Case Study: Kubernetes Application with Basic Monitoring

Aim : **Kubernetes Application with Basic Monitoring**

- Concepts Used: Kubernetes, AWS Cloud9, and Nagios.
- Problem Statement: "Deploy a simple application (e.g., an Nginx server) on a Kubernetes cluster using AWS Cloud9 and monitor its health using Nagios."
- Tasks:
  - Use AWS Cloud9 to deploy the Nginx server on a Kubernetes cluster.
  - Install and configure Nagios to monitor the Nginx pod's availability.
  - Verify that Nagios can detect when the Nginx pod is running or not.

Note: Due to the deprecation of AWS Cloud9 for Kubernetes-based application deployments, the decision was made to use Google Cloud Console for this experiment. Google Cloud Console provides a more robust environment for deploying and managing Kubernetes clusters, along with seamless integration with monitoring tools like Nagios.

1.Introduction

In the landscape of modern cloud computing, the orchestration of containerized applications has become critical for enhancing scalability, availability, and resilience. This case study explores the theoretical underpinnings of deploying a simple Nginx server on a Kubernetes cluster and monitoring its health using Nagios, focusing on the implications of these technologies in cloud-native environments.

2.Theoretical Framework

2.1. Containerization and Orchestration

Containerization is a lightweight form of virtualization that encapsulates an application and its dependencies into a single container. This approach facilitates consistent environments across development, testing, and production, thereby reducing deployment failures. Kubernetes, an open-source orchestration platform, automates the deployment, scaling, and management of containerized applications, enabling efficient resource utilization and improved application reliability.

Key Concepts:

- **Containers:** Isolated environments for running applications.
- **Orchestration:** Management of multiple containers, ensuring they run seamlessly in a distributed system.

- **Microservices Architecture:** A design approach where applications are structured as a collection of loosely coupled services, enhancing modularity and enabling easier scaling.

2.2 Monitoring in Distributed Systems

Monitoring is critical for maintaining the health and performance of applications. In distributed systems like Kubernetes, traditional monitoring approaches may fall short due to the dynamic nature of containers. Nagios, a widely used open-source monitoring system, provides a framework for monitoring application availability, performance, and health through customizable checks and alerts.

Key Concepts:

- **Health Checks:** Regular checks to assess the status of applications and services.
- **Alerts:** Notifications triggered by specific conditions indicating issues that need attention.
- **Service-Level Objectives (SLOs):** Metrics that define expected reliability and performance levels.

3.Methodology

3.1. Environment Setup

1. **Cloud Provider Selection:**
    - Due to the deprecation of AWS Cloud9 for Kubernetes deployments, Google Cloud Console was chosen as the preferred environment for deploying and managing the Kubernetes cluster.
2. **Kubernetes Cluster Creation:**
    - A Kubernetes cluster was created in Google Cloud Console, following best practices for cluster configuration and management.

3.2. Application Deployment

1. **Nginx Deployment:**
    - An Nginx server was deployed using Kubernetes deployment manifests, which define the desired state of the application, including the number of replicas, container images, and service exposure.

2. **Service Exposure:**

    - The Nginx application was exposed to the internet using a LoadBalancer service, allowing external access to the application.

3.3. Monitoring Setup

1. **Nagios Installation:**
    - Nagios was installed on a separate virtual machine, following its documentation for installation and configuration.
2. **Monitoring Configuration:**
    - Nagios was configured to monitor the Nginx server by defining service checks that assess the availability and performance of the application.
    - Custom command definitions were created to facilitate health checks using HTTP requests.

3 **Alerting Mechanism:**

- Nagios was configured to send alerts via email when the Nginx server was down or unreachable, providing timely notifications to the operations team.

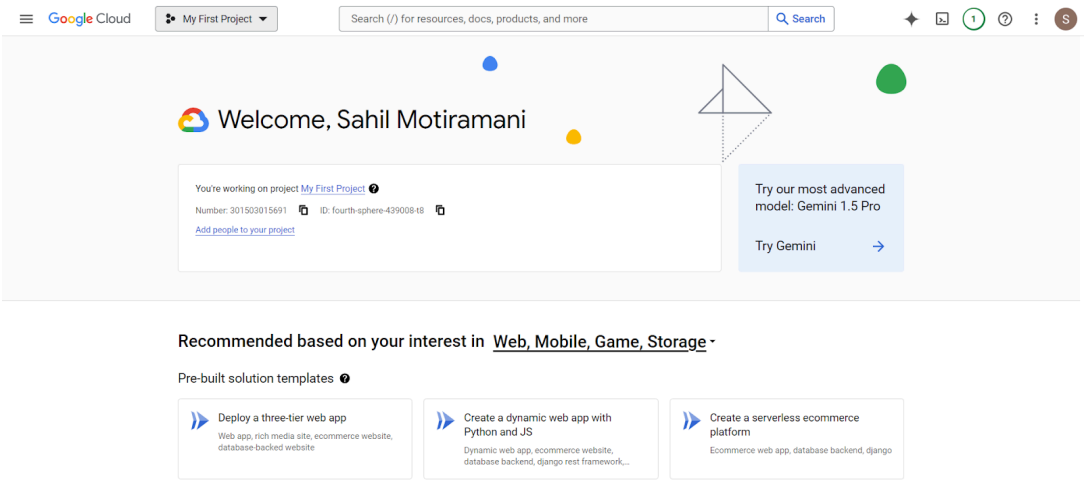
4. Steps performed:

Step 1:Make a account on Google cloud console:

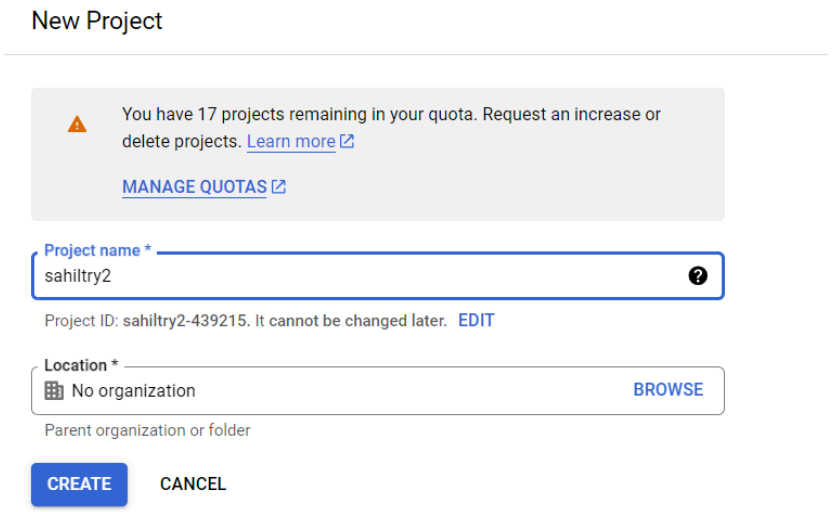
https://console.cloud.google.com/

After making account,

Screen will be as shown below:



Step2:Now you need to make a new project:



Step 3:Install Google cloud sdk
https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe
From this link

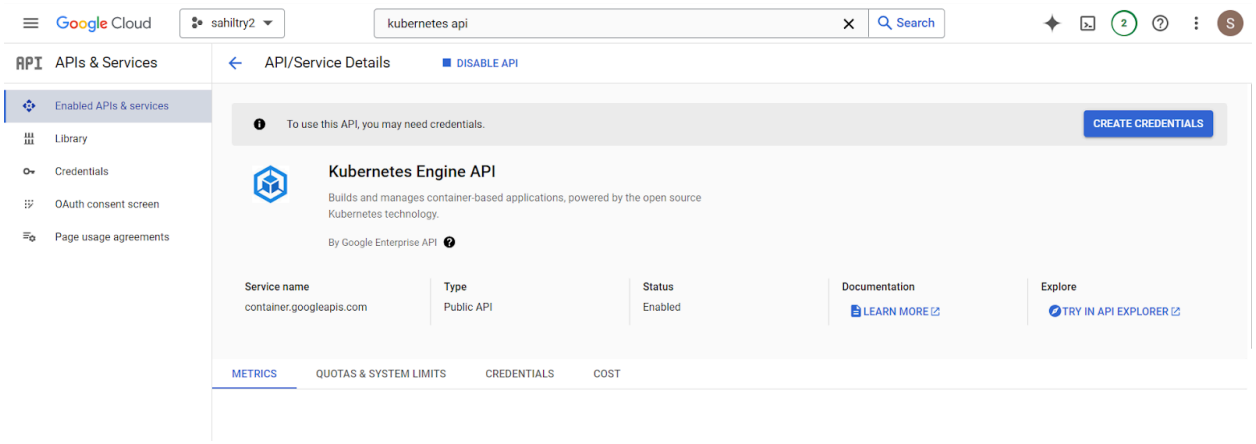
Step 4:
Authenticate your account
gcloud auth login

Set your GCP project
gcloud config set project <PROJECT_ID>

```
C:\Users\HP\AppData\Local\Google\Cloud SDK>gcloud auth login
Your browser has been opened to visit:

    https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fsqlservice.login+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&state=iEMYGEWPwqFrClMxsnMJuV2ZF9AeDO&access_type=offline&code_challenge=BHD03_zENAHKHRkwPih661rJ2tuAQelTho9hQ5EjMc4&code_challenge_method=S256


You are now logged in as [msd201104@gmail.com].
Your current project is [sahilassignment1].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID

C:\Users\HP\AppData\Local\Google\Cloud SDK>gcloud config set project sahiltry2
Updated property [core/project].

C:\Users\HP\AppData\Local\Google\Cloud SDK>
```

Name:Sahil Motiramani Roll No:35 Div:D15C

Step 5:Enable Kubernetes API:



Step 6:use gcloud to create a GKE cluster from your terminal
gcloud container clusters create my-cluster --num-nodes=3 --zone us-central1-a

```
C:\Users\HP\AppData\Local\Google\Cloud SDK>gcloud container clusters create my-cluster --num-nodes=3 --zone us-central1-a
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. Se
e https://cloud.google.com/kubernetes-engine/docs/how-to/disable-kubelet-readonly-port for ways to check usage and for migratio
n instructions.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster my-cluster in us-central1-a... Cluster is being health-checked (Kubernetes Control Plane is healthy)..
.done.
Created [https://container.googleapis.com/v1/projects/sahiltry2/zones/us-central1-a/clusters/my-cluster].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-central1-a/my-c
luster?project=sahiltry2
kubeconfig entry generated for my-cluster.
NAME        LOCATION       MASTER_VERSION      MASTER_IP       MACHINE_TYPE  NODE_VERSION        NUM_NODES  STATUS
my-cluster  us-central1-a  1.30.5-gke.1014001  35.223.251.161  e2-medium     1.30.5-gke.1014001  3          RUNNING

C:\Users\HP\AppData\Local\Google\Cloud SDK>
```

Step 6:
gcloud container clusters get-credentials my-cluster --zone us-central1-a
Connect to GKE Cluster
Get cluster credentials to interact with the Kubernetes cluster

```
C:\Users\HP\AppData\Local\Google\Cloud SDK>gcloud container clusters get-credentials my-cluster --zone us-central1-a
Fetching cluster endpoint and auth data.
kubeconfig entry generated for my-cluster.

C:\Users\HP\AppData\Local\Google\Cloud SDK>
```

Step 7:
Create Nginx Deployment
Use kubectl to deploy an Nginx server

kubectl create deployment nginx --image=nginx

```
C:\Users\HP\AppData\Local\Google\Cloud SDK>kubectl create deployment nginx --image=nginx
deployment.apps/nginx created

C:\Users\HP\AppData\Local\Google\Cloud SDK>
```

Step 8:Expose the Nginx deployment as a service
kubectl expose deployment nginx --type=LoadBalancer --port=80

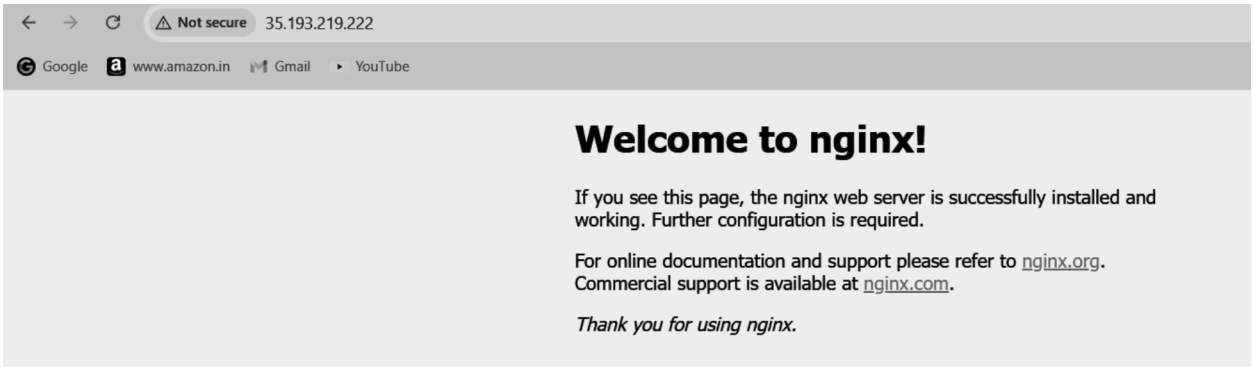
```
C:\Users\HP\AppData\Local\Google\Cloud SDK>kubectl expose deployment nginx --type=LoadBalancer --port=80
service/nginx exposed

C:\Users\HP\AppData\Local\Google\Cloud SDK>
```

This creates a load balancer that allows you to access the Nginx application externally.

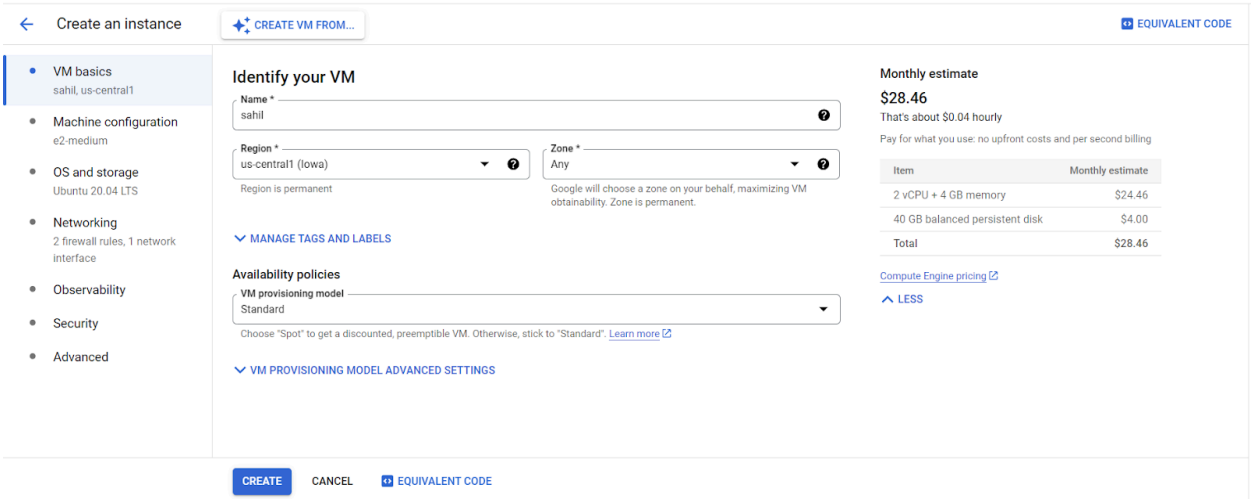
Step 9:
To get the external IP address of the service:
kubectl get services

```
C:\Users\HP\AppData\Local\Google\Cloud SDK>kubectl get services
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
kubernetes   ClusterIP      34.118.224.1    <none>           443/TCP        5m53s
nginx        LoadBalancer   34.118.235.241  35.193.219.222   80:32372/TCP   39s

C:\Users\HP\AppData\Local\Google\Cloud SDK>
```

Once the external IP is available, you can access the Nginx server at
http://35.193.219.222

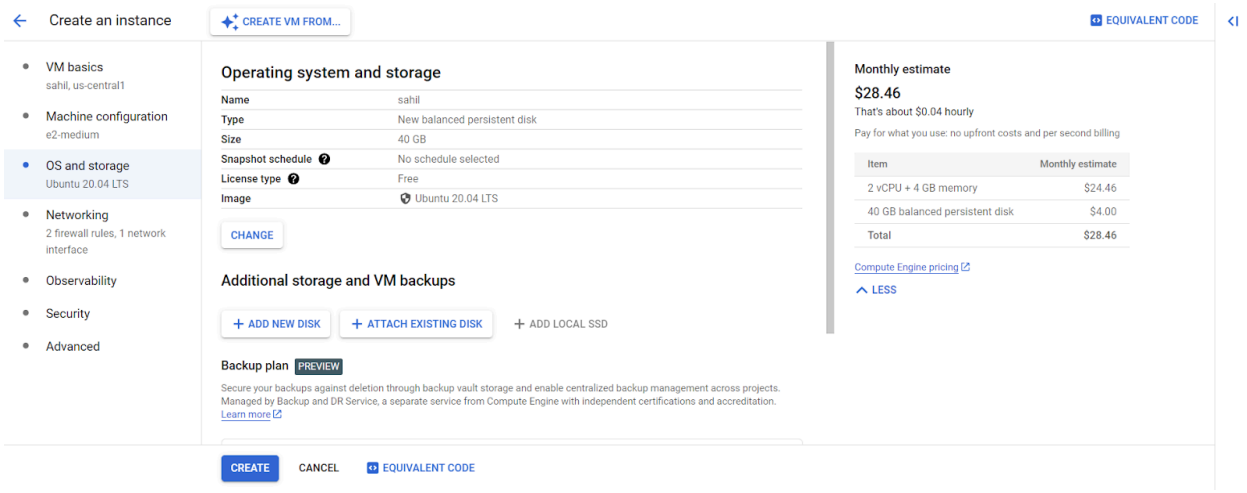


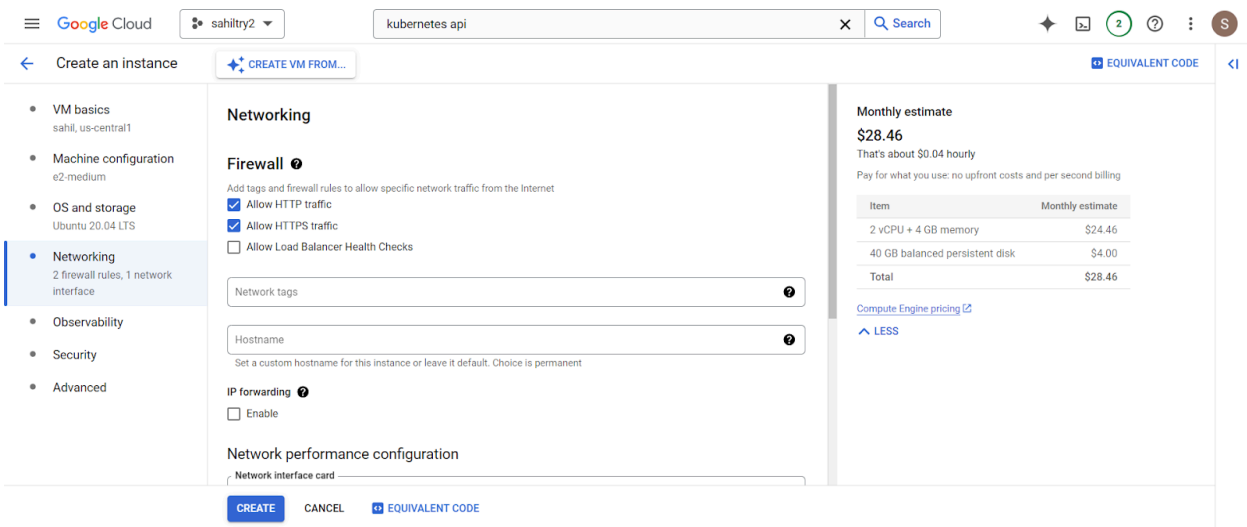
Step10: Create a VM Instance:



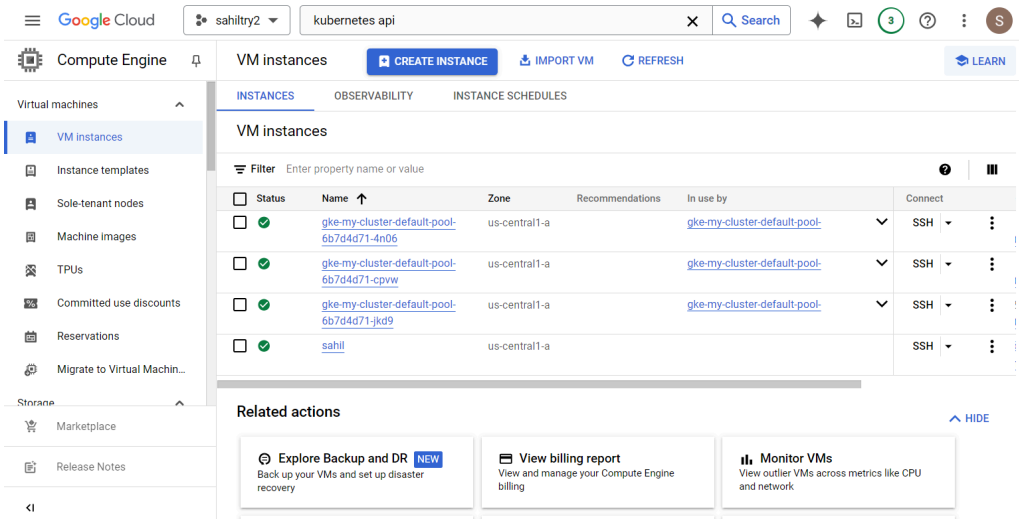
Name:Sahil Motiramani Roll No:35 Div:D15C

Do these configurations;





Your instance will get created



Step 11:SSH into the VM and install the necessary dependencies:
sudo apt update

```
msd201104@sahil:~$ sudo apt update
Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:3 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get:5 http://us-central1.gce.archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:6 http://us-central1.gce.archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]
Get:7 http://us-central1.gce.archive.ubuntu.com/ubuntu focal/universe amd64 c-n-f Metadata [265 kB]
Get:8 http://us-central1.gce.archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [144 kB]
Get:9 http://us-central1.gce.archive.ubuntu.com/ubuntu focal/multiverse Translation-en [104 kB]
Get:10 http://us-central1.gce.archive.ubuntu.com/ubuntu focal/multiverse amd64 c-n-f Metadata [9136 B]
Get:11 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [3640 kB]
Get:12 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [558 kB]
Get:13 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [3330 kB]
Get:14 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [465 kB]
Get:15 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1237 kB]
Get:16 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [297 kB]
Get:17 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [28.3 kB]
Get:18 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [27.1 kB]
Get:19 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/multiverse Translation-en [7936 B]
Get:20 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 c-n-f Metadata [616 B]
Get:21 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-backports/main amd64 Packages [45.7 kB]
Get:22 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-backports/main Translation-en [16.3 kB]
```

Step 12:Install dependency:
sudo apt install -y autoconf gcc libc6 make wget unzip apache2 apache2-utils php libgd-dev

```
msd201104@sahil:~$ sudo apt install -y autoconf gcc libc6 make wget unzip apache2 apache2-utils php libgd-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
libc6 is already the newest version (2.31-0ubuntu9.16).
libc6 set to manually installed.
wget is already the newest version (1.20.3-1ubuntu2.1).
wget set to manually installed.
The following additional packages will be installed:
  apache2-bin apache2-data automake autotools-dev binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-9
  fontconfig-config fonts-dejavu-core gcc-9 gcc-9-base libapache2-mod-php7.4 libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libasan5 libatomic1 libbinutils libc-dev-bin libc6-dev libcc1-0
  libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl libexpat1-dev libfile-fcntllock-perl libfontconfig1
  libfontconfig1-dev libfreetype-dev libfreetype6-dev libgcc-9-dev libgd3 libgomp1 libice-dev libice6 libisl22
  libitm1 libjansson4 libjbig-dev libjbig0 libjpeg-dev libjpeg-turbo8 libjpeg-turbo8-dev libjpeg8 libjpeg8-dev
  liblsan0 liblua5.2-0 liblzma-dev libmpc3 libpng-dev libpng-tools libpthread-stubs0-dev libquadmath0 libsm-dev
  libsm6 libtiff-dev libtiff5 libtiffxx5 libtsan0 libubsan1 libvpx-dev libvpx6 libwebp6 libx11-dev libxau-dev
  libxcb1-dev libxdmcp-dev libxpm-dev libxpm4 libxt-dev libxt6 linux-libc-dev m4 manpages-dev php-common php7.4
  php7.4-cli php7.4-common php7.4-json php7.4-opcache php7.4-readline pkg-config ssl-cert uuid-dev x11-common
  x11proto-core-dev x11proto-dev xorg-sgml-doctools xtrans-dev zlib1g-dev
```

Step 13:Download and install Nagios
cd /tmp
wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
tar -xzf nagios-4.4.6.tar.gz
cd nagios-4.4.6
./configure --with-httpd-conf=/etc/apache2/sites-enabled
make all

```
msd201104@sahil:~$ cd /tmp
msd201104@sahil:/tmp$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.tar.gz
--2024-10-20 15:23:40--  https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.4.6.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: 11333414 (11M) [application/x-gzip]
Saving to: 'nagios-4.4.6.tar.gz'

nagios-4.4.6.tar.gz          100%[=========================================>]  10.81M  38.6MB/s    in 0.3s

2024-10-20 15:23:41 (38.6 MB/s) - 'nagios-4.4.6.tar.gz' saved [11333414/11333414]

msd201104@sahil:/tmp$ tar -xzf nagios-4.4.6.tar.gz
msd201104@sahil:/tmp$ cd nagios-4.4.6
msd201104@sahil:/tmp/nagios-4.4.6$
```

```
msd201104@sahil:/tmp/nagios-4.4.6$ ./configure --with-httpd-conf=/etc/apache2/sites-enabled
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking whether ln -s works... yes
checking for strip... /usr/bin/strip
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for ANSI C header files... yes
checking whether time.h and sys/time.h may both be included... yes
checking for sys/wait.h that is POSIX.1 compatible... yes
```

```
msd201104@sahil:/tmp/nagios-4.4.6$ make all
cd ./base && make
make[1]: Entering directory '/tmp/nagios-4.4.6/base'
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c
nagios.c: In function 'main':
nagios.c:611:4: warning: ignoring return value of 'asprintf', declared with attribute warn_unused_result [-Wunus
d-result]
  611 |    asprintf(&mac->x[MACRO_PROCESSSTARTTIME], "%llu", (unsigned long long)program_start);
      |    ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
nagios.c:841:4: warning: ignoring return value of 'asprintf', declared with attribute warn_unused_result [-Wunus
d-result]
  841 |    asprintf(&mac->x[MACRO_EVENTSTARTTIME], "%llu", (unsigned long long)event_start);
      |    ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
nagios.c: In function 'nagios_core_worker':
nagios.c:176:3: warning: ignoring return value of 'read', declared with attribute warn_unused_result [-Wunused-r
sult]
  176 |   read(sd, response + 3, sizeof(response) - 4);
      |   ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
nagios.c: In function 'test_path_access':
nagios.c:122:3: warning: ignoring return value of 'asprintf', declared with attribute warn_unused_result [-Wunus
d-result]
  122 |   asprintf(&path, "%s/%s", p, program);
```

Step 14:Add groups:
sudo useradd nagios
sudo groupadd nagcmd
sudo usermod -aG nagcmd nagios
sudo usermod -aG nagcmd www-data

```
msd201104@sahil:/tmp/nagios-4.4.6$ ^C
msd201104@sahil:/tmp/nagios-4.4.6$ sudo useradd nagios
msd201104@sahil:/tmp/nagios-4.4.6$ sudo groupadd nagcmd
msd201104@sahil:/tmp/nagios-4.4.6$ sudo usermod -aG nagcmd nagios
msd201104@sahil:/tmp/nagios-4.4.6$ sudo usermod -aG nagcmd www-data
msd201104@sahil:/tmp/nagios-4.4.6$
```

Step 15:Perform rest steps for installing nagios:
sudo make install
sudo make install-init
sudo make install-commandmode
sudo make install-config
sudo make install-webconf

```
msd201104@sahil:/tmp/nagios-4.4.6$ sudo make install
cd ./base && make install
make[1]: Entering directory '/tmp/nagios-4.4.6/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 nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/tmp/nagios-4.4.6/base'
cd ./cgi && make install
make[1]: Entering directory '/tmp/nagios-4.4.6/cgi'
make install-basic
make[2]: Entering directory '/tmp/nagios-4.4.6/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
```

```
msd201104@sahil:/tmp/nagios-4.4.6$ sudo make install-init
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
msd201104@sahil:/tmp/nagios-4.4.6$ sudo make install-commandmode
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

msd201104@sahil:/tmp/nagios-4.4.6$ sudo make install-config
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/e
```

```
msd201104@sahil:/tmp/nagios-4.4.6$ sudo make install-webconf
/usr/bin/install -c -m 644 sample-config/httpd.conf /etc/apache2/sites-enabled/nagios.conf
if [ 0 -eq 1 ]; then \
        ln -s /etc/apache2/sites-enabled/nagios.conf /etc/apache2/sites-enabled/nagios.conf; \
fi

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

msd201104@sahil:/tmp/nagios-4.4.6$
```

Step 16:
Create a Nagios admin user
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

```
msd201104@sahil:/tmp/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
msd201104@sahil:/tmp/nagios-4.4.6$
```

Step 17:restart apache
sudo systemctl restart apache2

```
msd201104@sahil:/tmp/nagios-4.4.6$ sudo systemctl restart apache2
msd201104@sahil:/tmp/nagios-4.4.6$
```

Step 18:
Install the Nagios plugins:
cd /tmp
wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz
tar -xzf nagios-plugins-2.3.3.tar.gz
cd nagios-plugins-2.3.3
./configure
make
sudo make install

```
msd201104@sahil:/tmp/nagios-4.4.6$ cd /tmp
msd201104@sahil:/tmp$ wget https://nagios-plugins.org/download/nagios-plugins-2.3.3.tar.gz
--2024-10-20 15:29:08--  https://nagios-plugins.org/download/nagios-plugins-2.3.3.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: 2782610 (2.7M) [application/x-gzip]
Saving to: 'nagios-plugins-2.3.3.tar.gz'

nagios-plugins-2.3.3.tar.gz  100%[=========================================>]   2.65M  15.1MB/s    in 0.2s

2024-10-20 15:29:08 (15.1 MB/s) - 'nagios-plugins-2.3.3.tar.gz' saved [2782610/2782610]

msd201104@sahil:/tmp$ tar -xzf nagios-plugins-2.3.3.tar.gz
msd201104@sahil:/tmp$ cd nagios-plugins-2.3.3
msd201104@sahil:/tmp/nagios-plugins-2.3.3$ ./configure
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
```

```
msd201104@sahil:/tmp/nagios-plugins-2.3.3$ make
make  all-recursive
make[1]: Entering directory '/tmp/nagios-plugins-2.3.3'
Making all in gl
make[2]: Entering directory '/tmp/nagios-plugins-2.3.3/gl'
rm -f alloca.h-t alloca.h && \
{ echo '/* DO NOT EDIT! GENERATED AUTOMATICALLY! */'; \
  cat ./alloca.in.h; \
} > alloca.h-t && \
mv -f alloca.h-t alloca.h
```

```
msd201104@sahil:/tmp/nagios-plugins-2.3.3$ sudo make install
Making install in gl
make[1]: Entering directory '/tmp/nagios-plugins-2.3.3/gl'
make  install-recursive
make[2]: Entering directory '/tmp/nagios-plugins-2.3.3/gl'
make[3]: Entering directory '/tmp/nagios-plugins-2.3.3/gl'
make[4]: Entering directory '/tmp/nagios-plugins-2.3.3/gl'
if test yes = no; then \
  case 'linux-gnu' in \
    darwin[56]*) \
```

```
msd201104@sahil:/tmp/nagios-plugins-2.3.3$ sudo mkdir -p /usr/local/nagios/etc/servers
msd201104@sahil:/tmp/nagios-plugins-2.3.3$ sudo nano /usr/local/nagios/etc/servers/nginx.cfg
msd201104@sahil:/tmp/nagios-plugins-2.3.3$
```

Step 19:Configure and make new file:

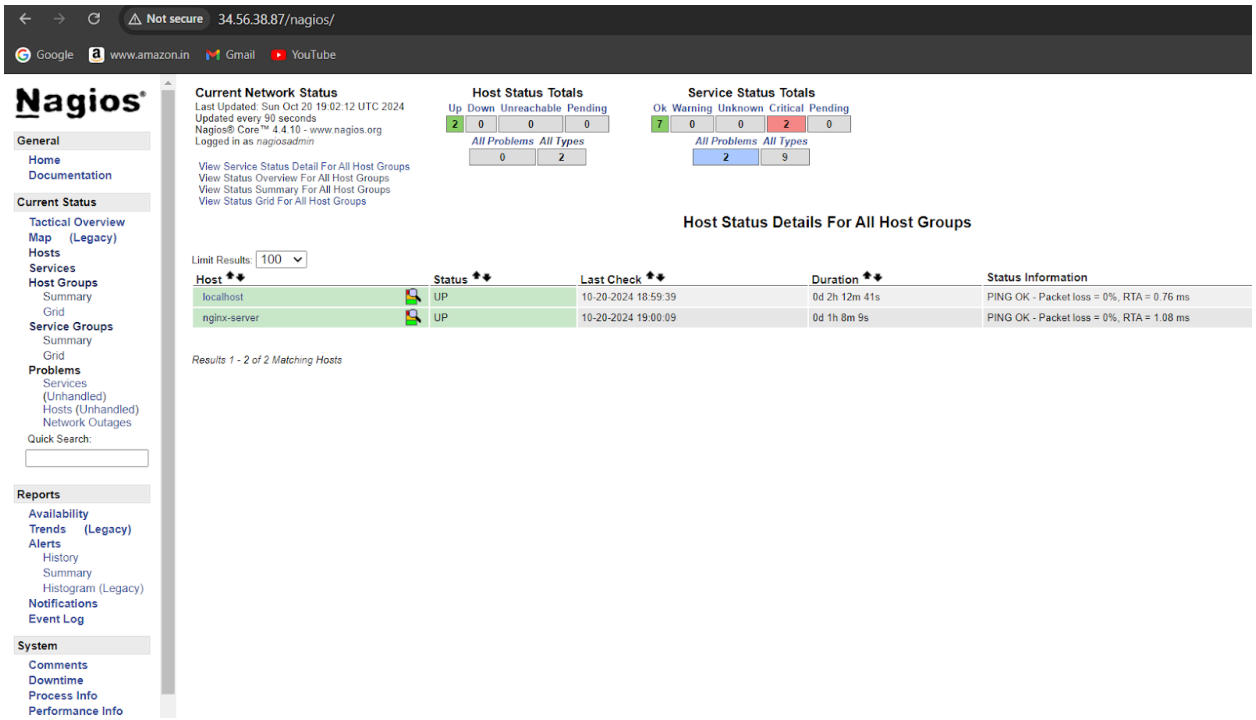
```
define host {
 use                linux-server
 host_name          nginx-server
 address            <EXTERNAL_IP_OF_NGINX>//your ip addressof nignix
 max_check_attempts     5
 check_period         24x7
 notification_interval   30
 notification_period    24x7
}

define service {
 use                generic-service
 host_name          nginx-server
 service_description    HTTP
 check_command        check_http
 notifications_enabled   1
}
```

```
  GNU nano 4.8                          /usr/local/nagi
define host {
  use                     linux-server
  host_name               nginx-server
  address                 35.193.219.222
  max_check_attempts      5
  check_period            24x7
  notification_interval   30
  notification_period     24x7
}

define service {
  use                     generic-service
  host_name               nginx-server
  service_description     HTTP
  check_command           check_http
  notifications_enabled   1
}
```

Step 20 Add this line
cfg_file=/usr/local/nagios/etc/objects/nginx.cfg
In
sudo nano /usr/local/nagios/etc/nagios.cfg

```
  GNU nano 4.8            /usr/local/nagios/etc/nagios.cfg


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

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

# Definitions for monitoring 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
```

Step 21:Now you will be able to monitor pods:



5.Conclusion:

The deployment of a simple Nginx server on a Kubernetes cluster using Google Cloud Console was successfully executed, replacing the previously planned AWS Cloud9 environment due to its deprecation. The process involved creating a Kubernetes cluster, deploying the Nginx application, and subsequently configuring Nagios for health monitoring.

The tasks performed included:

1. **Nginx Deployment**: An Nginx server was effectively deployed within the Kubernetes cluster, demonstrating the platform's capability to manage containerized applications seamlessly.
2. **Nagios Configuration**: Nagios was installed and configured to monitor the availability of the Nginx pod. This involved setting up the Nagios monitoring server and defining checks to determine the operational status of the Nginx application.
3. **Health Monitoring Verification**: The monitoring system was tested, successfully detecting the state of the Nginx pod. Nagios was able to provide real-time alerts regarding the pod's availability, confirming that the monitoring setup was effective and operational.

This project not only showcased the effectiveness of Google Cloud Console for Kubernetes deployments but also highlighted the importance of robust monitoring solutions like Nagios in maintaining the health and availability of cloud-based applications. Overall, the successful implementation of these technologies provides a strong foundation for further exploration into cloud-native application development and monitoring.