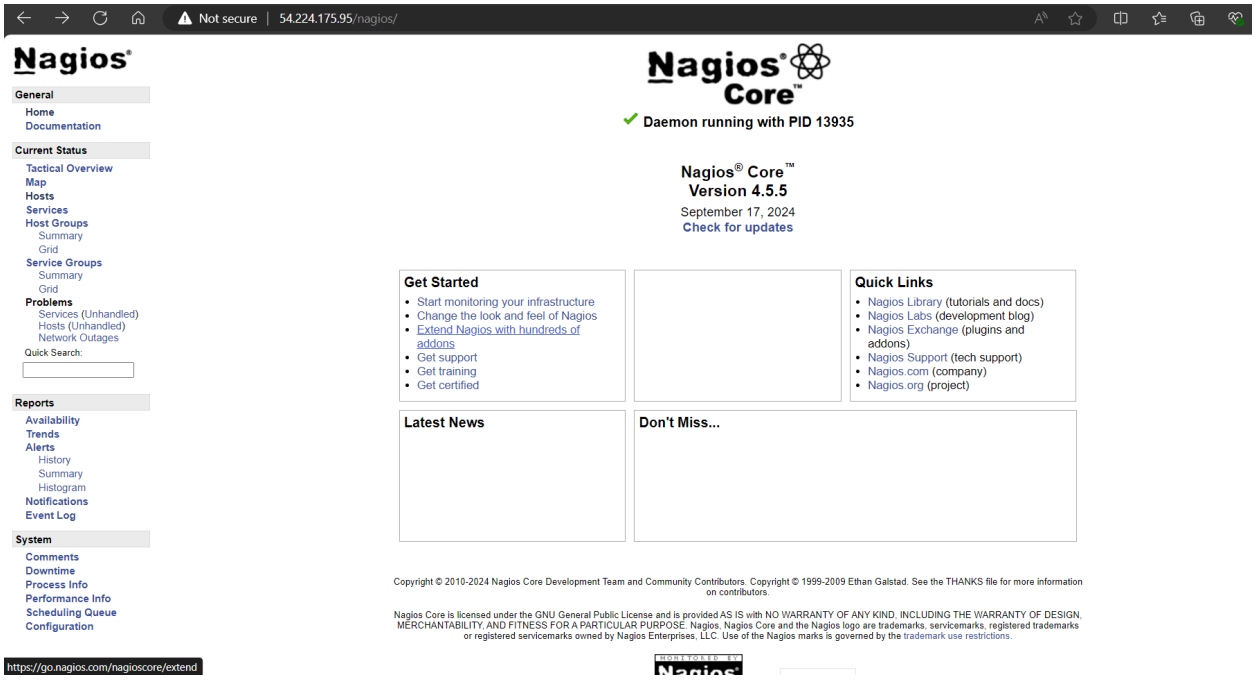
Open the nrpe.cfg file in nano editor
sudo nano /etc/nagios/nrpe.cfg

Under allowed_hosts, add the nagios host ip address (public)

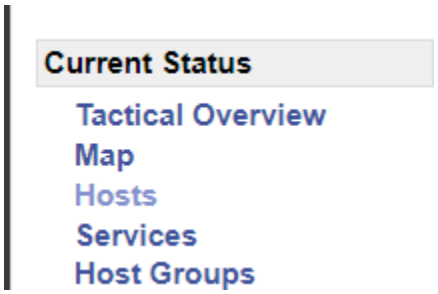
```
# You can either supply a username or a UID.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xin>

nrpe_user=nagios



# NRPE GROUP
# This determines the effective group that the NRPE daemon should run as.
# You can either supply a group name or a GID.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xin>

nrpe_group=nagios



# ALLOWED HOST ADDRESSES
# This is an optional comma-delimited list of IP address or hostnames
# that are allowed to talk to the NRPE daemon. Network addresses with a bit>
# (i.e. 192.168.1.0/24) are also supported. Hostname wildcards are not curr>
# supported.
#
# Note: The daemon only does rudimentary checking of the client's IP
# address.  I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xin>

allowed_hosts=127.0.0.1,54.224.175.95



# COMMAND ARGUMENT PROCESSING
# This option determines whether or not the NRPE daemon will allow clients
```

again save and exit the nano editor.

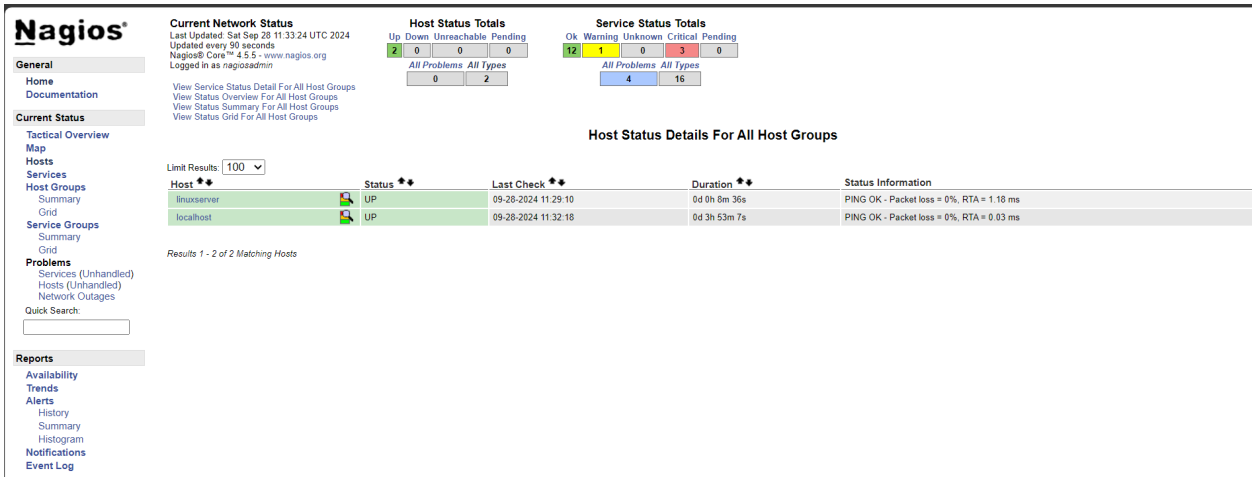
4) Go to nagios dashboard and click on hosts

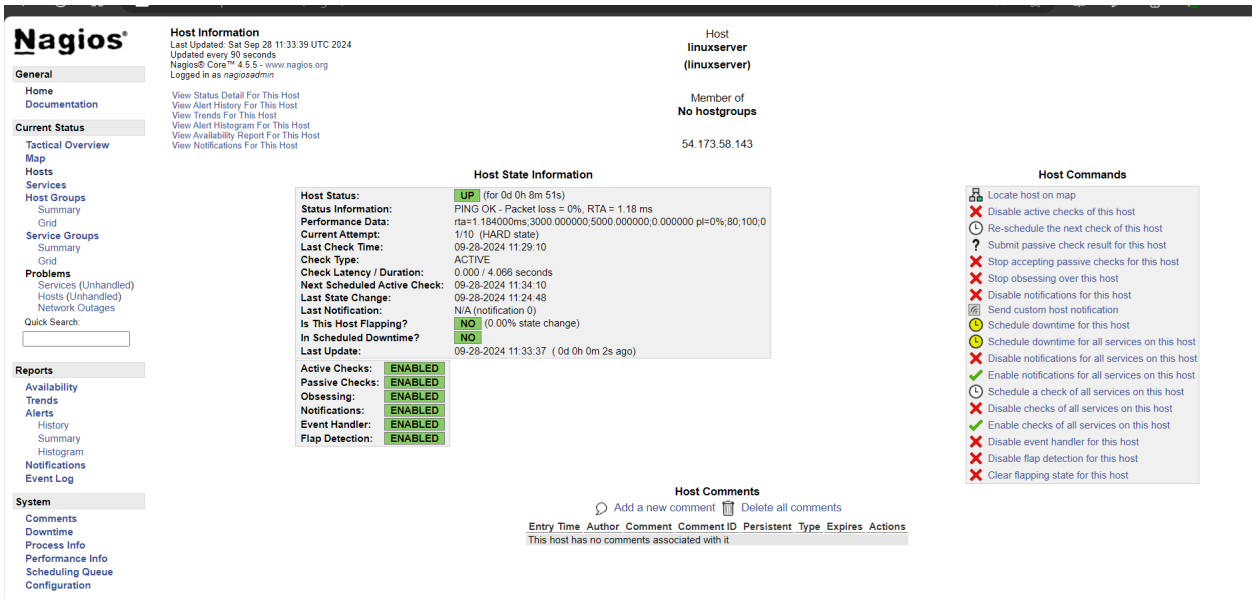


Click on hosts

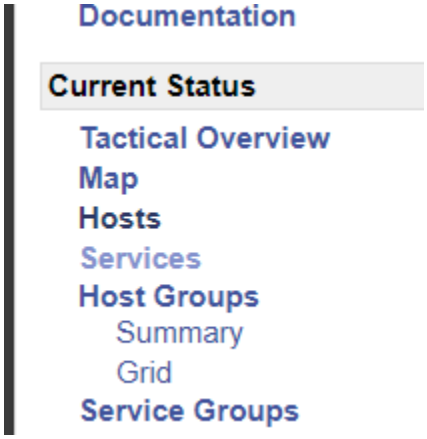


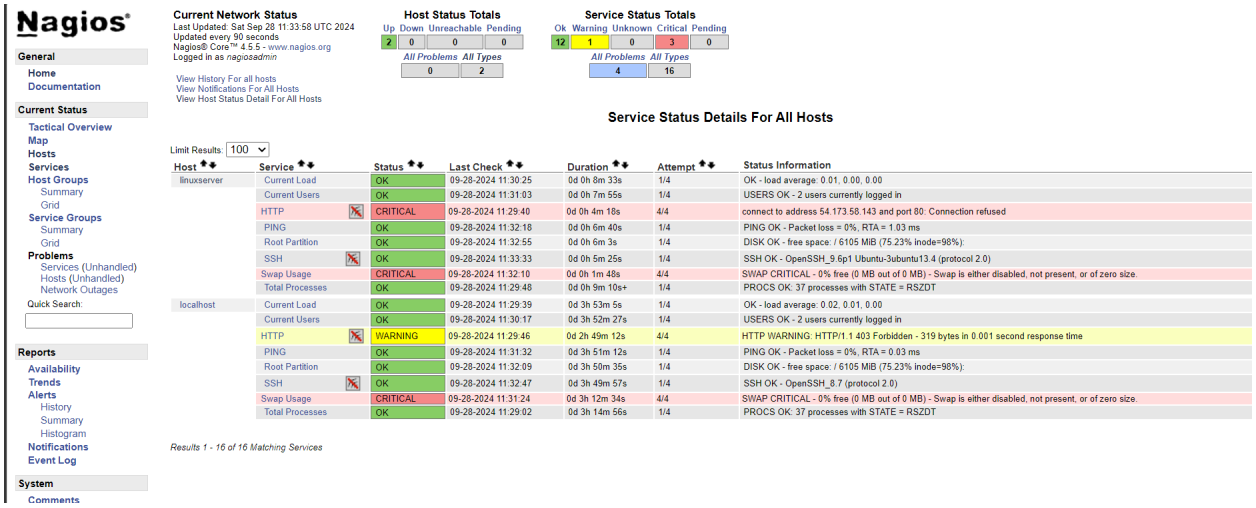
5) Click on linux server





6) Click on nagios services





Conclusion:

In this experiment, we successfully set up Nagios to monitor a remote Linux server (Ubuntu instance) from a Nagios host machine, both operating within an EC2 environment. The process involved careful configuration of both the Nagios host (referred to as the "exp9 machine") and the remote client machine, allowing for effective real-time monitoring of server performance. By launching an Ubuntu EC2 instance as the client machine, we ensured that consistent security group policies and key-pairs were maintained, enabling seamless SSH access to both machines for configuration purposes.

The configuration on the Nagios host involved creating a new directory for the remote client's monitoring configuration, where we modified the **localhost.cfg** file to reflect the IP address of the remote client, adjusted the hostgroup, and ensured proper integration with the Nagios setup.

We also verified these changes through a Nagios configuration check before restarting the service, ensuring that the Nagios dashboard was ready to monitor the remote machine.

On the client side, we installed the **nagios-nrpe-server** and **nagios-plugins**, which allowed communication between the Nagios host and client machine. Correctly configuring the **nrpe.cfg** file, particularly the **allowed_hosts** directive, was crucial in establishing connectivity. Once the configuration was complete and the Nagios service was restarted, we successfully confirmed that the remote Linux server was being monitored on the Nagios dashboard, achieving the experiment's objective.

This experiment not only highlighted the steps for setting up monitoring between a Nagios host and a remote Linux client but also demonstrated key troubleshooting techniques. By resolving common issues such as restarting the Apache server on the Nagios host, ensuring proper directory structure, and configuring the NRPE correctly on the client machine, we gained valuable insights into server monitoring and network management using Nagios. The hands-on experience provided a deeper understanding of how Nagios can be effectively used for real-world infrastructure monitoring.

.