Anushka Shahane D15A 55

ADVANCE DEVOPS CASE STUDY

Introduction

Case Study Overview :

The selected case study explores a practical scenario focusing on deploying a simple application using Kubernetes and ensuring its health through the monitoring capabilities of Nagios. This approach highlights the efficacy of Kubernetes in managing containerized applications and the indispensable role of Nagios in maintaining application health and performance.

In a world where applications are expected to run seamlessly and continuously, ensuring their health through reliable monitoring systems becomes critical. This case study demonstrates a real-world solution where Kubernetes and Nagios are integrated to create a robust and scalable application deployment and monitoring system.

Key Feature and Application :

The unique aspect of this case study is the seamless integration of Kubernetes' orchestration capabilities with Nagios' monitoring strengths. Kubernetes, renowned for its powerful and flexible container orchestration, manages the deployment of an Nginx server. Concurrently, Nagios performs real-time health monitoring of the application, providing alerts and insights to ensure continuous availability and performance.

Kubernetes' capacity to manage containers efficiently makes it an ideal choice for application deployment, while Nagios' monitoring tools ensure that any potential issues are quickly identified and addressed. This integration serves as a practical demonstration of maintaining high availability and reliability in a production environment, making it highly relevant for modern, resilient application management.

The practical use of this integration in a real-world scenario emphasizes the importance of using both deployment and monitoring tools to ensure application health. By combining Kubernetes' deployment efficiency with Nagios' robust monitoring, the case study showcases a comprehensive solution for maintaining application health and availability.

Third-Year Project Integration :

Integrating Kubernetes and Nagios into **GlamEase**, the SaaS-based salon management system, will significantly boost its reliability and scalability. Kubernetes will manage the deployment of Glamease's various services, such as **appointment scheduling** and client management, ensuring they run smoothly and can scale based on demand. If one service fails, Kubernetes will automatically restart it, minimizing downtime. Meanwhile, Nagios will continuously monitor the health of these services, providing real-time alerts if any issues arise. This setup allows prompt addressing of any problems, ensuring a seamless experience for salon owners. By leveraging these tools, Glamease becomes a robust and efficient salon management system, capable of handling increased user loads while maintaining high availability and performance. This integration of modern cloud technologies aligns with industry best practices, preparing for real-world challenges in software development and deployment.

Step-by-Step Explanation

1) **Deploying NGINX server in cloudshell using kubernetes**

Using AWS CloudShell:

Access AWS CloudShell
Log into the AWS Management Console.
Click on the CloudShell icon at the top right of the console.

```
sudo yum -y update

curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.21.14/2022-11-01/bin/linux/amd64/kubectl
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
```

curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin

Create an EKS cluster

eksctl create cluster --name my-cluster --version 1.29 --region us-west-2 --nodegroup-name linux-nodes --node-type t2.micro --nodes 2 --nodes-min 1 --nodes-max 4 --managed

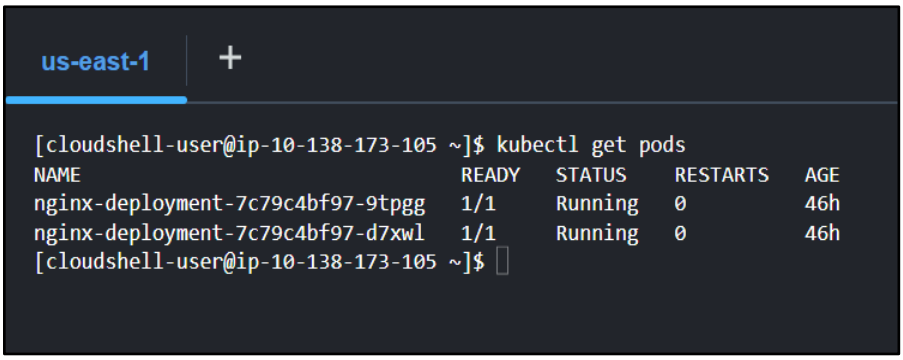
kubectl get svc

Now Create a YAML file named nginx-pod.yaml with the following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx-container
    image: nginx:latest
    ports:
    - containerPort: 80
```

kubectl apply -f nginx-pod.yaml

kubectl get pods



The pods are in running state

Thus , Nginx server is successfully deployed on kubernetes

2) Monitor the health of Nginx server using Nagios :

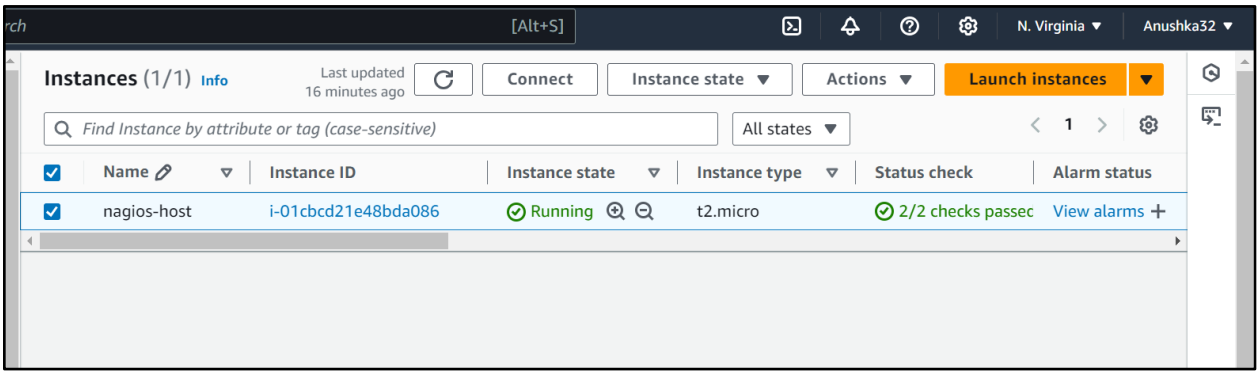
Set Up Nagios for Monitoring

1. Install Nagios Core:
2. Install Nagios Core on a separate machine or server
3. Download the latest version of Nagios Core from the official website
4. Add the necessary Nagios plugins to monitor HTTP services
5. Configure Nagios to check the health of the Nginx pods by adding a command and service check in the Nagios configuration files.

[A] Steps for Installing Nagios

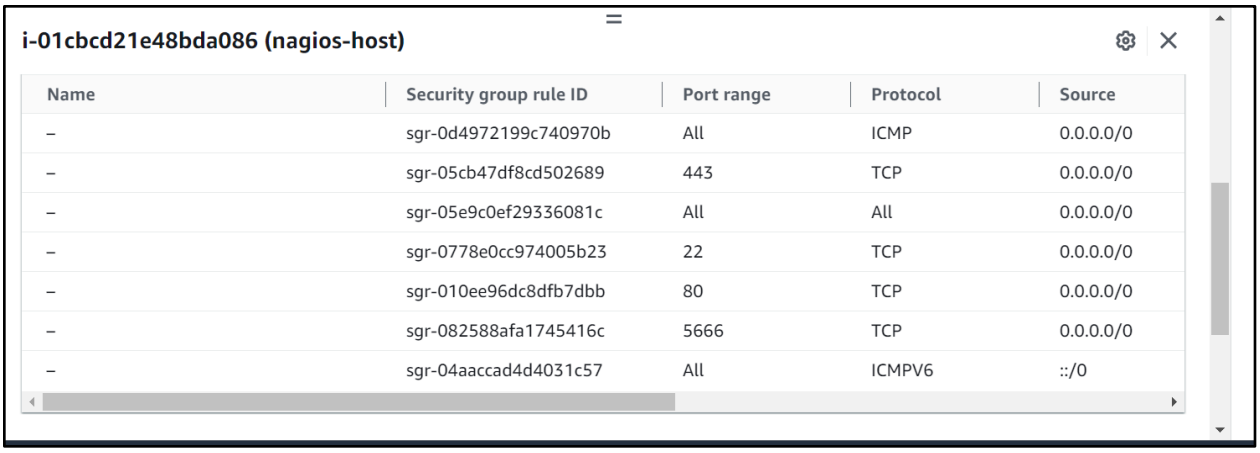
1. Create EC2 Instance:

Launch an Amazon Linux EC2 instance named nagios-host.



2. Configure Security Group:

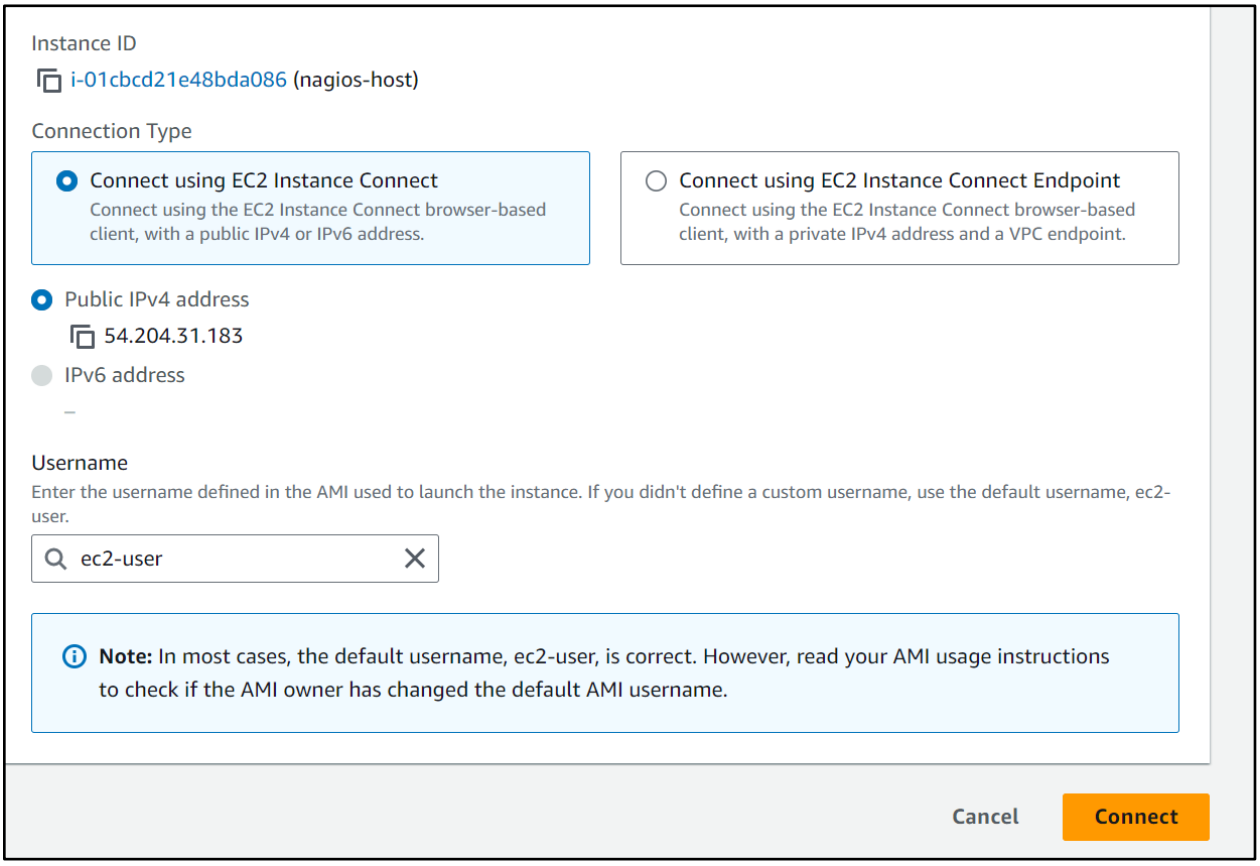
Open inbound rules for HTTP, HTTPS, SSH, and ICMP.



Name	Security group rule ID	Port range	Protocol	Source
–	sgr-0d4972199c740970b	All	ICMP	0.0.0.0/0
–	sgr-05cb47df8cd502689	443	TCP	0.0.0.0/0
–	sgr-05e9c0ef29336081c	All	All	0.0.0.0/0
–	sgr-0778e0cc974005b23	22	TCP	0.0.0.0/0
–	sgr-010ee96dc8dfb7dbb	80	TCP	0.0.0.0/0
–	sgr-082588afa1745416c	5666	TCP	0.0.0.0/0
–	sgr-04aaccad4d4031c57	All	ICMPV6	::/0

3. SSH into EC2 Instance:

Connect to the instance via SSH or EC2 Instance Connect.



4. Update Package Indices and Install Required Packages

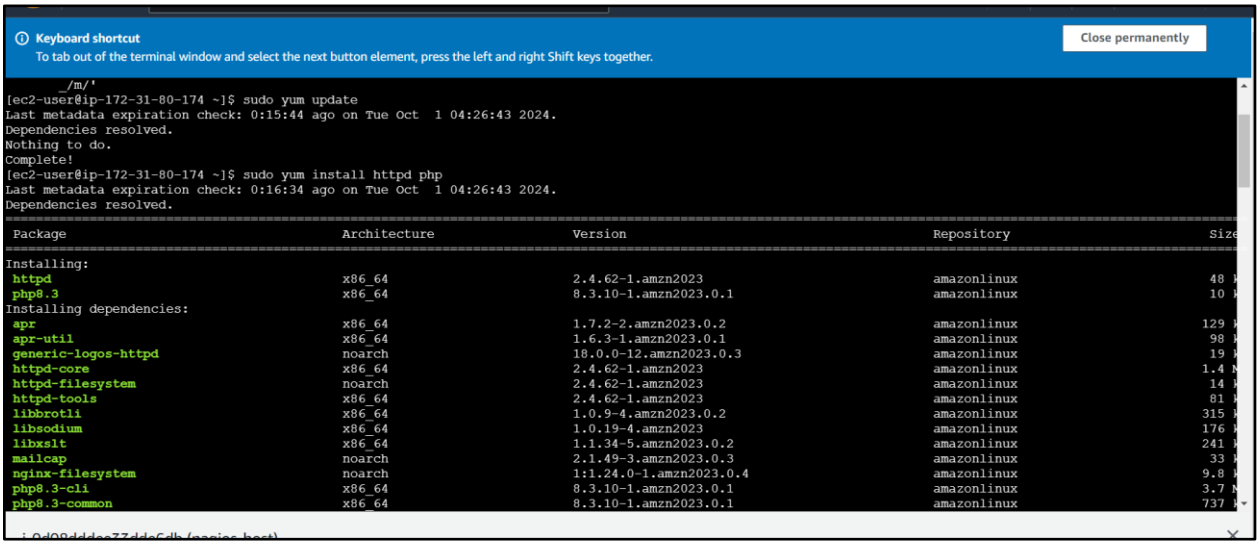
Commands -

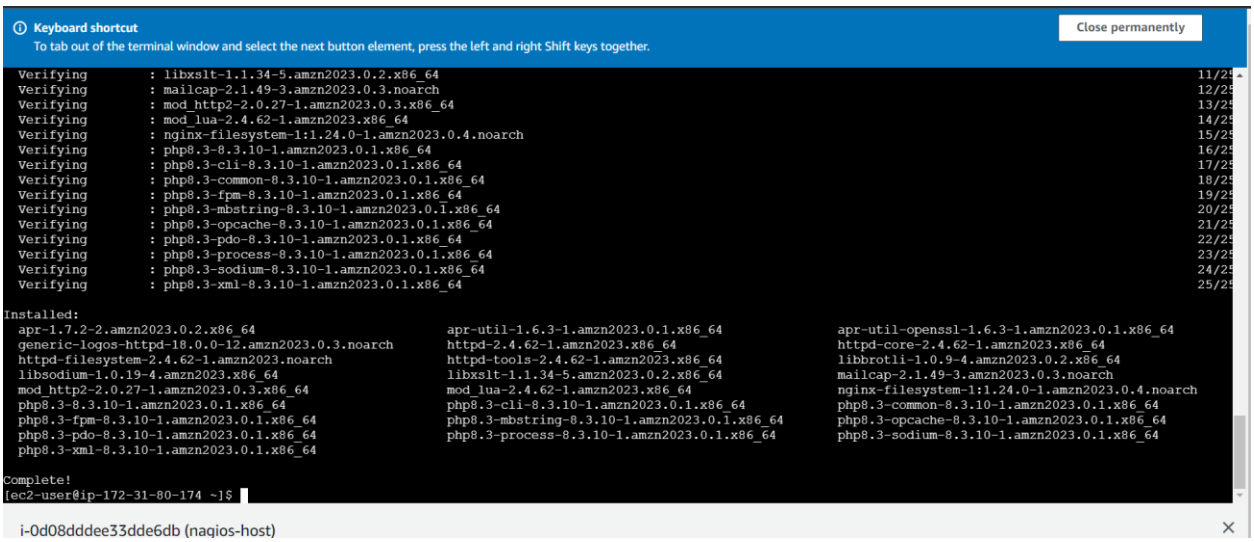
sudo yum update

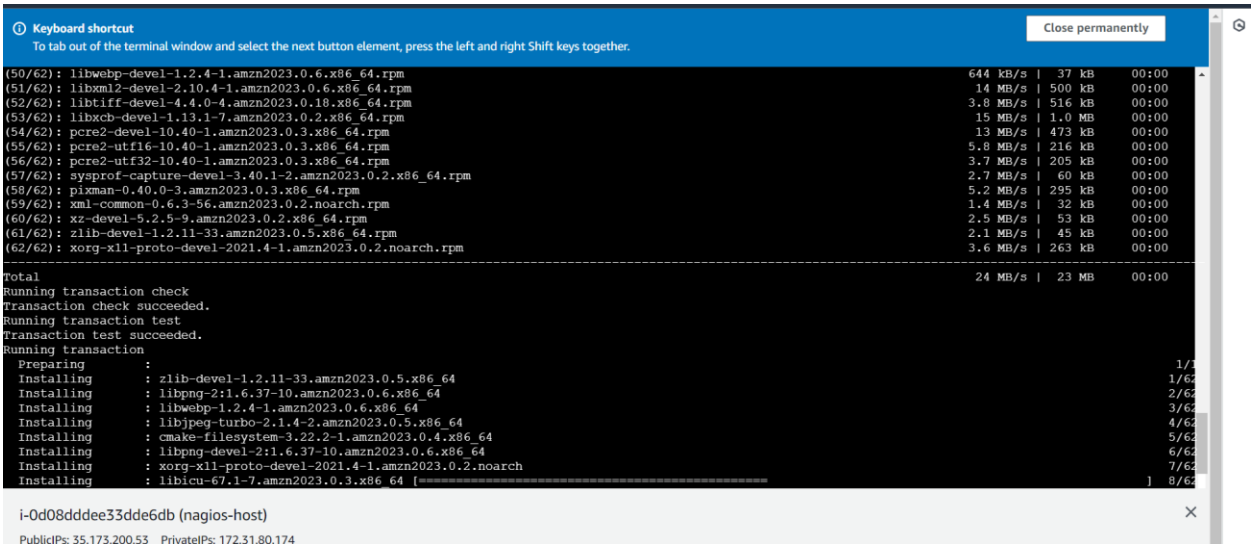
sudo yum install httpd php

sudo yum install gcc glibc glibc-common

sudo yum install gd gd-devel





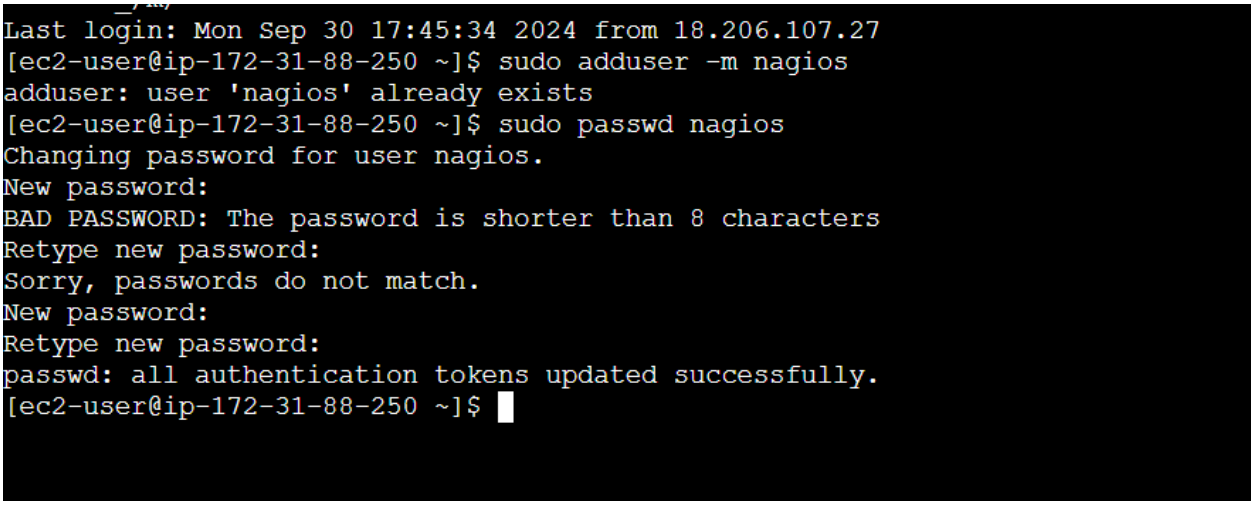


5. Create a New Nagios User

Commands -

sudo adduser -m nagios

sudo passwd nagios



6. Create a New User Group

Commands -

sudo groupadd nagcmd

```
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[ec2-user@ip-172-31-88-250 ~]$ sudo groupadd nagcmd
[ec2-user@ip-172-31-88-250 ~]$ sudo usermod -a -G nagcmd nagios
[ec2-user@ip-172-31-88-250 ~]$ sudo usermod -a -G nagcmd apache
[ec2-user@ip-172-31-88-250 ~]$
```

7. Add Users to the Group

Commands -

sudo usermod -a -G nagcmd nagios

sudo usermod -a -G nagcmd apache

```
[ec2-user@ip-172-31-80-22 ~]$ sudo usermod -aG nagcmd nagios
sudo usermod -aG nagcmd apache
```

8. Create a Directory for Nagios Downloads

Commands -

mkdir ~/downloads

cd ~/downloads

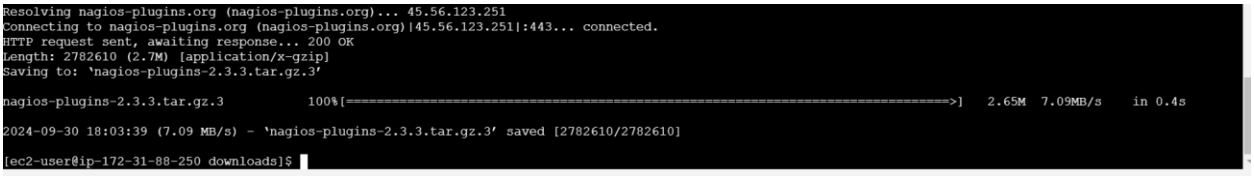
```
[ec2-user@ip-172-31-88-250 ~]$ mkdir ~/downloads
[ec2-user@ip-172-31-88-250 ~]$ cd ~/downloads
```

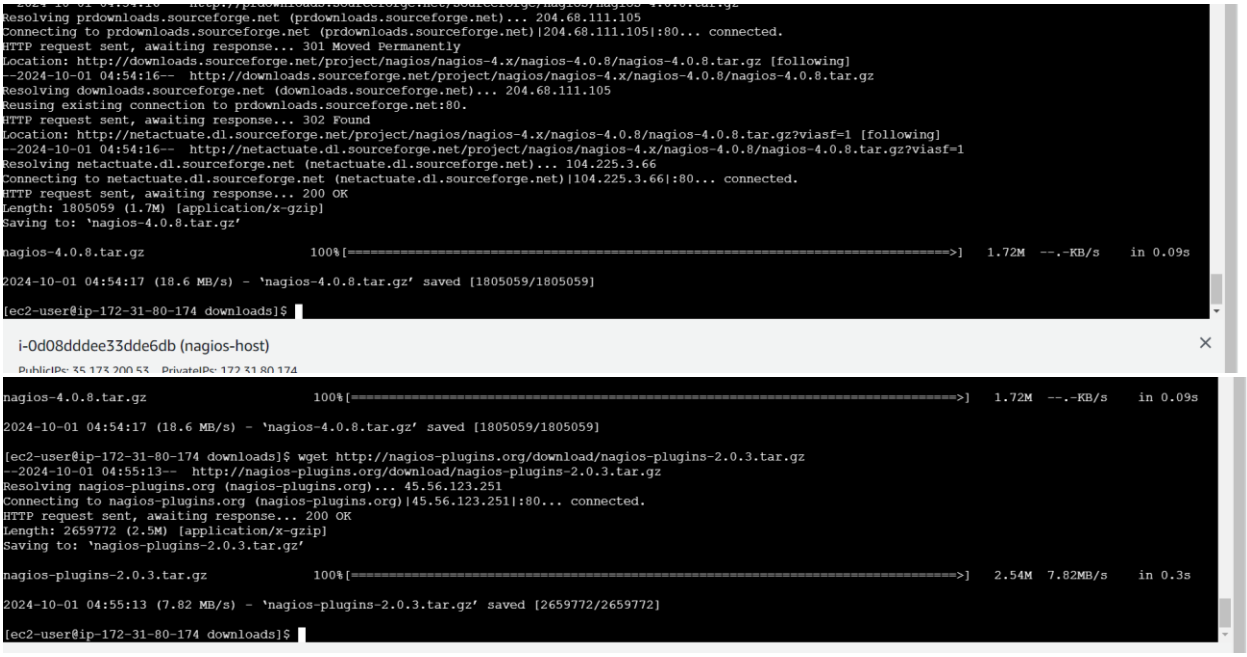
9. Download Nagios and Plugins Source Files

Commands -

wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz

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





10. Extract the Nagios Source File

Commands -

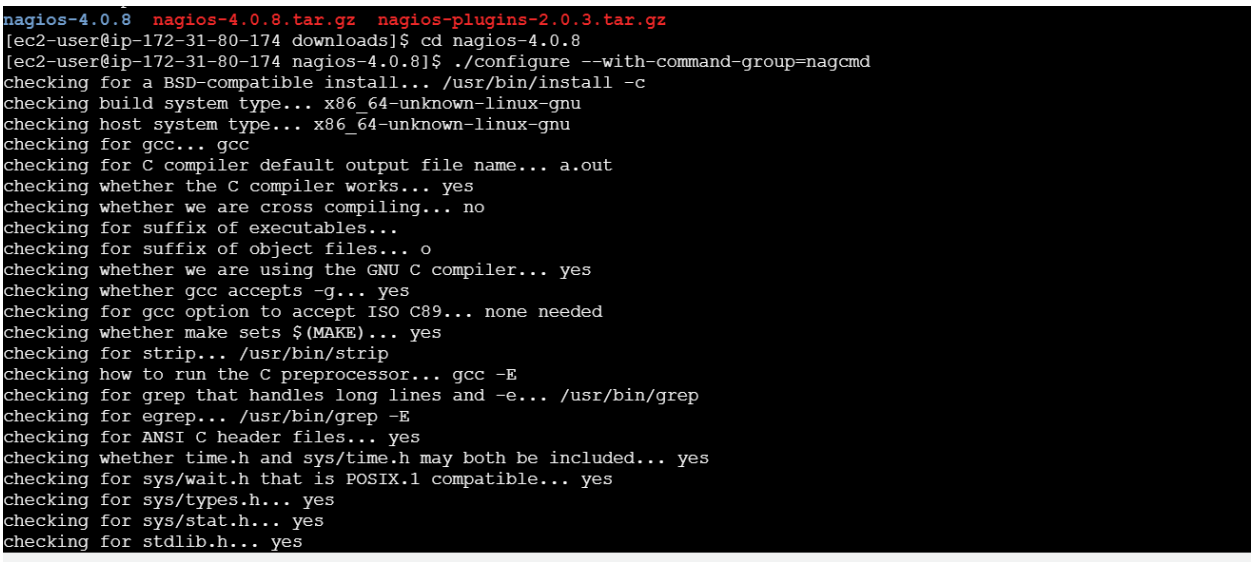
tar zxvf nagios-4.4.6.tar.gz

cd nagios-4.4.6

11. Run the Configuration Script

Commands -

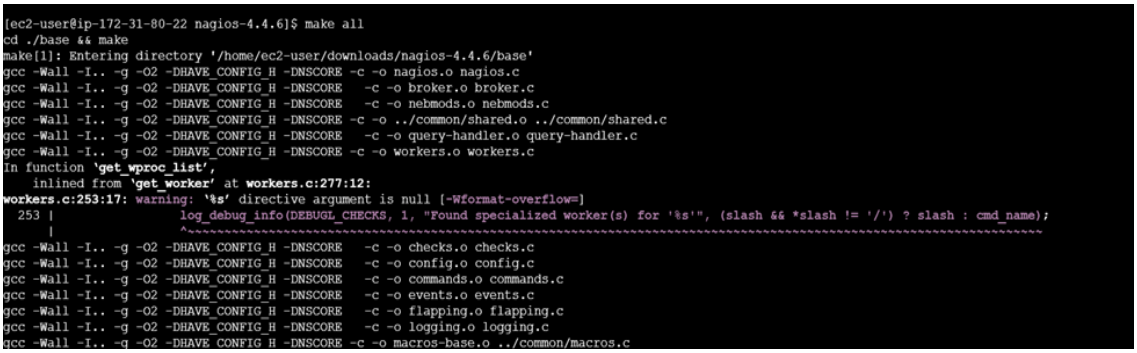
./configure --with-command-group=nagcmd

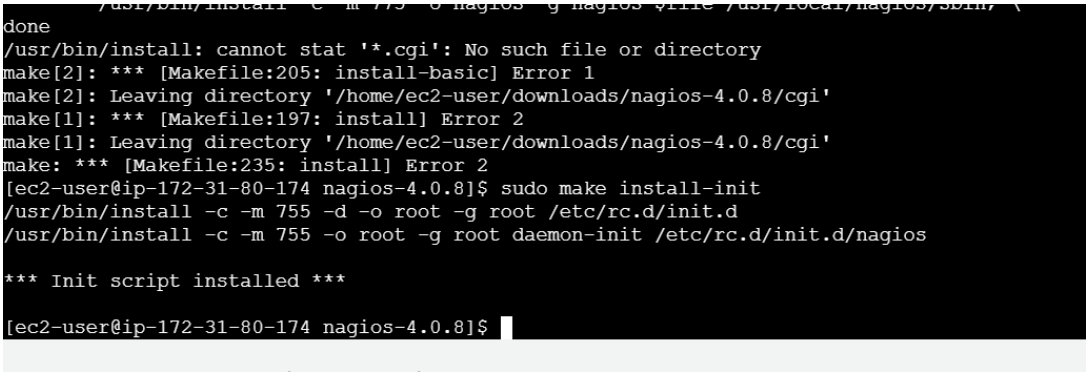


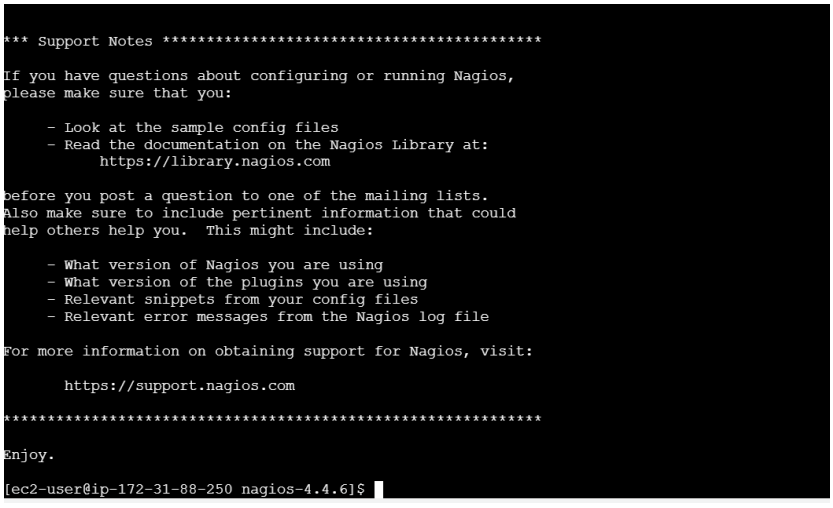
12. Compile the Source Code

Commands -

make all







13. Install Binaries, Init Script, and Sample Config Files

Commands -

./sudo make install

sudo make install-init

sudo make install-config

sudo make install-commandmode

```
*** External command directory configured ***

[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ sudo nano /usr/local/nagios/etc/objects/contacts.cfg
[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf

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

[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ sudo service httpd restart
Redirecting to /bin/systemctl restart httpd.service
[ec2-user@ip-172-31-80-174 nagios-4.0.8]$ cd ~/downloads
[ec2-user@ip-172-31-80-174 downloads]$ tar zxvf nagios-plugins-2.0.3.tar.gz
nagios-plugins-2.0.3/
nagios-plugins-2.0.3/perlmods/
nagios-plugins-2.0.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.0.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.0.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.0.3/perlmods/Makefile.in
nagios-plugins-2.0.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.0.3/perlmods/Makefile.am
nagios-plugins-2.0.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.0.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.0.3/perlmods/Params-Validate-1.08.tar.gz
nagios-plugins-2.0.3/perlmods/Class-Accessor-0.34.tar.gz
```

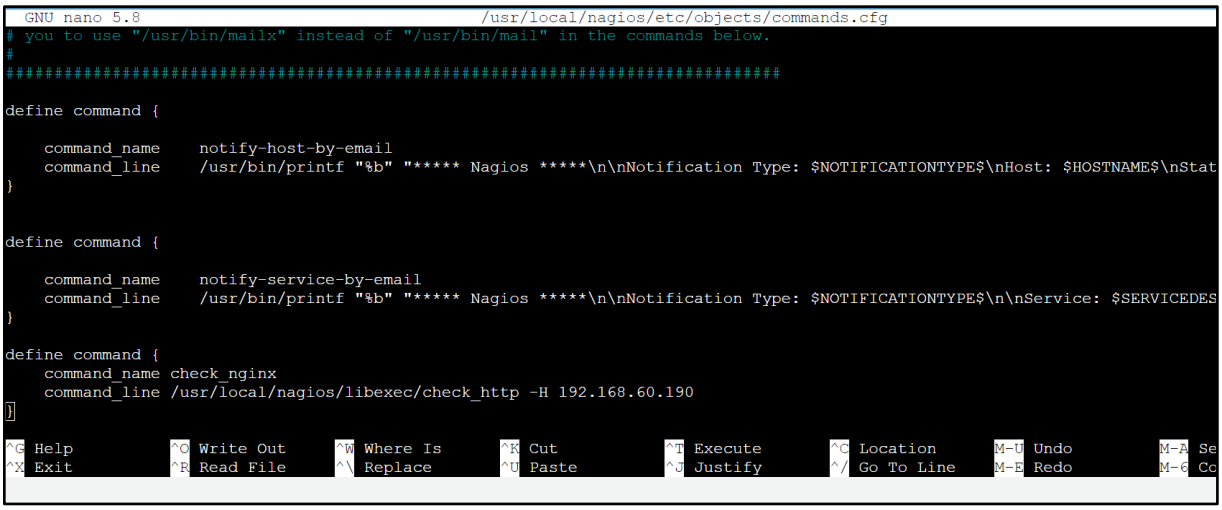
14. Edit the Config File

Commands -

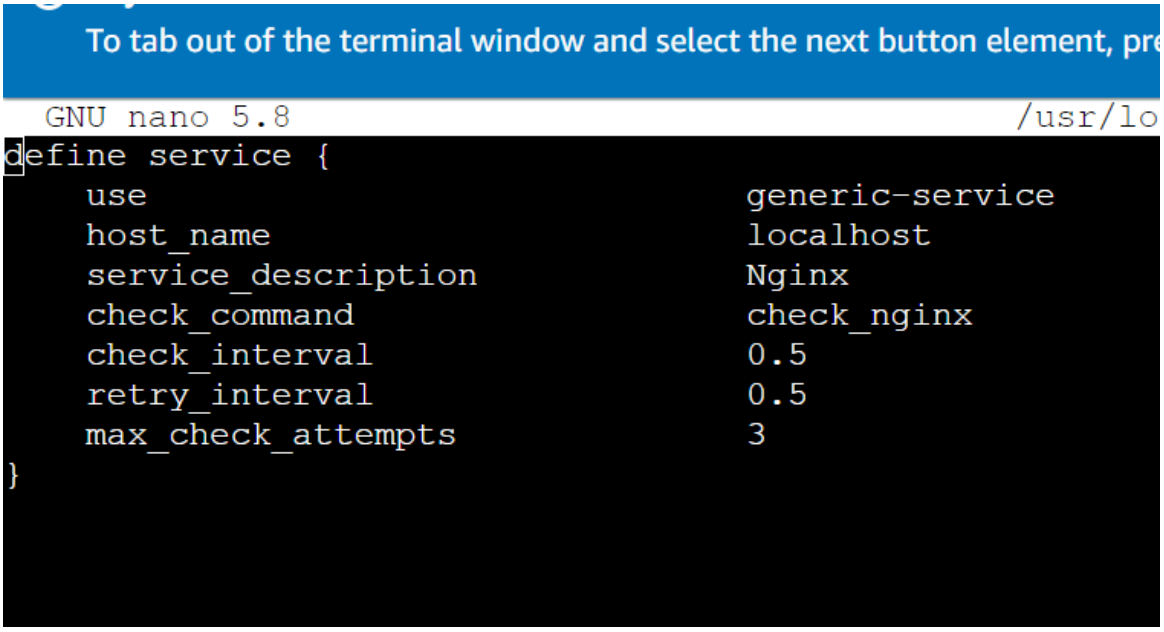
sudo nano /usr/local/nagios/etc/objects/commands.cfg

sudo nano /usr/local/nagios/etc/objects/services.cfg

commands.cfg



services.cfg



15. Configure the Web Interface

Commands -

sudo make install-webconf

```
[ec2-user@ip-172-31-88-250 nagios-4.4.6]$ sudo nano /usr/local/nagios/etc/objects/contacts.
[ec2-user@ip-172-31-88-250 nagios-4.4.6]$ 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-172-31-88-250 nagios-4.4.6]$
```

16. Create a Nagios Admin Account

Commands -

sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

You will be prompted to enter and confirm the password for the nagiosadmin user

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

[ec2-user@ip-172-31-88-250 nagios-4.4.6]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-172-31-88-250 nagios-4.4.6]$
```

```
nagios-plugins-2.3.3/pkg/fedora/requires
nagios-plugins-2.3.3/pkg/solaris/
nagios-plugins-2.3.3/pkg/solaris/preinstall
nagios-plugins-2.3.3/pkg/solaris/solpkg
nagios-plugins-2.3.3/pkg/solaris/pkginfo.in
nagios-plugins-2.3.3/pkg/solaris/pkginfo
nagios-plugins-2.3.3/pkg/redhat/
nagios-plugins-2.3.3/pkg/redhat/requires
[ec2-user@ip-172-31-88-250 downloads]$ cd nagios-plugins-2.3.3
[ec2-user@ip-172-31-88-250 nagios-plugins-2.3.3]$
```

```
make  install-exec-hook
make[3]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins'
cd /usr/local/nagios/libexec && \
for i in check_ftp check_imap check_nntp check_pop check_udp check_clamd  ; do rm -f $i; ln -s check_tcp $i ; done ;\
if [ -x check_ldap ] ; then rm -f check_ldaps ; ln -s check_ldap check_ldaps ; fi
make[3]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins'
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins'
Making install in plugins-scripts
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-scripts'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-scripts'
test -z "/usr/local/nagios/libexec" || /usr/bin/mkdir -p "/usr/local/nagios/libexec"
 /usr/bin/install -c -o nagios -g nagios check_breeze check_disk_smb check_flexlm check_ircd check_log check_oracle check_rpc check_sensors check_wave check_ifs
tatus check_ifoperstatus check_mailq check_file_age check_ssl_validity utils.sh utils.pm '/usr/local/nagios/libexec'
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-scripts'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.3.3/plugins-scripts'
```

17. Restart Apache

Commands -

sudo systemctl restart httpd

```
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$
```

18. Extract the Plugins Source File

Commands -

cd ~/downloads

tar zxvf nagios-plugins-2.3.3.tar.gz

cd nagios-plugins-2.3.3

```
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-80-22 nagios-4.4.6]$ cd ~/downloads
tar zxvf nagios-plugins-2.3.3.tar.gz
cd nagios-plugins-2.3.3
nagios-plugins-2.3.3/
nagios-plugins-2.3.3/perlmods/
nagios-plugins-2.3.3/perlmods/Config-Tiny-2.14.tar.gz
nagios-plugins-2.3.3/perlmods/parent-0.226.tar.gz
nagios-plugins-2.3.3/perlmods/Test-Simple-0.98.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.in
nagios-plugins-2.3.3/perlmods/version-0.9903.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile.am
nagios-plugins-2.3.3/perlmods/Module-Runtime-0.013.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Metadata-1.000014.tar.gz
nagios-plugins-2.3.3/perlmods/Params-Validate-1.08.tar.gz
nagios-plugins-2.3.3/perlmods/Class-Accessor-0.34.tar.gz
nagios-plugins-2.3.3/perlmods/Try-Tiny-0.18.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Implementation-0.07.tar.gz
nagios-plugins-2.3.3/perlmods/Makefile
nagios-plugins-2.3.3/perlmods/Perl-OSType-1.003.tar.gz
nagios-plugins-2.3.3/perlmods/install_order
nagios-plugins-2.3.3/perlmods/Nagios-Plugin-0.36.tar.gz
nagios-plugins-2.3.3/perlmods/Math-Calc-Units-1.07.tar.gz
nagios-plugins-2.3.3/perlmods/Module-Build-0.4007.tar.gz
nagios-plugins-2.3.3/ABOUT-NLS
nagios-plugins-2.3.3/configure.ac
nagios-plugins-2.3.3/Makefile.in
```

```
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.0.3/po'
make[1]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.0.3'
make[2]: Entering directory '/home/ec2-user/downloads/nagios-plugins-2.0.3'
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-2.0.3'
make[1]: Leaving directory '/home/ec2-user/downloads/nagios-plugins-2.0.3'
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$ sudo chkconfig --add nagios
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$ sudo chkconfig nagios on
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.0.8
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 08-12-2014
License: GPL

Website: http://www.nagios.org
Reading configuration data...
Error in configuration file '/usr/local/nagios/etc/nagios.cfg' - Line 452 (Check result path '/usr/local/nagios/var/spool/checkresults' is not a valid directory)
   Error processing main config file!

[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$
```

```
   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...
        Checked 1 hosts
        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
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$
```

If Error
sudo mkdir -p /usr/local/nagios/var/spool/checkresults
sudo chown -R nagios:nagcmd /usr/local/nagios/var

```
Things look okay - No serious problems were detected during the pre-flight check
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$ sudo service nagios start
Reloading systemd:                                         [  OK  ]
Starting nagios (via systemctl):                           [  OK  ]
[ec2-user@ip-172-31-80-174 nagios-plugins-2.0.3]$
```

19. Compile and Install Plugins

Commands -

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

make

sudo make install

```
[ec2-user@ip-172-31-80-22 nagios-plugins-2.3.3]$ ./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
sudo make install
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 to disable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
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 for style of include used by make... GNU
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
```

20. Start Nagios

Commands -

sudo chkconfig --add nagios

sudo chkconfig nagios on

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

sudo systemctl start nagios

21.Check Nagios Status:

sudo systemctl status nagios

```
      _/m/'
Last login: Mon Oct 21 16:50:18 2024 from 18.206.107.29
[ec2-user@ip-172-31-34-142 ~]$ ^[[200~sudo systemctl restart nagios
-bash: $'\E[200~sudo': command not found
[ec2-user@ip-172-31-34-142 ~]$ sudo systemctl restart nagios
[ec2-user@ip-172-31-34-142 ~]$ sudo systemctl status nagios
● nagios.service - Nagios Core 4.4.6
     Loaded: loaded (/usr/lib/systemd/system/nagios.service; enabled; preset: disabled)
     Active: active (running) since Mon 2024-10-21 17:06:18 UTC; 2min 19s ago
       Docs: https://www.nagios.org/documentation
    Process: 3817 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
    Process: 3819 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg (code=exited, status=0/SUCCESS)
   Main PID: 3820 (nagios)
      Tasks: 6 (limit: 1112)
     Memory: 1.9M
        CPU: 46ms
     CGroup: /system.slice/nagios.service
             ├─3820 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
             ├─3821 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─3822 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─3823 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             ├─3824 /usr/local/nagios/bin/nagios --worker /usr/local/nagios/var/rw/nagios.qh
             └─3825 /usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

Nagios Page will be visible:

Username : nagiosadmin
Password : your password

[B] Monitoring nginx server using nagios

Go to services :

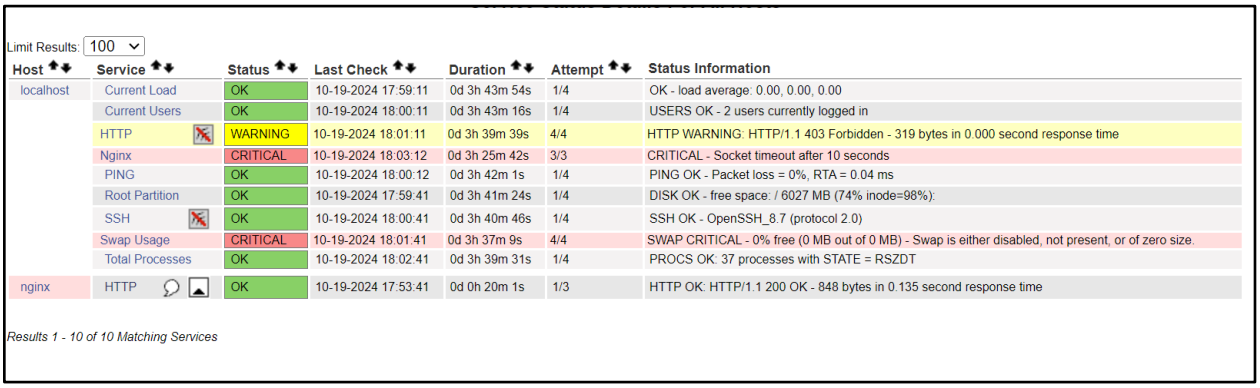
I] At first we have scaled up the pods so you will see the condition of the nginx server as OK.

If → **kubectl scale deployment nginx-deployment --replicas=2 : to scale up**

```
No resources found in default namespace.
[cloudshell-user@ip-10-138-173-105 ~]$ kubectl scale deployment nginx-deployment --replicas=2
deployment.apps/nginx-deployment scaled
[cloudshell-user@ip-10-138-173-105 ~]$
[cloudshell-user@ip-10-138-173-105 ~]$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE                                                NOMINATED NODE   READINESS GATES
nginx-deployment-7c79c4bf97-g7lvf   1/1     Running   0          4s    192.168.87.116   ip-192-168-84-41.us-west-2.compute.internal    <none>           <none>
nginx-deployment-7c79c4bf97-tl5nl   1/1     Running   0          4s    192.168.60.190   ip-192-168-43-230.us-west-2.compute.internal   <none>           <none>
[cloudshell-user@ip-10-138-173-105 ~]$
```

The pods are in running state thus the condition of nginx server is OK

nginx	HTTP	OK	10-19-2024 17:53:41	0d 0h 20m 1s	1/3	HTTP OK: HTTP/1.1 200 OK - 848 bytes in 0.135 second response time

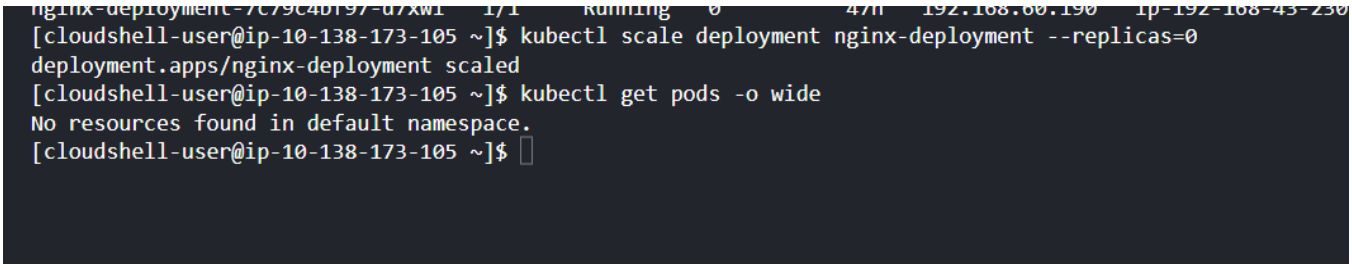


Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	10-19-2024 17:59:11	0d 3h 43m 54s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-19-2024 18:00:11	0d 3h 43m 16s	1/4	USERS OK - 2 users currently logged in
	HTTP	WARNING	10-19-2024 18:01:11	0d 3h 39m 39s	4/4	HTTP WARNING: HTTP/1.1 403 Forbidden - 319 bytes in 0.000 second response time
	Nginx	CRITICAL	10-19-2024 18:03:12	0d 3h 25m 42s	3/3	CRITICAL - Socket timeout after 10 seconds
	PING	OK	10-19-2024 18:00:12	0d 3h 42m 1s	1/4	PING OK - Packet loss = 0%, RTA = 0.04 ms
	Root Partition	OK	10-19-2024 17:59:41	0d 3h 41m 24s	1/4	DISK OK - free space: / 6027 MB (74% inode=98%):
	SSH	OK	10-19-2024 18:00:41	0d 3h 40m 46s	1/4	SSH OK - OpenSSH_8.7 (protocol 2.0)
	Swap Usage	CRITICAL	10-19-2024 18:01:41	0d 3h 37m 9s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
	Total Processes	OK	10-19-2024 18:02:41	0d 3h 39m 31s	1/4	PROCS OK: 37 processes with STATE = RSZDT
nginx	HTTP	OK	10-19-2024 17:53:41	0d 0h 20m 1s	1/3	HTTP OK: HTTP/1.1 200 OK - 848 bytes in 0.135 second response time

Results 1 - 10 of 10 Matching Services

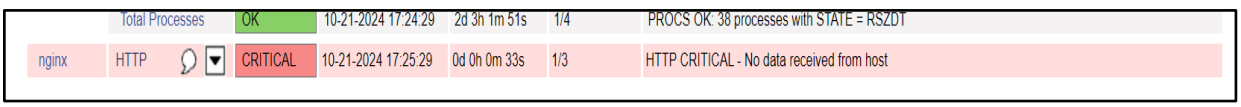
II] If → **kubectl scale deployment nginx-deployment --replicas=0 : To scale down pods**



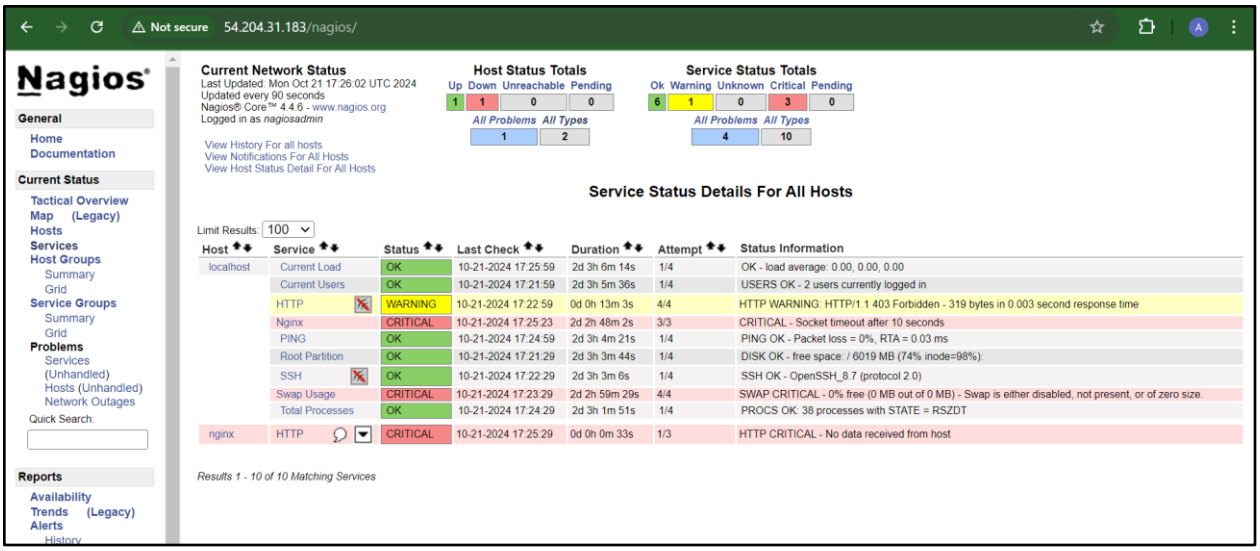
```
nginx-deployment-7c79c4bf97-d7xwl   1/1     Running   0          47m   192.168.60.190   ip-192-168-43-230
[cloudshell-user@ip-10-138-173-105 ~]$ kubectl scale deployment nginx-deployment --replicas=0
deployment.apps/nginx-deployment scaled
[cloudshell-user@ip-10-138-173-105 ~]$ kubectl get pods -o wide
No resources found in default namespace.
[cloudshell-user@ip-10-138-173-105 ~]$
```

The pods are scaled down so you will see that the nginx server is in critical condition.

Along with a error : No data received from host



	Total Processes	OK	10-21-2024 17:24:29	2d 3h 1m 51s	1/4	PROCS OK: 38 processes with STATE = RSZDT
nginx	HTTP	CRITICAL	10-21-2024 17:25:29	0d 0h 0m 33s	1/3	HTTP CRITICAL - No data received from host



Thus the Nginx server deployed on kubernetes is monitored using nagios successfully !

Conclusion : Thus, this case study illustrates the effectiveness of using Kubernetes and Nagios to deploy and monitor a simple application like an Nginx server. By utilizing AWS Cloud9, Kubernetes ensures the seamless deployment and management of the application, providing scalability and resilience. Nagios complements this by continuously monitoring the application's health, ensuring any issues are promptly detected and addressed. This setup not only demonstrates the practical application of these technologies but also highlights their importance in maintaining application availability and performance. Such integrations are vital for creating robust, scalable, and reliable systems in modern cloud environments.