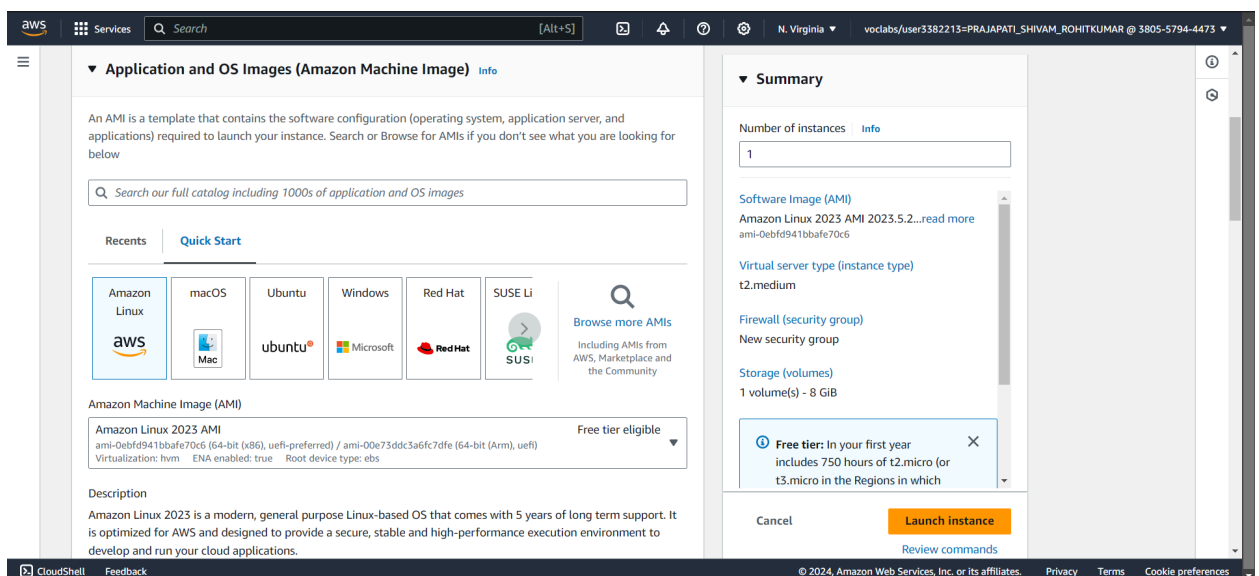
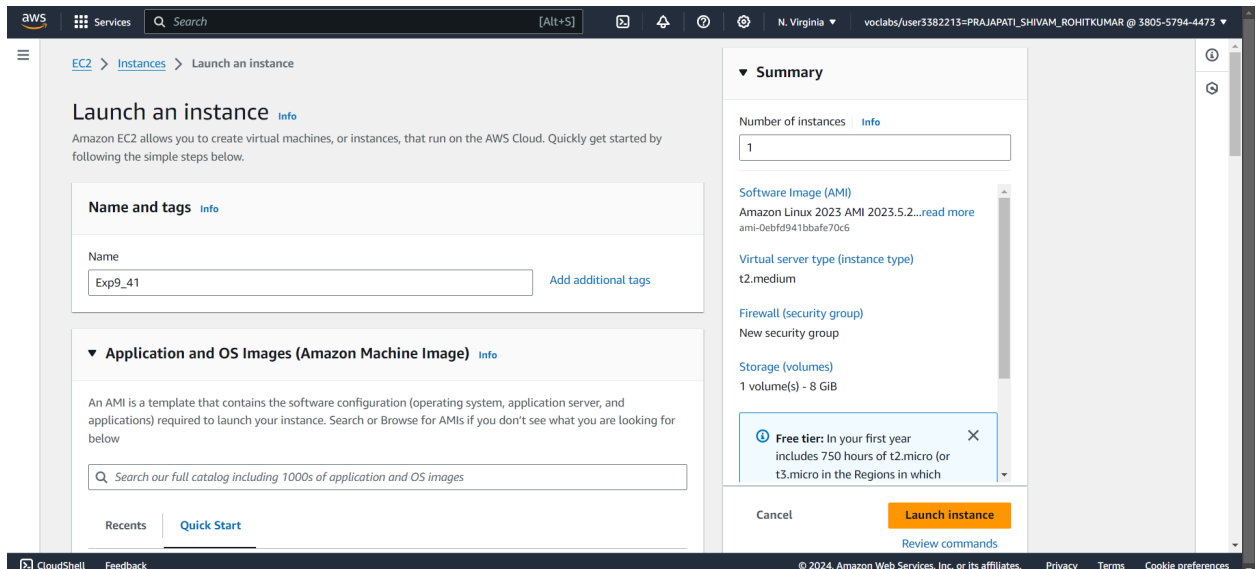


## Experiment No: 9

**AIM:** To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

**Step 1:** Sign in to your AWS account. Look for EC2 in the services list. Open it and click on "Create Instance."



Select The OS Image as Amazon Linux.

**Step 2:** If you haven't created a private key or a .pem file yet, click on "Create a key pair." Otherwise, choose the key pair you created earlier. (Be sure to remember where the .pem file for that key is located on your system.) .in my case i have created a new one

The image displays two screenshots of the AWS Management Console during the 'Launch Instance' process.

**Top Screenshot:**

- Instance type:** t2.medium. Family: t2. 2 vCPU, 4 GiB Memory. Current generation: true. On-Demand Linux base pricing: 0.0464 USD per Hour. On-Demand RHEL base pricing: 0.0752 USD per Hour. On-Demand Windows base pricing: 0.0644 USD per Hour. On-Demand SUSE base pricing: 0.1464 USD per Hour. Additional costs apply for AMIs with pre-installed software.
- Key pair (login):** You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance. Key pair name - required: Lab9\_41. [Create new key pair](#)
- Network settings:** Edit
- Summary:**
  - Number of instances: 1
  - Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2...[read more](#) (ami-0ebfd941bbafe70c6)
  - Virtual server type (instance type): t2.medium
  - Firewall (security group): New security group
  - Storage (volumes): 1 volume(s) - 8 GiB
  - Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which
- Buttons: Cancel, **Launch instance**, [Review commands](#)

**Bottom Screenshot:**

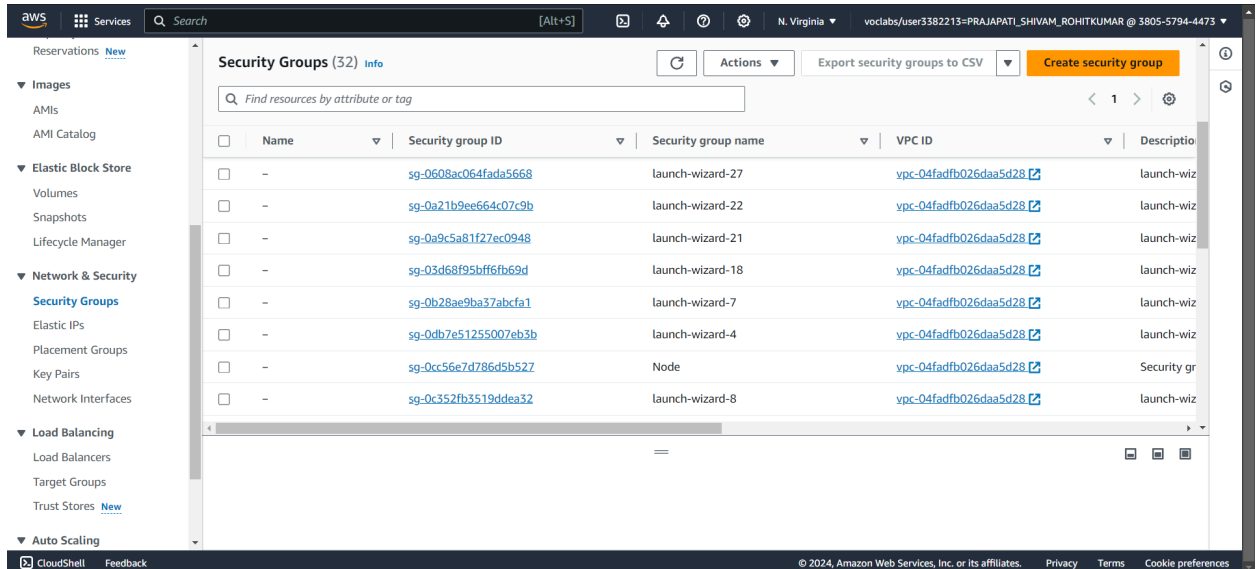
- Network settings:** Edit
- Network:** vpc-04fadb026daa5d28
- Subnet:** No preference (Default subnet in any availability zone)
- Auto-assign public IP:** Enable
- Additional charges:** Apply when outside of free tier allowance
- Firewall (security groups):** Info. A security group is a set of Firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
  - ☒ Create security group
  - ☐ Select existing security group
- We'll create a new security group called 'launch-wizard-27' with the following rules:**
  - ☒ Allow SSH traffic from: Helps you connect to your instance. Anywhere (0.0.0.0/0)
  - ☐ Allow HTTPS traffic from the internet: To set up an endpoint, for example when creating a web server
  - ☐ Allow HTTP traffic from the internet: To set up an endpoint, for example when creating a web server
- Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.**
- Summary:**
  - Number of instances: 1
  - Software Image (AMI): Amazon Linux 2023 AMI 2023.5.2...[read more](#) (ami-0ebfd941bbafe70c6)
  - Virtual server type (instance type): t2.medium
  - Firewall (security group): New security group
  - Storage (volumes): 1 volume(s) - 8 GiB
  - Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which
- Buttons: Cancel, **Launch instance**, [Review commands](#)

AWS will create a security group for this instance. Keep the name of that instance saved.

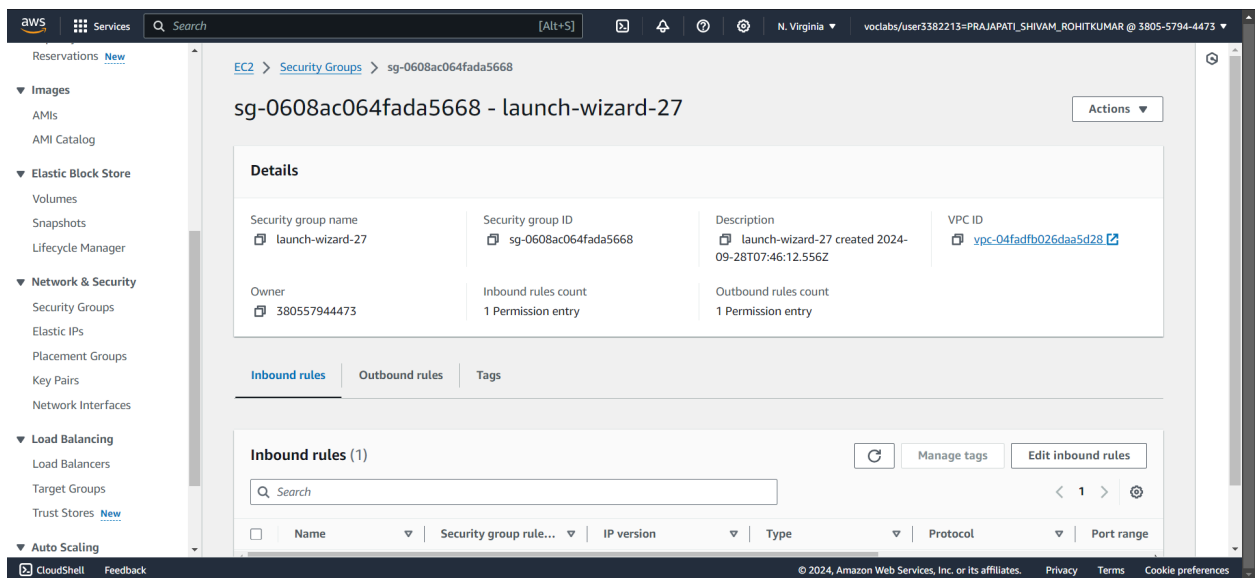
Instance:

The image shows the instance details for 'Exp9\_41' in the AWS Management Console. The instance is in the 'Running' state. The instance ID is i-03b653a01796c6519. The instance type is t2.medium.

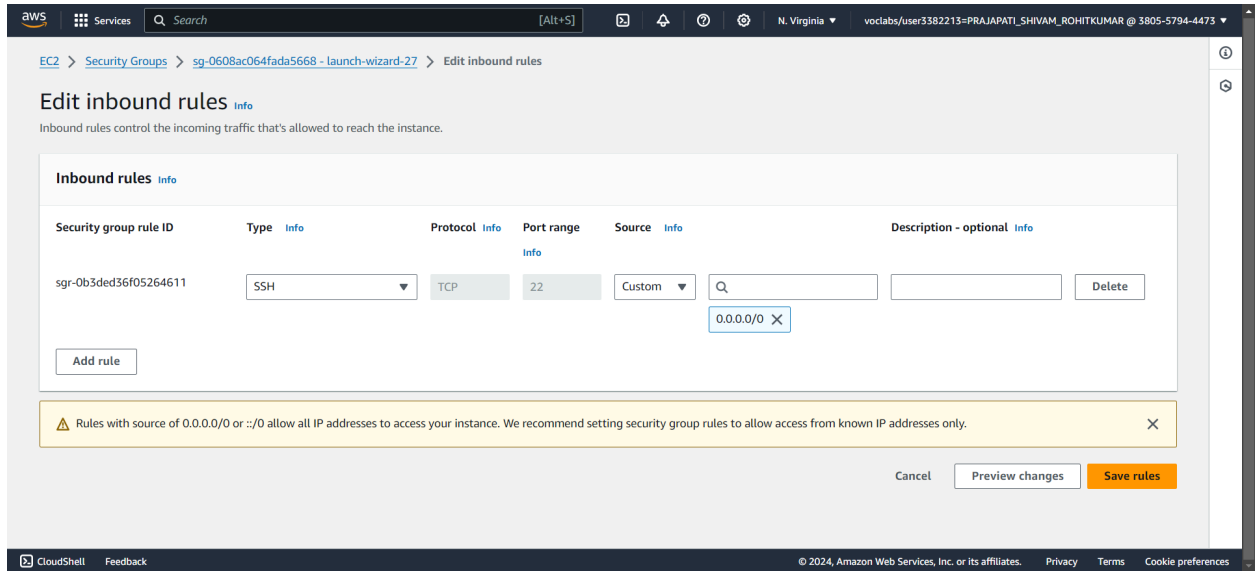
**Step3:** After you create the instance, click on **"Security Groups"** in the left sidebar. Look for the security group that corresponds to your instance, and then click on the security group ID for that group. (in my case launch-wizard-27 is the latest one.)



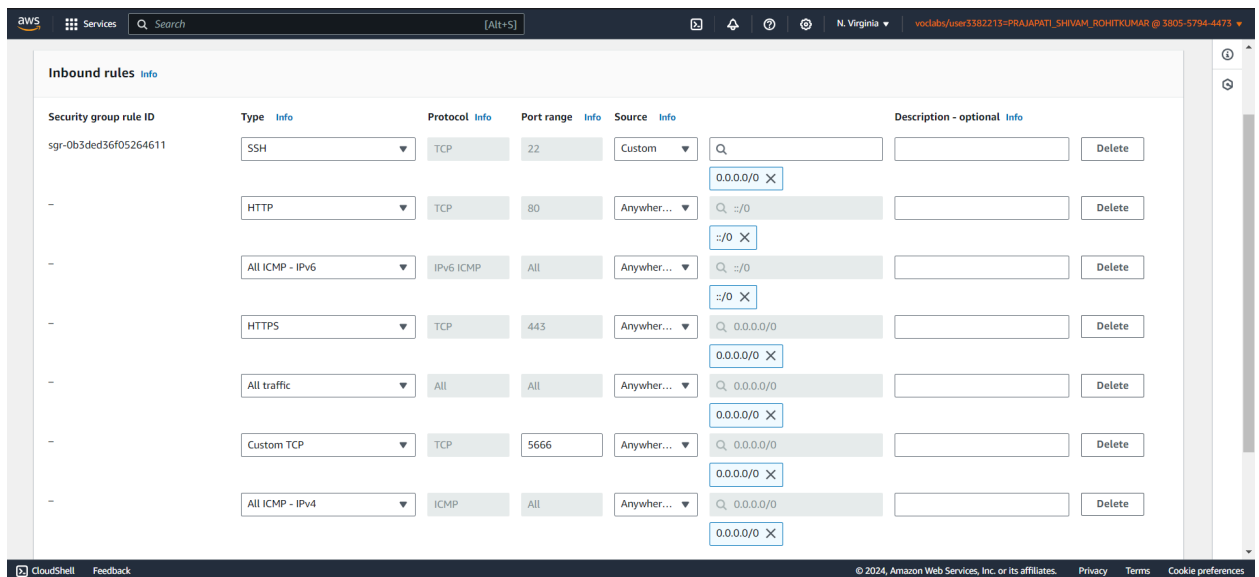
Click on Id



Click on the edit inbound rules



Next, click on "Add rules" and set up rules for the following protocols: HTTP, All ICMP (IPv6), HTTPS, All traffic, Custom TCP (Port 5666), and All ICMP (IPv4).



Click on save. This will add all the inbound rules to the security group.

The screenshot displays the AWS Management Console interface. At the top, a green notification banner states: "Inbound security group rules successfully modified on security group (sg-0608ac064fada5668 | launch-wizard-27)". Below this, the "Details" tab for the security group "sg-0608ac064fada5668 - launch-wizard-27" is active. It shows the security group name, ID, description, owner, and rule counts. The "Inbound rules" tab is selected, showing a list of 7 rules. The table below lists these rules with columns for Name, Security group rule, IP version, Type, Protocol, Port range, and Source.

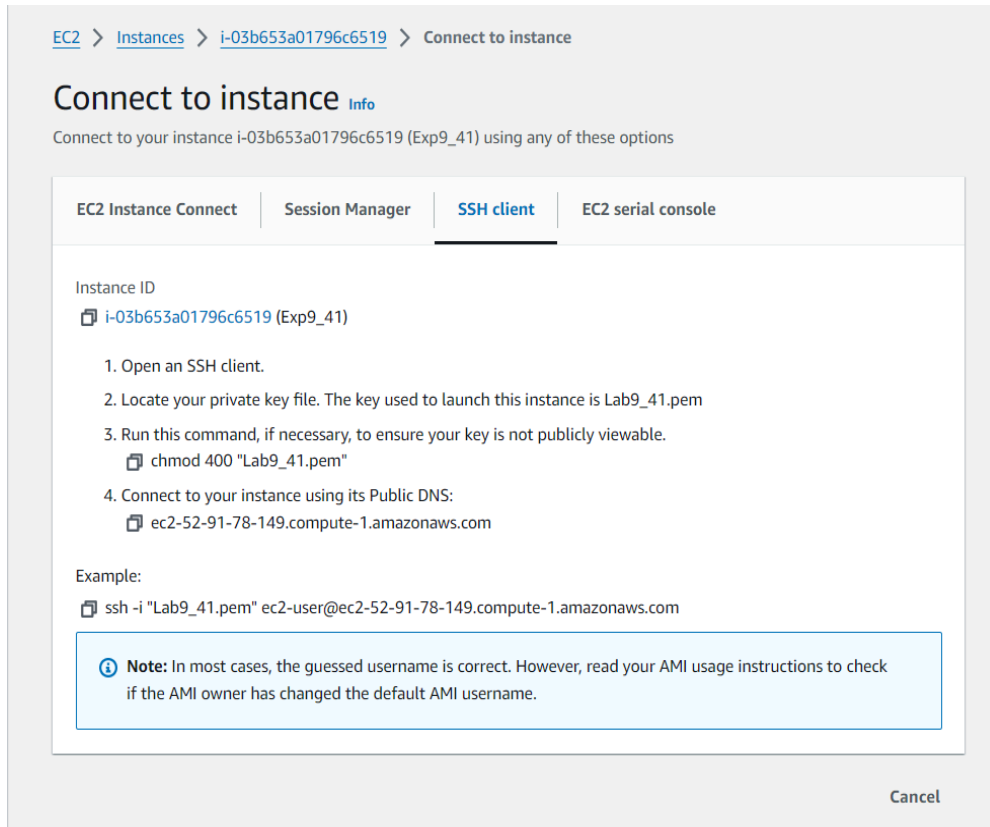
Name	Security group rule	IP version	Type	Protocol	Port range	Source
-	sgr-084a1a8d347cd98af	IPv4	Custom TCP	TCP	5666	0.0.0.0/0
-	sgr-0032db935ccd03fdd	IPv6	HTTP	TCP	80	::/0
-	sgr-00ab4addbf9eedf06	IPv6	All ICMP - IPv6	IPv6 ICMP	All	::/0
-	sgr-0b3ded36f052646...	IPv4	SSH	TCP	22	0.0.0.0/0
-	sgr-034d1384f9e7dc594	IPv4	HTTPS	TCP	443	0.0.0.0/0
-	sgr-0fc6da58919d0e445	IPv4	All ICMP - IPv4	ICMP	All	0.0.0.0/0
-	sgr-0627fb925abfb0489	IPv4	All traffic	All	All	0.0.0.0/0

**Step 4:** Return to the instances screen and click on the instance ID of your instance. Then, click on "Connect."

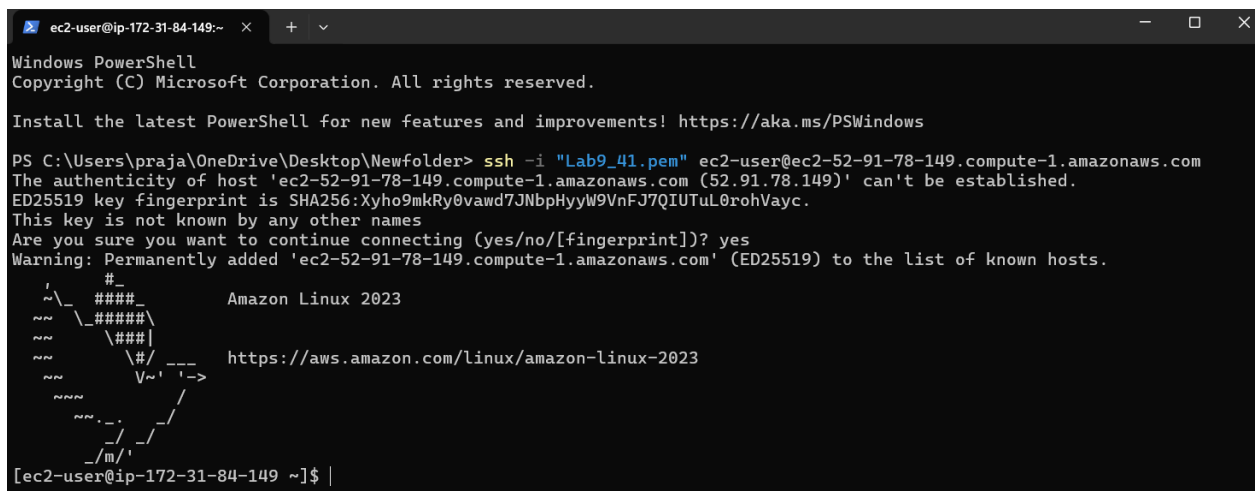
The screenshot shows the AWS Management Console "Instances" page. A table lists several EC2 instances. The instance "Exp9\_41" with ID "i-03b653a01796c6519" is highlighted in blue and has a status of "Running". The "Connect" button is visible at the top right of the instance list.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
41_WorkerNode1	i-03e9c0a0b0a13e074	Stopped	t2.medium	-	View alarms	us-east-1b	-
41_WorkerNode2	i-08432f85057a68a50	Stopped	t2.medium	-	View alarms	us-east-1b	-
41_Master	i-0d978abb7f4d24fa4	Stopped	t2.medium	-	View alarms	us-east-1b	-
Exp9_41	i-03b653a01796c6519	Running	t2.medium	2/2 checks passed	View alarms	us-east-1b	ec2-52-91-78-149.com...
Exp4_41	i-0ecfe3f98ef7f2b3e	Stopped	t2.medium	-	View alarms	us-east-1b	-
Webapp1-env	i-011c2cceff1589f0	Running	t3.micro	3/3 checks passed	View alarms	us-east-1b	ec2-52-207-16-235.co...
Webapp1-env	i-0639869278f7f2b7a	Terminated	t3.micro	-	View alarms	us-east-1b	-

Click on **"SSH client"** and copy the example command provided.



**Step 5:** Now, we need to connect our local terminal to the instance using SSH. Open the terminal where your private key file (.pem) is located by actually going to the folder which has the .pem file, paste the copied SSH command, and run it.



**Step 6:** Now we start working on this terminal. First run the command **sudo yum update**. This command will check for any updates for the YUM library to ensure that all libraries are with up to date with the latest features and security fixes

```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum update
Last metadata expiration check: 0:30:14 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-84-149 ~]$ |
```

**Step 7:** We are going to set up a web server software application called **Apache** and a programming language called **PHP** in this instance. To do this, run this command in your terminal

**sudo yum install httpd php**

```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum install httpd php
Last metadata expiration check: 0:31:34 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
=====
Package                                Architecture      Version           Repository        Size
=====
Installing:
httpd                                   x86_64            2.4.62-1.amzn2023 amazonlinux        48 k
php8.3                                 x86_64            8.3.10-1.amzn2023 amazonlinux        10 k
Installing dependencies:
apr                                     x86_64            1.7.2-2.amzn2023.0.2 amazonlinux        129 k
apr-util                               x86_64            1.6.3-1.amzn2023.0.1 amazonlinux        98 k
generic-logos-httpd                   noarch            18.0.0-12.amzn2023.0.3 amazonlinux        19 k
httpd-core                             x86_64            2.4.62-1.amzn2023 amazonlinux        1.4 M
httpd-filesystem                       noarch            2.4.62-1.amzn2023 amazonlinux        14 k
httpd-tools                            x86_64            2.4.62-1.amzn2023 amazonlinux        81 k
libbrotli                              x86_64            1.0.9-4.amzn2023.0.2 amazonlinux        315 k
libsodium                              x86_64            1.0.19-4.amzn2023 amazonlinux        176 k
libxslt                                x86_64            1.1.34-5.amzn2023.0.2 amazonlinux        241 k
mailcap                                noarch            2.1.49-3.amzn2023.0.3 amazonlinux        33 k
nginx-filesystem                       noarch            1:1.24.0-1.amzn2023.0.4 amazonlinux        9.8 k
php8.3-cli                             x86_64            8.3.10-1.amzn2023.0.1 amazonlinux        3.7 M
php8.3-common                          x86_64            8.3.10-1.amzn2023.0.1 amazonlinux        737 k
php8.3-process                         x86_64            8.3.10-1.amzn2023.0.1 amazonlinux        45 k
php8.3-xml                             x86_64            8.3.10-1.amzn2023.0.1 amazonlinux        154 k
Installing weak dependencies:
apr-util-openssl                      x86_64            1.6.3-1.amzn2023.0.1 amazonlinux        17 k
mod_http2                             x86_64            2.0.27-1.amzn2023.0.3 amazonlinux        166 k
mod_lua                               x86_64            2.4.62-1.amzn2023 amazonlinux        61 k

Installed:
apr-1.7.2-2.amzn2023.0.2.x86_64          apr-util-1.6.3-1.amzn2023.0.1.x86_64
apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
httpd-2.4.62-1.amzn2023.x86_64          generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
httpd-2.4.62-1.amzn2023.x86_64          httpd-core-2.4.62-1.amzn2023.x86_64
httpd-filesystem-2.4.62-1.amzn2023.noarch
httpd-tools-2.4.62-1.amzn2023.x86_64
libbrotli-1.0.9-4.amzn2023.0.2.x86_64
libxslt-1.1.34-5.amzn2023.0.2.x86_64
libsodium-1.0.19-4.amzn2023.x86_64
mailcap-2.1.49-3.amzn2023.0.3.noarch
mod_http2-2.0.27-1.amzn2023.0.3.x86_64
mod_lua-2.4.62-1.amzn2023.x86_64
nginx-filesystem-1:1.24.0-1.amzn2023.0.4.noarch
php8.3-cli-8.3.10-1.amzn2023.0.1.x86_64
php8.3-fpm-8.3.10-1.amzn2023.0.1.x86_64
php8.3-opcache-8.3.10-1.amzn2023.0.1.x86_64
php8.3-process-8.3.10-1.amzn2023.0.1.x86_64
php8.3-xml-8.3.10-1.amzn2023.0.1.x86_64
php8.3-common-8.3.10-1.amzn2023.0.1.x86_64
php8.3-mbstring-8.3.10-1.amzn2023.0.1.x86_64
php8.3-pdo-8.3.10-1.amzn2023.0.1.x86_64
php8.3-sodium-8.3.10-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-84-149 ~]$ |
```

**Step 8:** Now, we will install the GCC compiler, which is used for compiling and running C and C++ programs, along with the essential C libraries. To do this, enter the following command: **sudo yum install gcc glibc glibc-common**

```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum install gcc glibc glibc-common
Last metadata expiration check: 0:34:20 ago on Sat Sep 28 07:51:31 2024.
Package glibc-2.34-52.amzn2023.0.11.x86_64 is already installed.
Package glibc-common-2.34-52.amzn2023.0.11.x86_64 is already installed.
Dependencies resolved.
=====
Package                        Architecture      Version           Repository        Size
=====
Installing:
gcc                            x86_64            11.4.1-2.amzn2023.0.2    amazonlinux        32 M
Installing dependencies:
annobin-docs                   noarch            10.93-1.amzn2023.0.1    amazonlinux         92 k
annobin-plugin-gcc             x86_64            10.93-1.amzn2023.0.1    amazonlinux        887 k
cpp                             x86_64            11.4.1-2.amzn2023.0.2    amazonlinux         10 M
gc                              x86_64            8.0.4-5.amzn2023.0.2    amazonlinux        105 k
glibc-devel                    x86_64            2.34-52.amzn2023.0.11    amazonlinux         27 k
glibc-headers-x86             noarch            2.34-52.amzn2023.0.11    amazonlinux        427 k
guile22                        x86_64            2.2.7-2.amzn2023.0.3    amazonlinux         6.4 M
kernel-headers                 x86_64            6.1.109-118.189.amzn2023    amazonlinux        1.4 M
libmpc                         x86_64            1.2.1-2.amzn2023.0.2    amazonlinux         62 k
libtool-ltdl                   x86_64            2.4.7-1.amzn2023.0.3    amazonlinux         38 k
libxcrypt-devel                x86_64            4.4.33-7.amzn2023        amazonlinux         32 k
make                           x86_64            1:4.3-5.amzn2023.0.2    amazonlinux        534 k

Transaction Summary
=====
Install 13 Packages

Total download size: 52 M
```

```
Installed:
annobin-docs-10.93-1.amzn2023.0.1.noarch
cpp-11.4.1-2.amzn2023.0.2.x86_64
gcc-11.4.1-2.amzn2023.0.2.x86_64
glibc-headers-x86-2.34-52.amzn2023.0.11.noarch
kernel-headers-6.1.109-118.189.amzn2023.x86_64
libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64
make-1:4.3-5.amzn2023.0.2.x86_64
annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64
gc-8.0.4-5.amzn2023.0.2.x86_64
glibc-devel-2.34-52.amzn2023.0.11.x86_64
guile22-2.2.7-2.amzn2023.0.3.x86_64
libmpc-1.2.1-2.amzn2023.0.2.x86_64
libxcrypt-devel-4.4.33-7.amzn2023.x86_64

Complete!
[ec2-user@ip-172-31-84-149 ~]$ |
```

**Step 9:** Next, we need to install the GD library, along with its development tools. This library helps with creating and manipulating images. For that, run this command **sudo yum install gd gd-devel**

```
[ec2-user@ip-172-31-84-149 ~]$ sudo yum install gd gd-devel
Last metadata expiration check: 0:36:28 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
=====
Package                        Architecture      Version           Repository        Size
=====
Installing:
gd                             x86_64            2.3.3-5.amzn2023.0.3    amazonlinux        139 k
gd-devel                       x86_64            2.3.3-5.amzn2023.0.3    amazonlinux         38 k
Installing dependencies:
brotli                         x86_64            1.0.9-4.amzn2023.0.2    amazonlinux        314 k
brotli-devel                   x86_64            1.0.9-4.amzn2023.0.2    amazonlinux         31 k
bzip2-devel                     x86_64            1.0.8-6.amzn2023.0.2    amazonlinux        214 k
cairo                           x86_64            1.17.6-2.amzn2023.0.1    amazonlinux        684 k
cmake-filesystem                x86_64            3.22.2-1.amzn2023.0.4    amazonlinux         16 k
fontconfig                      x86_64            2.13.94-2.amzn2023.0.2    amazonlinux        273 k
fontconfig-devel                x86_64            2.13.94-2.amzn2023.0.2    amazonlinux        128 k
fonts-filessystem                noarch            1:2.0.5-12.amzn2023.0.2    amazonlinux         9.5 k
freetype                       x86_64            2.13.2-5.amzn2023.0.1    amazonlinux        423 k
freetype-devel                  x86_64            2.13.2-5.amzn2023.0.1    amazonlinux        912 k
```



```
libjpeg-turbo-devel-2.1.4-2.amzn2023.0.5.x86_64      libmount-devel-2.37.4-1.amzn2023.0.4.x86_64
libpng-2:1.6.37-10.amzn2023.0.6.x86_64             libpng-devel-2:1.6.37-10.amzn2023.0.6.x86_64
libselinux-devel-3.4-5.amzn2023.0.2.x86_64         libsepol-devel-3.4-3.amzn2023.0.3.x86_64
libtiff-4.4.0-4.amzn2023.0.18.x86_64               libtiff-devel-4.4.0-4.amzn2023.0.18.x86_64
libwebp-1.2.4-1.amzn2023.0.6.x86_64                libwebp-devel-1.2.4-1.amzn2023.0.6.x86_64
libxcb-1.13.1-7.amzn2023.0.2.x86_64                libxcb-devel-1.13.1-7.amzn2023.0.2.x86_64
libxml2-devel-2.10.4-1.amzn2023.0.6.x86_64          pcre2-devel-10.40-1.amzn2023.0.3.x86_64
pcre2-utf16-10.40-1.amzn2023.0.3.x86_64            pcre2-utf32-10.40-1.amzn2023.0.3.x86_64
pixman-0.40.0-3.amzn2023.0.3.x86_64                sysprof-capture-devel-3.40.1-2.amzn2023.0.2.x86_64
xml-common-0.6.3-56.amzn2023.0.2.noarch            xorg-x11-proto-devel-2021.4-1.amzn2023.0.2.noarch
xz-devel-5.2.5-9.amzn2023.0.2.x86_64               zlib-devel-1.2.11-33.amzn2023.0.5.x86_64

Complete!
[ec2-user@ip-172-31-84-149 ~]$ |
```

**Step 10:** Now, we create a user called ‘**nagios**’ and make sure that it has a home directory, and set up a password for it.

**sudo adduser -m nagios**

**sudo passwd nagios**

```
[ec2-user@ip-172-31-84-149 ~]$ sudo adduser -m nagios
sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-84-149 ~]$ |
```

Password :Shivam2@

**Step 11:** Now, we need to create a user group named **nagcmd**, which will be used to execute Nagios commands. To do this, run the following command: **sudo groupadd nagcmd**

```
[ec2-user@ip-172-31-84-149 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-84-149 ~]$ |
```

**Step 12:** Next, we'll add the users **apache** and **nagios** to the **nagcmd** group. This allows them to execute Nagios commands.

**sudo usermod -a -G nagcmd nagios**

**sudo usermod -a -G nagcmd apache**

```
[ec2-user@ip-172-31-84-149 ~]$ sudo usermod -a -G nagcmd nagios
sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-84-149 ~]$ |
```

**Step 13:** We'll create a directory called **downloads** to store the files related to the Nagios server that we download.

**mkdir ~/downloads**

**cd ~/downloads**

```
[ec2-user@ip-172-31-84-149 ~]$ mkdir ~/downloads
cd ~/downloads
[ec2-user@ip-172-31-84-149 downloads]$ |
```

**Step 14:** Now we need to install the latest versions of nagios-core and nagios-plugins. Go to the respective websites and check whether a better version is available. If newer versions are available, then right click on the download button → Copy link address. Paste this link address in place of the current link in command. If not run these commands.

**wget <https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz>**

```
[ec2-user@ip-172-31-84-149 downloads]$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
--2024-09-28 08:37:22-- https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.5.5.tar.gz
Resolving assets.nagios.com (assets.nagios.com)... 45.79.49.120, 2600:3c00::f03c:92ff:fef7:45ce
Connecting to assets.nagios.com (assets.nagios.com)|45.79.49.120|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2065473 (2.0M) [application/x-gzip]
Saving to: 'nagios-4.5.5.tar.gz'

nagios-4.5.5.tar.gz      100%[=====] 1.97M  5.07MB/s  in 0.4s
2024-09-28 08:37:23 (5.07 MB/s) - 'nagios-4.5.5.tar.gz' saved [2065473/2065473]
[ec2-user@ip-172-31-84-149 downloads]$ |
```

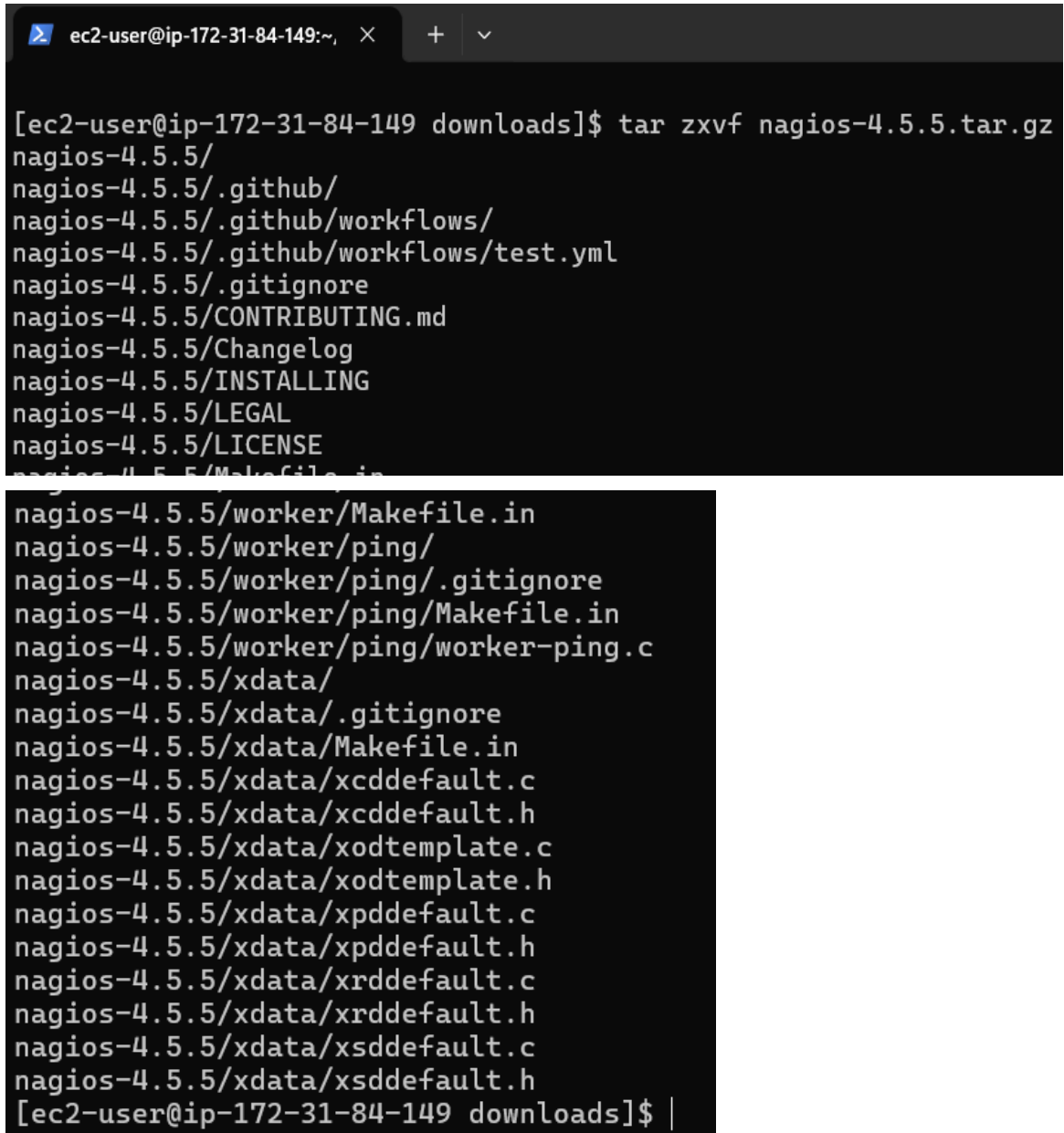
**wget <https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz>**

```
[ec2-user@ip-172-31-84-149 downloads]$ wget https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
--2024-09-28 08:38:31-- https://nagios-plugins.org/download/nagios-plugins-2.4.11.tar.gz
Resolving nagios-plugins.org (nagios-plugins.org)... 45.56.123.251
Connecting to nagios-plugins.org (nagios-plugins.org)|45.56.123.251|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2753049 (2.6M) [application/x-gzip]
Saving to: 'nagios-plugins-2.4.11.tar.gz'

nagios-plugins-2.4.11.tar.gz 100%[=====] 2.62M  7.16MB/s  in 0.4s
2024-09-28 08:38:32 (7.16 MB/s) - 'nagios-plugins-2.4.11.tar.gz' saved [2753049/2753049]
[ec2-user@ip-172-31-84-149 downloads]$ |
```

**Step 15:** Now, we need to extract the Nagios Core file into the same directory. We can do this using the tar command.

**tar zxvf nagios-4.5.5.tar.gz**

A terminal window with a dark background. The title bar shows 'ec2-user@ip-172-31-84-149:~' and window controls. The terminal output shows the command 'tar zxvf nagios-4.5.5.tar.gz' being executed, followed by a list of files and directories extracted from the archive. The files listed include .github/, .github/workflows/, .github/workflows/test.yml, .gitignore, CONTRIBUTING.md, Changelog, INSTALLING, LEGAL, LICENSE, Makefile.in, worker/, worker/ping/, worker/ping/.gitignore, worker/ping/Makefile.in, worker/ping/worker-ping.c, xdata/, xdata/.gitignore, xdata/Makefile.in, xdata/xcddefault.c, xdata/xcddefault.h, xdata/xodtemplate.c, xdata/xodtemplate.h, xdata/xpddefault.c, xdata/xpddefault.h, xdata/xrddefault.c, xdata/xrddefault.h, xdata/xsddefault.c, and xdata/xsddefault.h. The prompt returns to the user's shell.

```
[ec2-user@ip-172-31-84-149 downloads]$ tar zxvf nagios-4.5.5.tar.gz
nagios-4.5.5/
nagios-4.5.5/.github/
nagios-4.5.5/.github/workflows/
nagios-4.5.5/.github/workflows/test.yml
nagios-4.5.5/.gitignore
nagios-4.5.5/CONTRIBUTING.md
nagios-4.5.5/Changelog
nagios-4.5.5/INSTALLING
nagios-4.5.5/LEGAL
nagios-4.5.5/LICENSE
nagios-4.5.5/Makefile.in
nagios-4.5.5/worker/Makefile.in
nagios-4.5.5/worker/ping/
nagios-4.5.5/worker/ping/.gitignore
nagios-4.5.5/worker/ping/Makefile.in
nagios-4.5.5/worker/ping/worker-ping.c
nagios-4.5.5/xdata/
nagios-4.5.5/xdata/.gitignore
nagios-4.5.5/xdata/Makefile.in
nagios-4.5.5/xdata/xcddefault.c
nagios-4.5.5/xdata/xcddefault.h
nagios-4.5.5/xdata/xodtemplate.c
nagios-4.5.5/xdata/xodtemplate.h
nagios-4.5.5/xdata/xpddefault.c
nagios-4.5.5/xdata/xpddefault.h
nagios-4.5.5/xdata/xrddefault.c
nagios-4.5.5/xdata/xrddefault.h
nagios-4.5.5/xdata/xsddefault.c
nagios-4.5.5/xdata/xsddefault.h
[ec2-user@ip-172-31-84-149 downloads]$ |
```

**Step16:** Now, we need to ensure that Nagios uses the nagcmd group for executing external commands.

**./configure --with-command-group=nagcmd**

```
nagios-4.5.5/xdata/xsddata.c
[ec2-user@ip-172-31-84-149 downloads]$ ./configure --with-command-group=nagcmd
-bash: ./configure: No such file or directory
[ec2-user@ip-172-31-84-149 downloads]$ |
```

An error was encountered here: `./configure: no such path or directory` . So Navigate to the nagios-4.5.5 folder in downloads. (version could vary)

ls :

```
[ec2-user@ip-172-31-84-149 downloads]$ ls
nagios-4.5.5  nagios-4.5.5.tar.gz  nagios-plugins-2.4.11.tar.gz
[ec2-user@ip-172-31-84-149 downloads]$ |
```

- cd nagios-4.5.5 (use the version shown by your ls command)
- ./configure --with-command-group=nagcmd

Another error could be Cannot find SSL headers. To solve this, we need to install OpenSSL Dev Library : **sudo yum install openssl-devel**

```
[ec2-user@ip-172-31-83-157 nagios-4.5.5]$ sudo yum install openssl-devel
Last metadata expiration check: 0:21:59 ago on Sat Sep 28 03:46:46 2024.
Dependencies resolved.
=====
Package                                Repository      Architecture    Size      Version
=====
Installing:
openssl-devel                          amazonlinux     x86_64          3.0 M     1:3.0.8-1.amzn2023.0.14
Transaction Summary
=====
Install 1 Package
Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
openssl-devel-3.0.8-1.amzn2023.0.14.x86_64.rpm
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
```

```
Last metadata expiration check: 0:58:10 ago on Sat Sep 28 07:51:31 2024.
Dependencies resolved.
=====
Package                                Architecture    Version          Repository      Size
=====
Installing:
openssl-devel                          x86_64          1:3.0.8-1.amzn2023.0.14  amazonlinux     3.0 M
Transaction Summary
=====
Install 1 Package
Total download size: 3.0 M
Installed size: 4.7 M
Is this ok [y/N]: y
Downloading Packages:
openssl-devel-3.0.8-1.amzn2023.0.14.x86_64.rpm
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
```

**./configure --with-command-group=nagcmd**

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ ./configure --with-command-group=nagcmd
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 the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for strings.h... yes
checking for sys/stat.h... yes
checking for sys/types.h... yes
```

```
*** Configuration summary for nagios 4.5.5 2024-09-17 ***:

General Options:
-----
Nagios executable: nagios
Nagios user/group: nagios,nagios
Command user/group: nagios,nagcmd
Event Broker: yes
Install ${prefix}: /usr/local/nagios
Install ${includedir}: /usr/local/nagios/include/nagios
Lock file: /run/nagios.lock
Check result directory: /usr/local/nagios/var/spool/checkresults
Init directory: /lib/systemd/system
Apache conf.d directory: /etc/httpd/conf.d
Mail program: /bin/mail
Host OS: linux-gnu
IOBroker Method: epoll

Web Interface Options:
-----
HTML URL: http://localhost/nagios/
CGI URL: http://localhost/nagios/cgi-bin/
Traceroute (used by WAP): /usr/bin/traceroute

Review the options above for accuracy. If they look okay,
type 'make all' to compile the main program and CGIs.

[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ |
```

**Step 17:** Next, we need to compile all the components of the software based on the instructions in the Makefile. For that, use this command: **make all** Then, **sudo make install**

sudo make install-init

sudo make install-config

sudo make install-commandmode

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ make all
cd ./base && make
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o ./nagios.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o broker.o broker.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nebmods.o nebmods.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ./common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o query-handler.o query-handler.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
In function 'get_wproc_list',
    inlined from 'get_worker' at workers.c:277:12:
workers.c:253:17: warning: '%s' directive argument is null [-Wformat-overflow=]
   253 |         log_debug_info(DEBUGL_CHECKS, 1, "Found specialized worker(s) for '%s'", (slash && *slash != '/')
       |         ^
       |         |
       |         ^~~~~~
^~~~~~
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o checks.o checks.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o config.o config.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o commands.o commands.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o events.o events.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o flapping.o flapping.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o logging.o logging.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o macros-base.o ../common/macros.c
gcc -Wall -I.. -I. -I../lib -I../include -I../include -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o netutils.o netutils.
```

### \*\*\* Support Notes \*\*\*\*\*

If you have questions about configuring or running Nagios, please make sure that you:

- Look at the sample config files
- Read the documentation on the Nagios Library at:  
<https://library.nagios.com>

before you post a question to one of the mailing lists. Also make sure to include pertinent information that could help others help you. This might include:

- What version of Nagios you are using
- What version of the plugins you are using
- Relevant snippets from your config files
- Relevant error messages from the Nagios log file

For more information on obtaining support for Nagios, visit:

<https://support.nagios.com>

\*\*\*\*\*

Enjoy.

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ |
```

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
cd ./base && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiosstats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make install-basic
make[2]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
    /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-4.5.5/cgi'
cd ./html && make install
make[1]: Entering directory '/home/ec2-user/downloads/nagios-4.5.5/html'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/media
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/stylesheets
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/contexthelp
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/share/docs/images
```

\*\*\* 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.

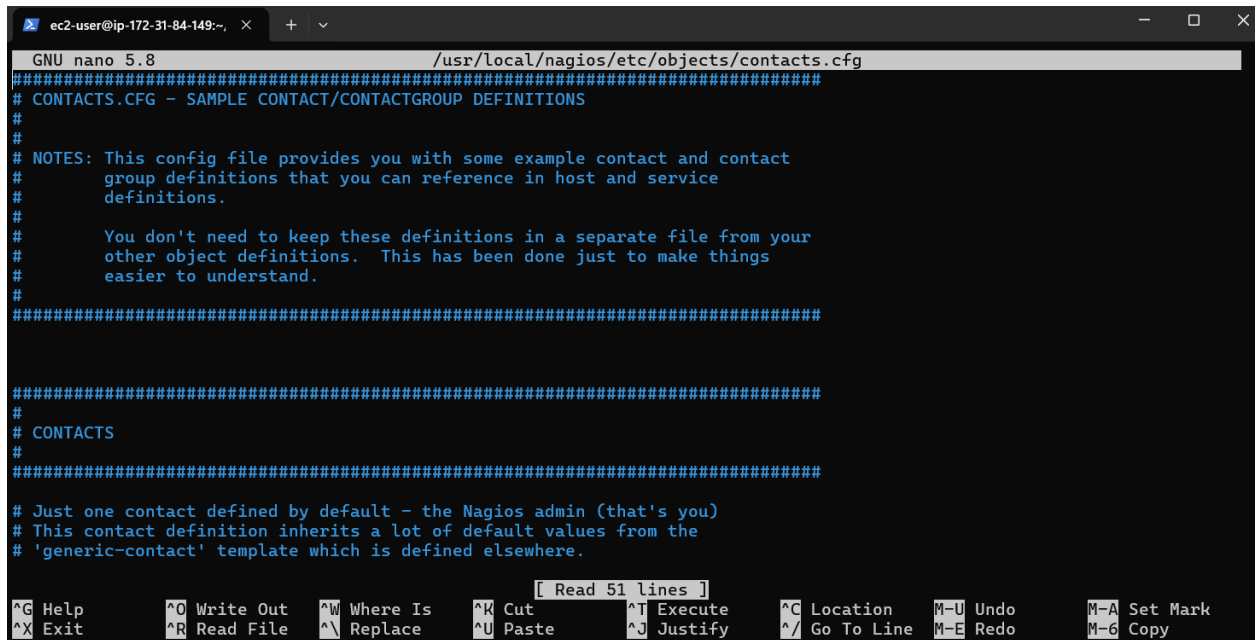
```
/usr/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw
```

\*\*\* External command directory configured \*\*\*

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ |
```

**Step 18:** We need to update the email linked with this server to our email for it to send notifications (if any needed). **sudo nano /usr/local/nagios/etc/objects/contacts.cfg**

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg|
```



```
GNU nano 5.8 /usr/local/nagios/etc/objects/contacts.cfg
#####
# CONTACTS.CFG - SAMPLE CONTACT/CONTACTGROUP DEFINITIONS
#
#
# NOTES: This config file provides you with some example contact and contact
#        group definitions that you can reference in host and service
#        definitions.
#
#        You don't need to keep these definitions in a separate file from your
#        other object definitions. This has been done just to make things
#        easier to understand.
#
#####

# CONTACTS
#
#####

# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the
# 'generic-contact' template which is defined elsewhere.

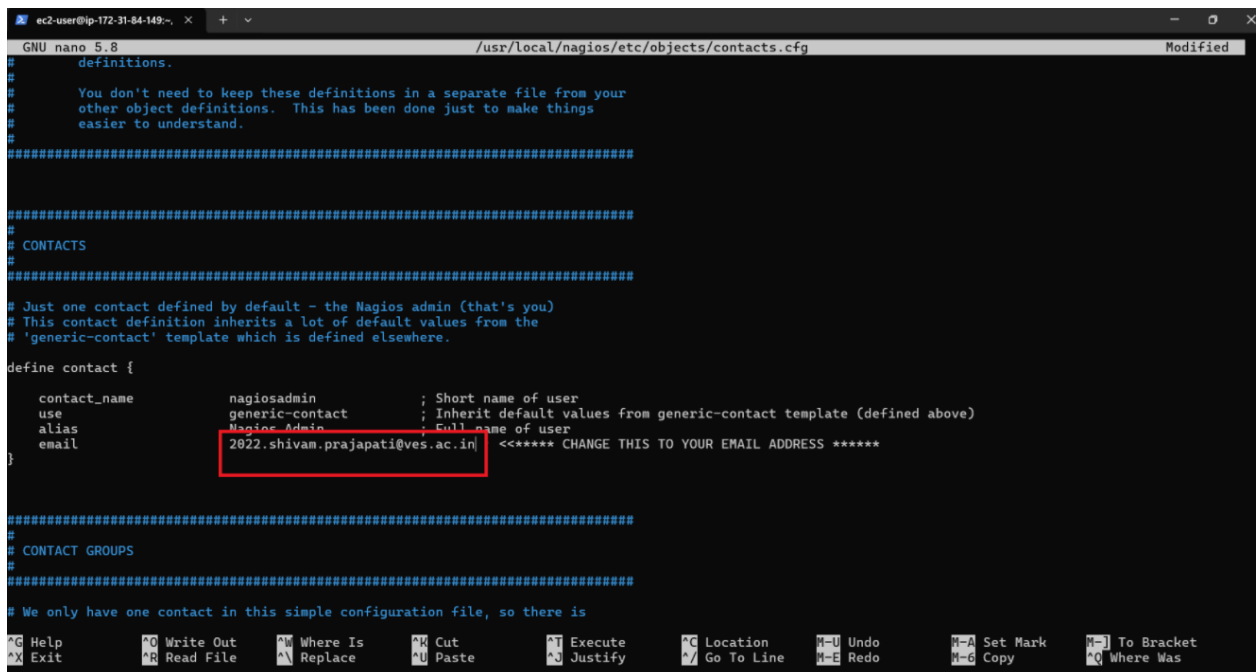
define contact {
    contact_name    nagiosadmin          ; Short name of user
    use             generic-contact      ; Inherit default values from generic-contact template (defined above)
    alias           Nagios Admin        ; Full name of user
    email           2022.shivam.prajapati@ves.ac.in  <***** CHANGE THIS TO YOUR EMAIL ADDRESS *****>
}

#####

# CONTACT GROUPS
#
#####

# We only have one contact in this simple configuration file, so there is
```

Here, change the email under 'define contact{' to your email address



```
GNU nano 5.8 /usr/local/nagios/etc/objects/contacts.cfg Modified
#####
# CONTACTS.CFG - SAMPLE CONTACT/CONTACTGROUP DEFINITIONS
#
#
# NOTES: This config file provides you with some example contact and contact
#        group definitions that you can reference in host and service
#        definitions.
#
#        You don't need to keep these definitions in a separate file from your
#        other object definitions. This has been done just to make things
#        easier to understand.
#
#####

# CONTACTS
#
#####

# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the
# 'generic-contact' template which is defined elsewhere.

define contact {
    contact_name    nagiosadmin          ; Short name of user
    use             generic-contact      ; Inherit default values from generic-contact template (defined above)
    alias           Nagios Admin        ; Full name of user
    email           2022.shivam.prajapati@ves.ac.in  <***** CHANGE THIS TO YOUR EMAIL ADDRESS *****>
}

#####

# CONTACT GROUPS
#
#####

# We only have one contact in this simple configuration file, so there is
```

To save this use the following shortcut sequence CTRL+O→Enter→CTRL+X.

CTRL+O: Overwrite the existing file with edited file

CTRL+X: Exit nano editor



**Step 19:** We need to install the necessary configuration files for the Nagios web interface.  
**sudo make install-webconf**

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ [ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf
if [ 0 -eq 1 ]; then \
    ln -s /etc/httpd/conf.d/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

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

[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ |
```

**Step 20:** Now we need to create a user to access the Nagios web interface. For that, run this command to create a user named '**nagiosadmin**'. Keep this username and password saved as it is needed to login to the web interface. **sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin**

```
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ |
```

Kingmaker is the password

**Step 21:** Now, we need to restart the Apache server to apply all the recent configurations. Use this command: **sudo service httpd restart**

```
Adding password for user nagiosadmin
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ |
```

**Step 22:** Now we go back to the downloads folder and extract the files of nagios plugin.  
**cd ~/downloads**  
**tar xzvf nagios-plugins-2.4.11.tar.gz (Version may vary)**

```
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-84-149 nagios-4.5.5]$ cd ~/downloads
[ec2-user@ip-172-31-84-149 downloads]$ |
```

```
[ec2-user@ip-172-31-84-149 downloads]$ tar zxvf nagios-plugins-2.4.11.tar.gz
nagios-plugins-2.4.11/
nagios-plugins-2.4.11/build-aux/
nagios-plugins-2.4.11/build-aux/compile
nagios-plugins-2.4.11/build-aux/config.guess
nagios-plugins-2.4.11/build-aux/config.rpath
nagios-plugins-2.4.11/build-aux/config.sub
nagios-plugins-2.4.11/build-aux/install-sh
nagios-plugins-2.4.11/build-aux/ltmain.sh
nagios-plugins-2.4.11/build-aux/missing
nagios-plugins-2.4.11/build-aux/mkinstalldirs
nagios-plugins-2.4.11/build-aux/depcomp
nagios-plugins-2.4.11/build-aux/snippet/
nagios-plugins-2.4.11/build-aux/snippet/_Noreturn.h
nagios-plugins-2.4.11/build-aux/snippet/arg-nonnull.h
nagios-plugins-2.4.11/build-aux/snippet/c++defs.h
nagios-plugins-2.4.11/build-aux/snippet/warn-on-use.h
nagios-plugins-2.4.11/build-aux/test-driver
```

**Step 23:** Again, we need to install the configurations for these files.

**cd nagios-plugins-2.4.11** (version may vary)

```
[ec2-user@ip-172-31-84-149 downloads]$ cd nagios-plugins-2.4.11
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ |
```

**./configure --with-nagios-user=nagios --with-nagios-group=nagios**

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
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 gcc understands -c and -o together... yes
checking whether make supports the include directive... yes (GNU style)
checking dependency style of gcc... gcc3
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
```

**Step 24:** We need to compile all the components of this software based on the instructions in the Makefile.

make

sudo make install

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ make
make all-recursive
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
Making all in gl
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make all-recursive
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Nothing to be done for 'all-am'.
```

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ sudo make install
Making install in gl
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make install-recursive
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
make[4]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/gl'
if test yes = no; then \
  case 'linux-gnu' in \
    darwin[56]*) \
      need_charset_alias=true ;; \
    darwin* | cygwin* | mingw* | pw32* | cegcc*) \
      need_charset_alias=false ;; \
    *) \
      need_charset_alias=true ;; \
  esac ; \
```

```
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11/po'
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.4.11'
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ |
```

**Step 25:** We need to register the Nagios service with the system to enable it to manage the server status

sudo chkconfig --add nagios

sudo chkconfig nagios on

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ sudo chkconfig --add nagios
sudo chkconfig nagios on
error reading information on service nagios: No such file or directory
Note: Forwarding request to 'systemctl enable nagios.service'.
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /usr/lib/systemd/system/nagios.service.
```

**Step 26:** We need to verify the Nagios configuration for any syntax errors or issues before starting or restarting the Nagios service.

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

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ sudo /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 8 services.
  Checked 1 hosts.
  Checked 1 host groups.
  Checked 0 service groups.
  Checked 1 contacts.
  Checked 1 contact groups.
  Checked 24 commands.
  Checked 5 time periods.
  Checked 0 host escalations.
  Checked 0 service escalations.
Checking for circular paths...
```

**sudo service nagios start**

```
[ec2-user@ip-172-31-84-149 nagios-plugins-2.4.11]$ cd
[ec2-user@ip-172-31-84-149 ~]$ sudo service nagios start
Redirecting to /bin/systemctl start nagios.service
[ec2-user@ip-172-31-84-149 ~]$ |
```

**Step 27:** Check the status of the nagios.

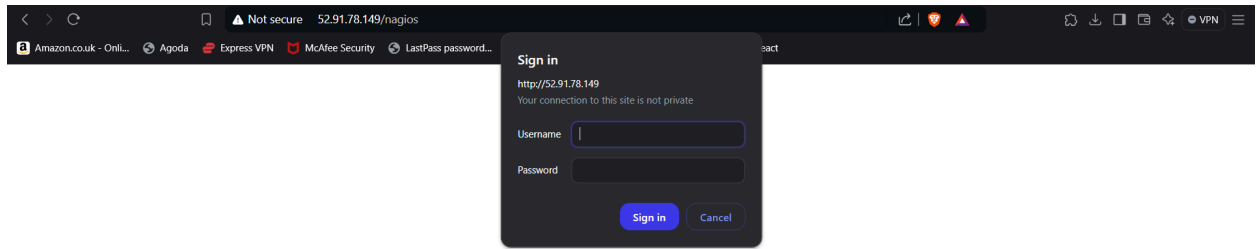
```
ec2-user@ip-172-31-84-149:~$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.5.5
   Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
   Active: active (running) since Sat 2024-09-28 09:51:48 UTC; 43s ago
     Docs: https://www.nagios.org/documentation
   Process: 67663 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SU
   Process: 67664 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SU
   Main PID: 67665 (nagios)
    Tasks: 6 (limit: 4658)
   Memory: 5.8M
      CPU: 87ms
   CGroup: /system.slice/nagios.service
           └─67665 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             └─67666 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
               └─67667 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
                 └─67668 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
                   └─67669 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
                     └─67670 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg

Sep 28 09:51:48 ip-172-31-84-149.ec2.internal nagios[67665]: qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successful
Sep 28 09:51:48 ip-172-31-84-149.ec2.internal nagios[67665]: qh: core query handler registered
Sep 28 09:51:48 ip-172-31-84-149.ec2.internal nagios[67665]: qh: echo service query handler registered
Sep 28 09:51:48 ip-172-31-84-149.ec2.internal nagios[67665]: qh: help for the query handler registered
Sep 28 09:51:48 ip-172-31-84-149.ec2.internal nagios[67665]: wproc: Successfully registered manager as @wproc with quer
Sep 28 09:51:48 ip-172-31-84-149.ec2.internal nagios[67665]: wproc: Registry request: name=Core Worker 67669;pid=67669
Sep 28 09:51:48 ip-172-31-84-149.ec2.internal nagios[67665]: wproc: Registry request: name=Core Worker 67666;pid=67666
Sep 28 09:51:48 ip-172-31-84-149.ec2.internal nagios[67665]: wproc: Registry request: name=Core Worker 67667;pid=67667
Sep 28 09:51:48 ip-172-31-84-149.ec2.internal nagios[67665]: wproc: Registry request: name=Core Worker 67668;pid=67668
Sep 28 09:51:49 ip-172-31-84-149.ec2.internal nagios[67665]: Successfully launched command file worker with pid 67670
[ec2-user@ip-172-31-84-149 ~]$
```

**Step 28:** Go back to EC2 Console and copy the Public IP address of this instance. Open up your browser and look for **http://<your\_public\_ip\_address>/nagios**

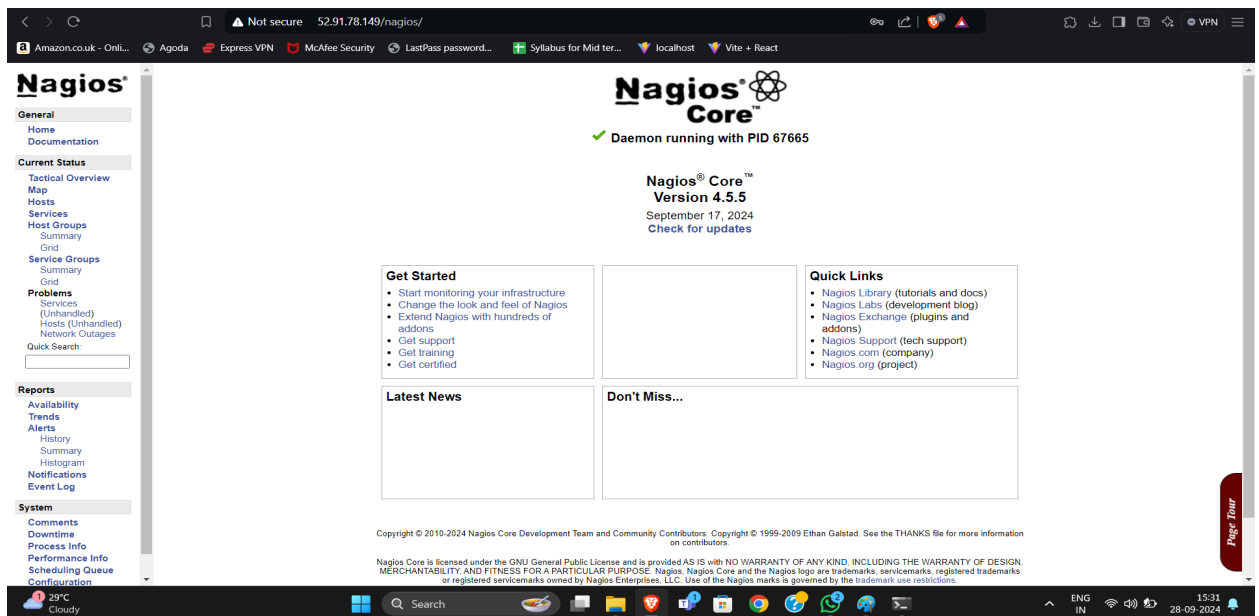
The screenshot displays the AWS Management Console interface for an EC2 instance. The left sidebar shows navigation options like EC2 Dashboard, EC2 Global View, Events, Console-to-Code, and a list of instances. The main panel shows the 'Instance summary' for instance ID i-03b653a01796c6519 (Exp9\_41). The instance is in a 'Running' state. Key details include: Public IPv4 address 52.91.78.149, Private IPv4 address 172.31.84.149, Hostname type IP name: ip-172-31-84-149.ec2.internal, Instance type t2.medium, VPC ID vpc-04fadfb026daa5d28, Subnet ID subnet-0da4329f08fc9981a, and Instance ARN arn:aws:ec2:us-east-1:380557944473:instance/i-03b653a01796c6519. There are also links to 'Connect', 'Instance state', and 'Actions'.

<http://52.91.78.149/nagios/>



Enter username as **nagiosadmin** and password as **Kingmaker**.

**Step 29:** After entering the correct credentials, you will see this page



## CONCLUSION:

In this experiment, we have learned how to install and configure Nagios Core, Nagios Plugins, and NRPE on a Linux machine. We used an Amazon Linux OS instance with the necessary security rules in place. It's important to ensure that the links for Nagios Core and Nagios Plugins are up to date (when using wget). After extracting and configuring these files, we should check for any issues before starting the server. Once everything is set up, we can start the Nagios server. By using the public IP address of the EC2 instance, we can access the Nagios dashboard by navigating to that IP followed by /nagios.