Name:Nikita Thadani Roll No:55 Div:D15C

Case Study: Kubernetes Application with Basic Monitoring

Objective: Set up a Kubernetes-based application with basic monitoring.

Key Technologies: Kubernetes, Google Cloud Console, and Nagios.

Problem Statement: "Deploy a basic application (such as an Nginx server) on a Kubernetes cluster using Google Cloud Console, and monitor its status with Nagios."

Tasks:

- Deploy the Nginx server on a Kubernetes cluster using Google Cloud Console.
- Install and configure Nagios to monitor the Nginx pod's status.
- Ensure that Nagios can detect the Nginx pod’s running status and notify when it is unavailable.

Note:

Due to the discontinuation of Kubernetes support in AWS Cloud9, this experiment uses Google Cloud Console, which offers a more robust platform for Kubernetes deployments and easier integration with monitoring tools like Nagios.

1. Introduction

In modern cloud computing, container orchestration is essential for ensuring scalability, availability, and fault tolerance. This case study examines the process of deploying an Nginx server on a Kubernetes cluster while using Nagios to monitor the application's health, focusing on the broader implications for cloud-native infrastructure.

2. Theoretical Overview

2.1 Containerization and Orchestration

Containerization packages an application and its dependencies into a lightweight, isolated environment, allowing consistent performance across various stages—development, testing, and production. Kubernetes, an open-source orchestration platform, automates the deployment and scaling of containerized applications, ensuring optimal resource use and high reliability.

Key Concepts:

- Containers: Isolated environments for applications.
- Orchestration: Managing multiple containers to ensure smooth operation across distributed systems.

- Microservices Architecture: An approach where applications are divided into loosely coupled services, improving modularity and scalability.

2.2 Monitoring in Distributed Systems

Monitoring plays a vital role in maintaining application performance. In distributed environments like Kubernetes, traditional monitoring methods may be insufficient due to the dynamic nature of containerized applications. Nagios, a popular open-source tool, is capable of tracking application health, availability, and performance through customizable checks and alerts**.**

Key Concepts:

- Health Checks: Regular assessments of application and service status.
- Alerts: Notifications triggered by specific conditions indicating potential issues.
- Service-Level Objectives (SLOs): Metrics that define expected performance and reliability levels.

3. Methodology

3.1 Environment Setup

1. Cloud Provider Choice:
    - Due to the discontinuation of AWS Cloud9 for Kubernetes, Google Cloud Console was chosen for deploying and managing the Kubernetes cluster.
2. Kubernetes Cluster Creation:
    - A Kubernetes cluster was created in Google Cloud Console, following recommended practices for setup and management.

3.2 Application Deployment

1. Nginx Deployment:
    - The Nginx server was deployed using Kubernetes deployment manifests, which specify the application's desired state, including replicas, container images, and service configuration.
2. Service Exposure:
    - The Nginx server was exposed via a LoadBalancer service, allowing external access to the application.

3.3 Monitoring Setup

1. Nagios Installation:

- Nagios was installed on a separate virtual machine, adhering to its installation and configuration documentation.

2. Monitoring Configuration:

- Nagios was set up to monitor the Nginx server by defining checks that assess the application's availability and performance. Custom commands were used to run HTTP-based health checks.

3. **Alerting Mechanism:**

- Nagios was configured to send email alerts when the Nginx server became unreachable, ensuring timely notifications for the operations team.

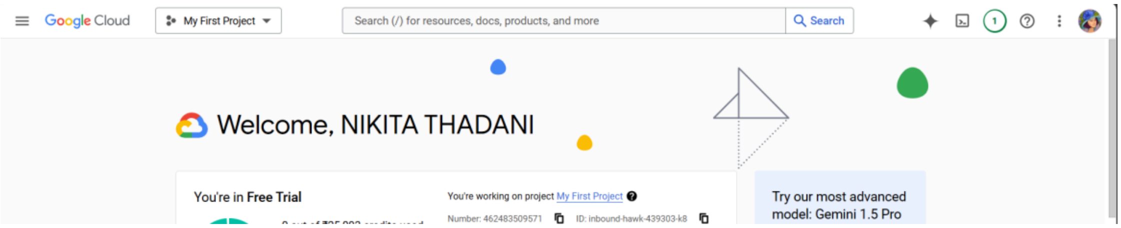
4.Steps performed:

Step 1:Make a account on Google cloud console:

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

After making account,

Screen will be as shown below:



Welcome, NIKITA THADANI

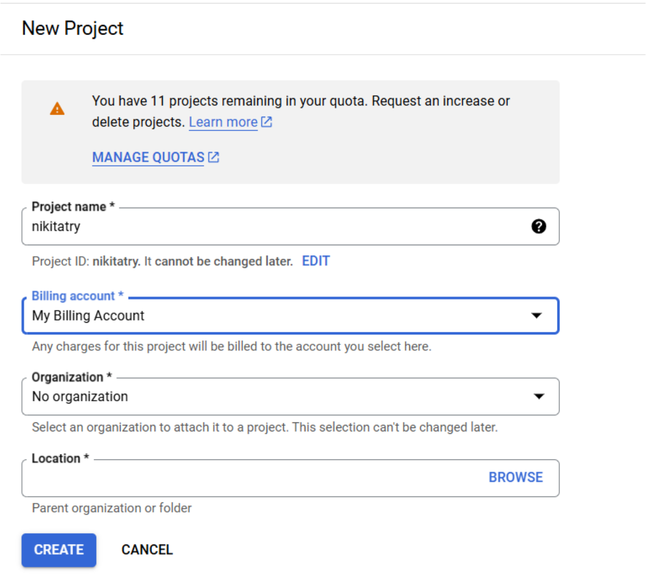
You're in Free Trial

You're working on project My First Project

Number: 462483509571 ID: inbound-hawk-439303-k8

Try our most advanced model: Gemini 1.5 Pro

Step2:Now you need to make a new project:



New Project

You have 11 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

Project name * nikitatry

Project ID: nikitatry. It cannot be changed later. EDIT

Billing account * My Billing Account

Any charges for this project will be billed to the account you select here.

Organization * No organization

Select an organization to attach it to a project. This selection can't be changed later.

Location * BROWSE

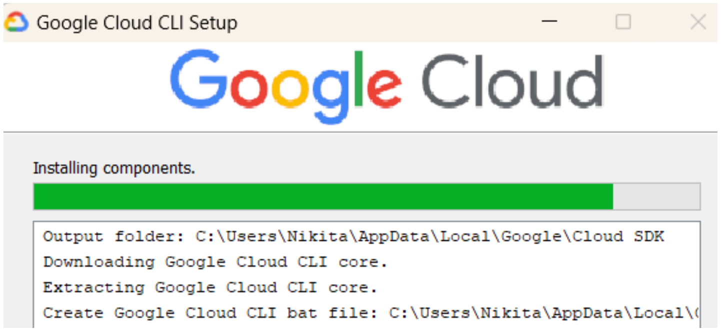
Parent organization or folder

Step 3:Install Google cloud sdk

https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe

From this link

Name:Nikita Thadani Roll No:55 Div:D15C

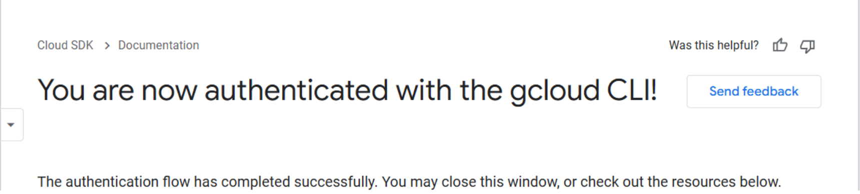


Installing components.

```
Output folder: C:\Users\Nikita\AppData\Local\Google\Cloud SDK
Downloading Google Cloud CLI core.
Extracting Google Cloud CLI core.
Create Google Cloud CLI bat file: C:\Users\Nikita\AppData\Local\(
```

Step 4:

Authenticate your account



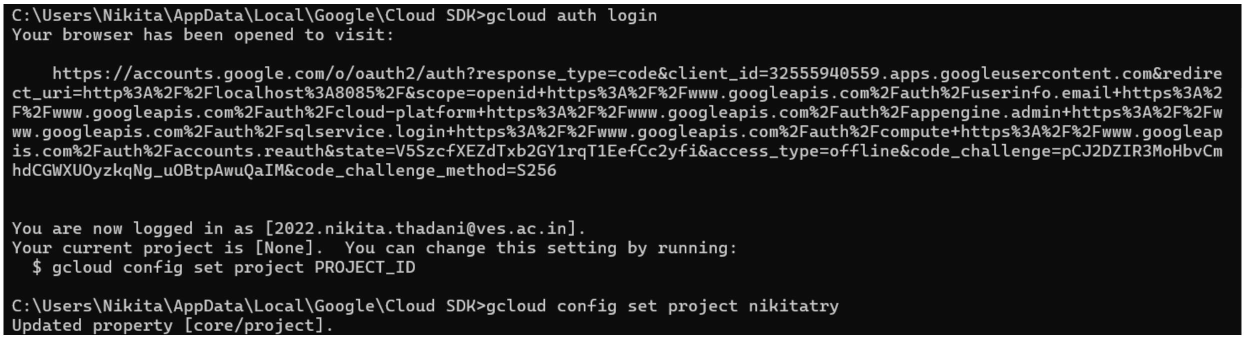
You are now authenticated with the gcloud CLI!

The authentication flow has completed successfully. You may close this window, or check out the resources below.

gcloud auth login

Set your GCP project

gcloud config set project <PROJECT_ID>



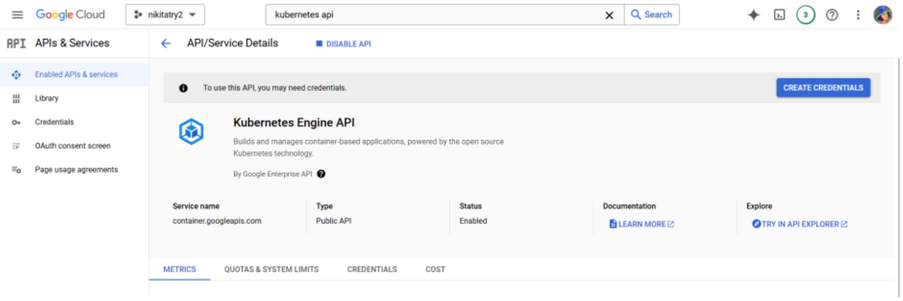
```
C:\Users\Nikita\AppData\Local\Google\Cloud SDK>gcloud auth login
Your browser has been opened to visit:

    https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redire
ct_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2
F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fw
ww.googleapis.com%2Fauth%2Fsqlservice.login+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleap
is.com%2Fauth%2Faccounts.reauth&state=V5SzcfXEZdTxb2GY1rqT1EefCc2yfi&access_type=offline&code_challenge=pCJ2DZIR3MoHbvCm
hdCGWXUOyzkqNg_uOBtpAwuQaIM&code_challenge_method=S256


You are now logged in as [2022.nikita.thadani@ves.ac.in].
Your current project is [None].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID

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

Step 5:Enable Kubernetes API:



To use this API, you may need credentials.

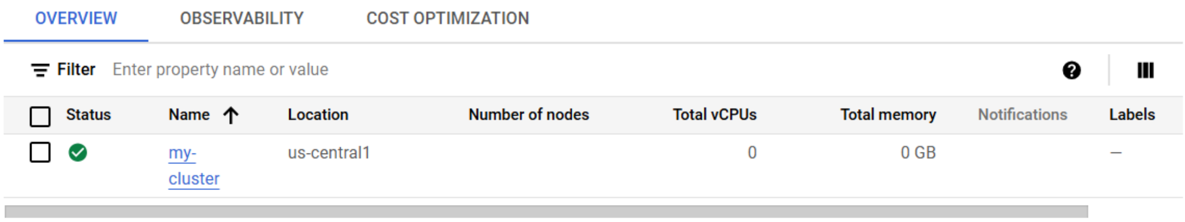
Kubernetes Engine API

Builds and manages container-based applications, powered by the open source Kubernetes technology.

By Google Enterprise API

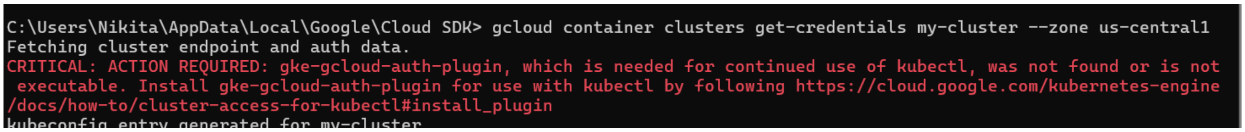
Service name	Type	Status	Documentation	Explore
container.googleapis.com	Public API	Enabled	LEARN MORE	TRY IN API EXPLORER

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



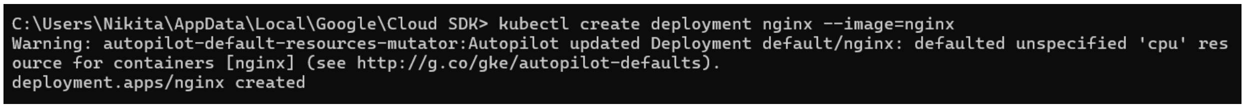
□	Status	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
□	✓	my-cluster	us-central1		0	0 GB		—

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\Nikita\AppData\Local\Google\Cloud SDK> gcloud container clusters get-credentials my-cluster --zone us-central1
Fetching cluster endpoint and auth data.
CRITICAL: ACTION REQUIRED: gke-gcloud-auth-plugin, which is needed for continued use of kubectl, was not found or is not
 executable. Install gke-gcloud-auth-plugin for use with kubectl by following https://cloud.google.com/kubernetes-engine
/docs/how-to/cluster-access-for-kubectl#install_plugin
kubeconfig entry generated for my-cluster
```

Step 7:
Create Nginx Deployment
Use kubectl to deploy an Nginx server : kubectl create deployment nginx --image=nginx

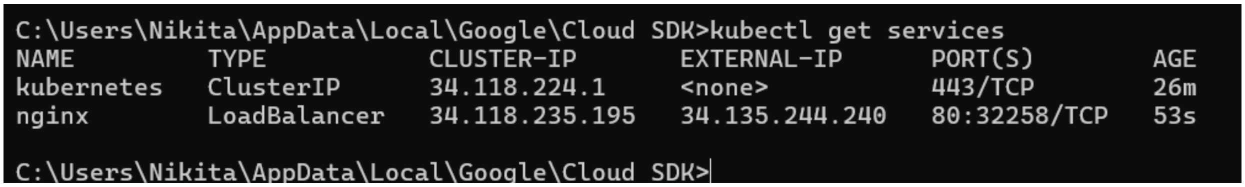


```
C:\Users\Nikita\AppData\Local\Google\Cloud SDK> kubectl create deployment nginx --image=nginx
Warning: autopilot-default-resources-mutator:Autopilot updated Deployment default/nginx: defaulted unspecified 'cpu' res
ource for containers [nginx] (see http://g.co/gke/autopilot-defaults).
deployment.apps/nginx created
```

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

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

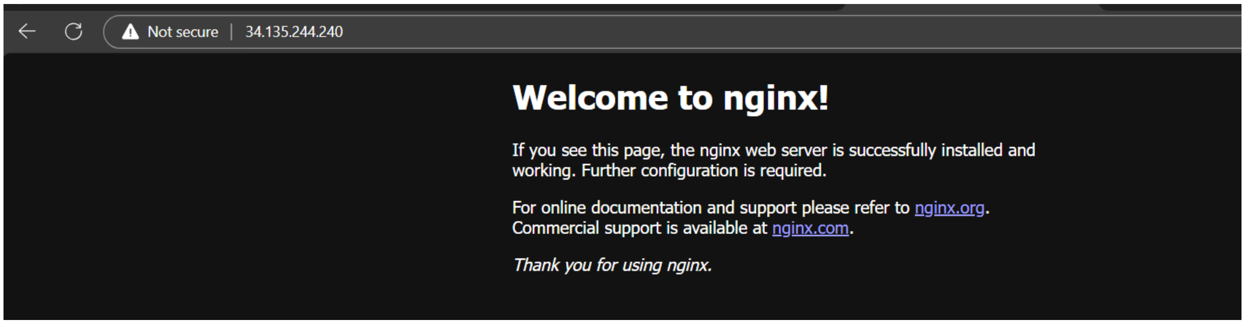
Step 9:
To get the external IP address of the service:
kubectl get services



```
C:\Users\Nikita\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        26m
nginx        LoadBalancer   34.118.235.195   34.135.244.240   80:32258/TCP   53s

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

Once the external IP is available, you can access the Nginx server at http://34.135.244.240



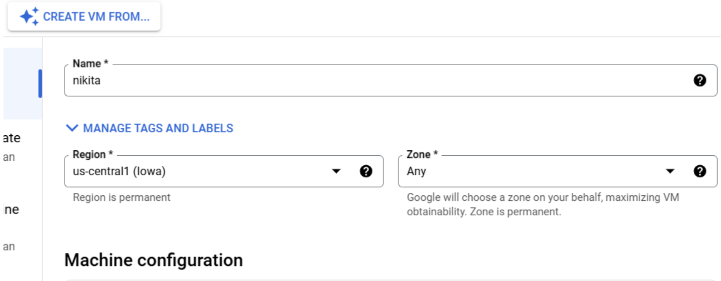
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Step10: Create a VM Instance:



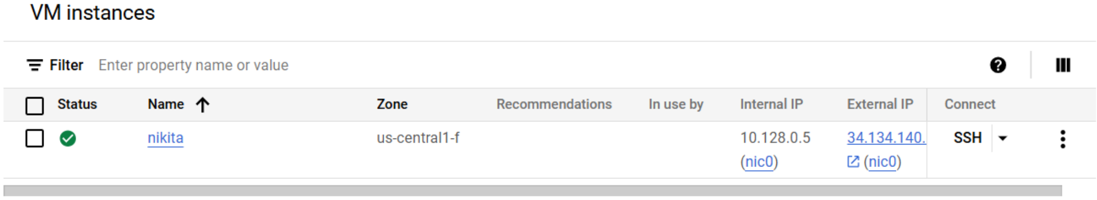
Name * nikita

Region * us-central1 (Iowa)
Region is permanent

Zone * Any
Google will choose a zone on your behalf, maximizing VM obtainability. Zone is permanent.

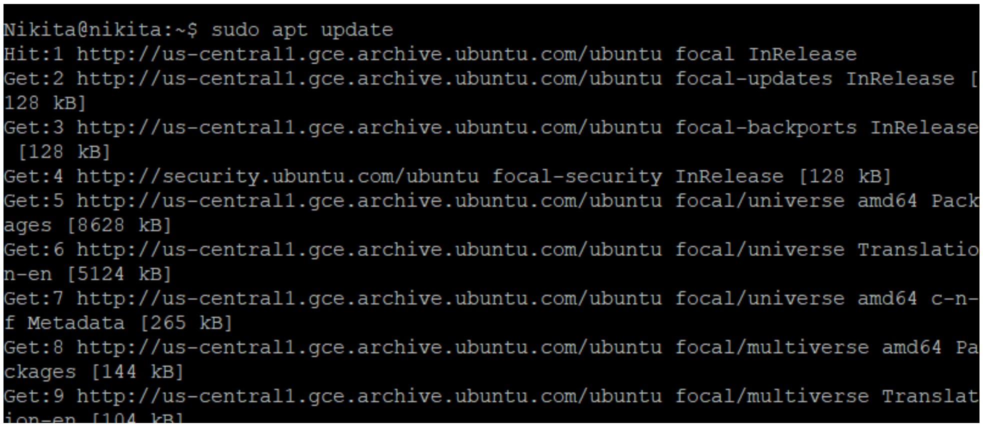
Machine configuration

Do these configurations;
Allow HTTP
Allow HTTPS
Your instance will get created



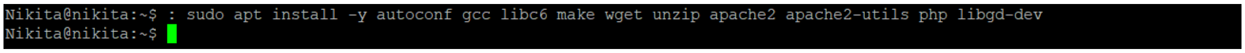
☐	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
☐	✓	nikita	us-central1-f			10.128.0.5 (nic0)	34.134.140. (nic0)	SSH ▾

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



```
Nikita@nikita:~$ 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 Pack
ages [8628 kB]
Get:6 http://us-central1.gce.archive.ubuntu.com/ubuntu focal/universe Translatio
n-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 Pa
ckages [144 kB]
Get:9 http://us-central1.gce.archive.ubuntu.com/ubuntu focal/multiverse Translat
ion-en [104 kB]
```

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



```
Nikita@nikita:~$ : sudo apt install -y autoconf gcc libc6 make wget unzip apache2 apache2-utils php libgd-dev
Nikita@nikita:~$ 
```

Step 13:Download and install Nagios

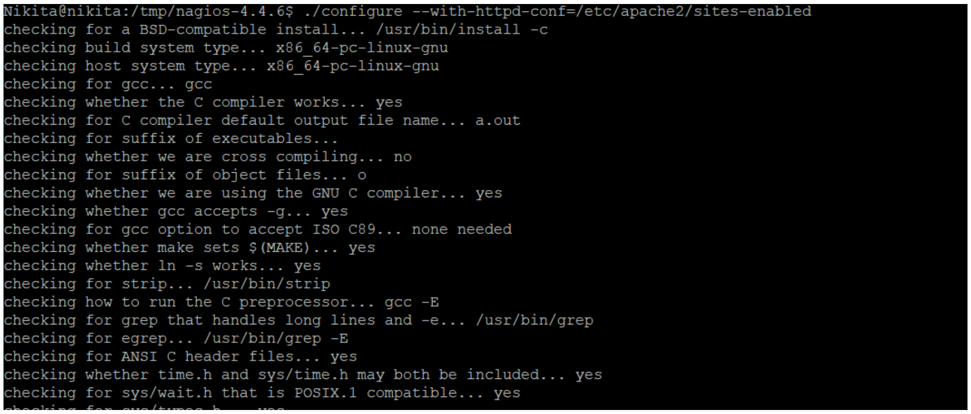
cd /tmp

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

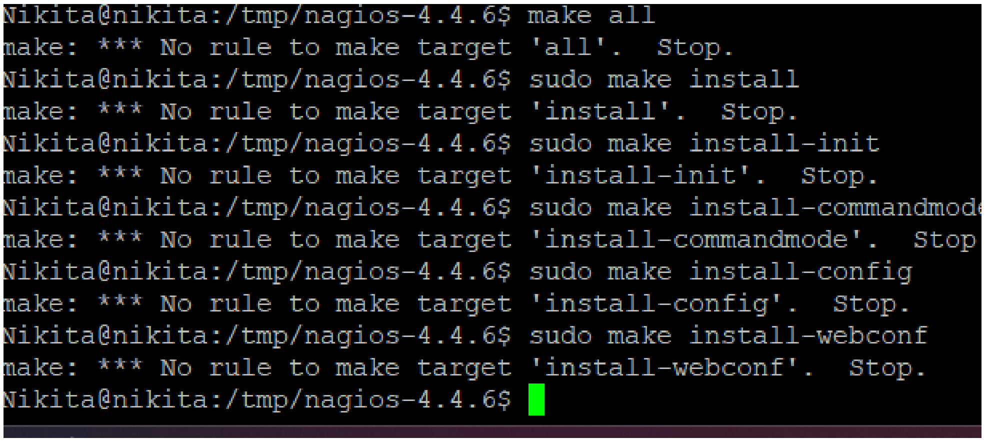
tar -xzf nagios-4.4.6.tar.gz cd nagios-4.4.6

./con igure --with-httpd-conf=/etc/apache2/sites-enabled

make all

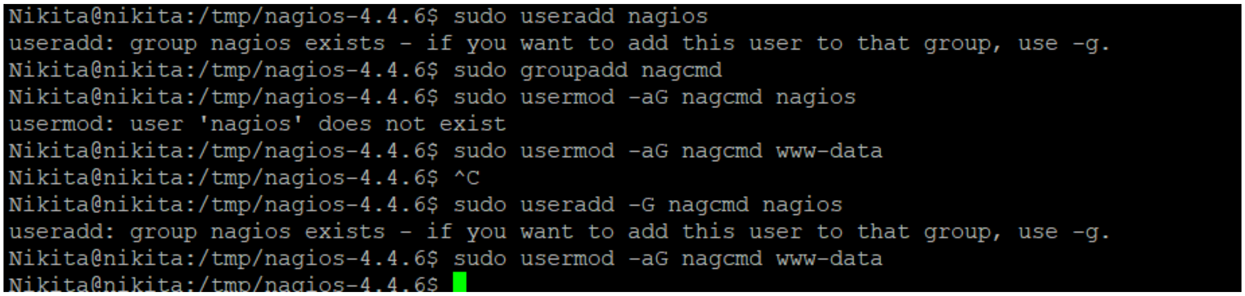


```
Nikita@nikita:/tmp/nagios-4.4.6$ ./configure --with-httpd-conf=/etc/apache2/sites-enabled
checking for a BSD-compatible install... /usr/bin/install -c
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C 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
```



```
Nikita@nikita:/tmp/nagios-4.4.6$ make all
make: *** No rule to make target 'all'.  Stop.
Nikita@nikita:/tmp/nagios-4.4.6$ sudo make install
make: *** No rule to make target 'install'.  Stop.
Nikita@nikita:/tmp/nagios-4.4.6$ sudo make install-init
make: *** No rule to make target 'install-init'.  Stop.
Nikita@nikita:/tmp/nagios-4.4.6$ sudo make install-commandmode
make: *** No rule to make target 'install-commandmode'.  Stop
Nikita@nikita:/tmp/nagios-4.4.6$ sudo make install-config
make: *** No rule to make target 'install-config'.  Stop.
Nikita@nikita:/tmp/nagios-4.4.6$ sudo make install-webconf
make: *** No rule to make target 'install-webconf'.  Stop.
Nikita@nikita:/tmp/nagios-4.4.6$ 
```

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

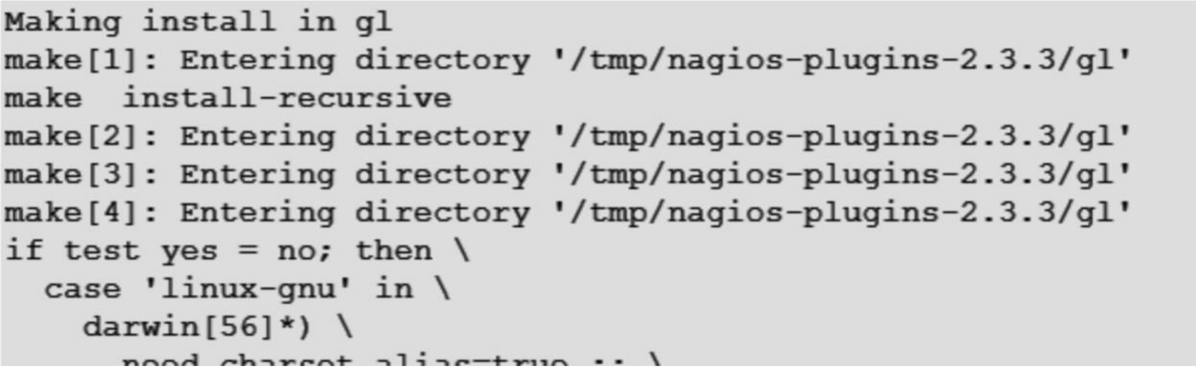


```
Nikita@nikita:/tmp/nagios-4.4.6$ sudo useradd nagios
useradd: group nagios exists - if you want to add this user to that group, use -g.
Nikita@nikita:/tmp/nagios-4.4.6$ sudo groupadd nagcmd
Nikita@nikita:/tmp/nagios-4.4.6$ sudo usermod -aG nagcmd nagios
usermod: user 'nagios' does not exist
Nikita@nikita:/tmp/nagios-4.4.6$ sudo usermod -aG nagcmd www-data
Nikita@nikita:/tmp/nagios-4.4.6$ ^C
Nikita@nikita:/tmp/nagios-4.4.6$ sudo useradd -G nagcmd nagios
useradd: group nagios exists - if you want to add this user to that group, use -g.
Nikita@nikita:/tmp/nagios-4.4.6$ sudo usermod -aG nagcmd www-data
Nikita@nikita:/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

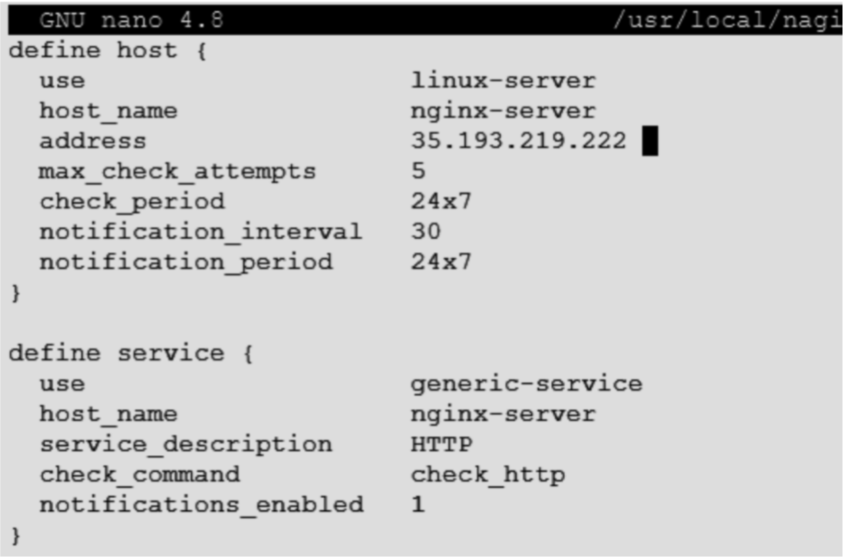
Step 16:
Create a Nagios admin user
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin

Step 17:
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



```
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]*) \
      need_charset_alias=true ;; \
```

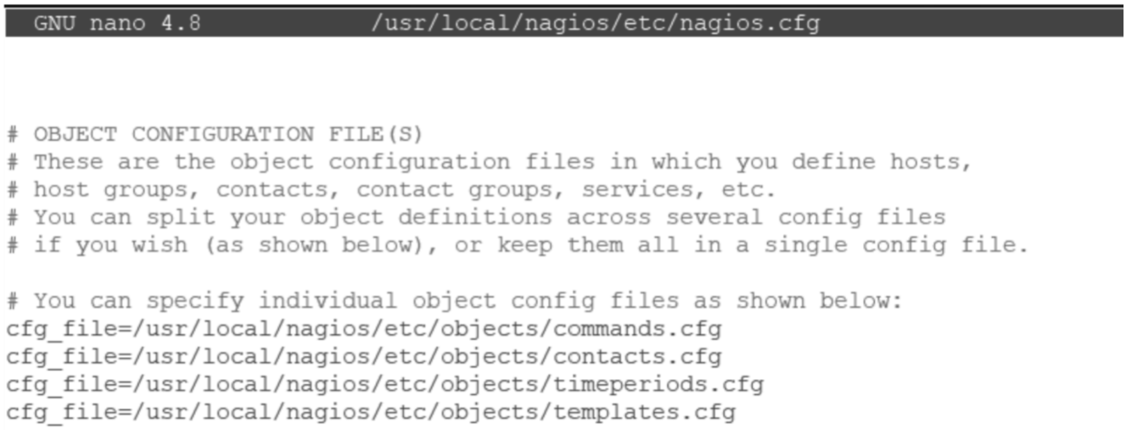
Step 19:Configure and make new file:



```
  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_ ile=/usr/local/nagios/etc/objects/nginx.cfg
In
sudo nano /usr/local/nagios/etc/nagios.cfg

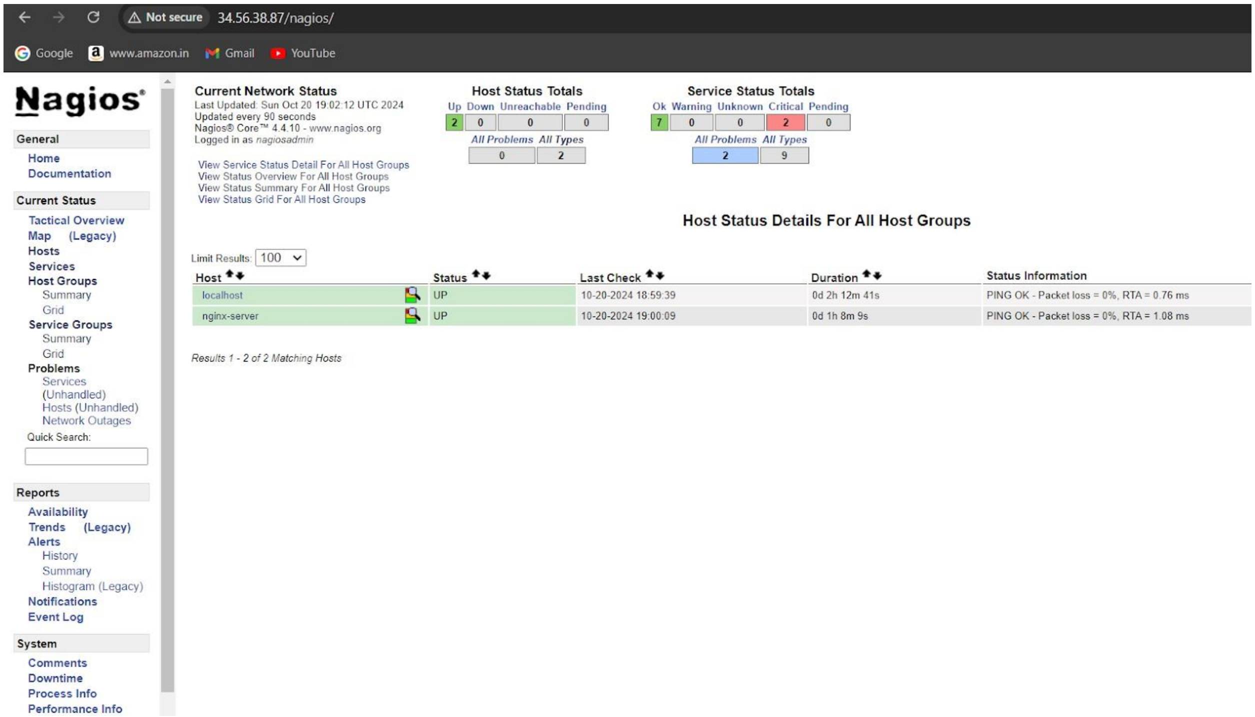


```
  GNU nano 4.8            /usr/local/nagios/etc/nagios.cfg


# 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
```

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



Current Network Status
Last Updated: Sun Oct 20 19:02:12 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.4.10 - www.nagios.org
Logged in as *nagiosadmin*

Host Status Totals

Up	Down	Unreachable	Pending
2	0	0	0

All Problems	*All Types*
0	2

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
7	0	0	2	0

All Problems	*All Types*
2	9

Host Status Details For All Host Groups

Host	Status	Last Check	Duration	Status Information
localhost	UP	10-20-2024 18:59:39	0d 2h 12m 41s	PING OK - Packet loss = 0%, RTA = 0.76 ms
nginx-server	UP	10-20-2024 19:00:09	0d 1h 8m 9s	PING OK - Packet loss = 0%, RTA = 1.08 ms

Results 1 - 2 of 2 Matching Hosts

5.Conclusion:

The deployment of a basic Nginx server on a Kubernetes cluster, utilizing Google Cloud Console, was carried out successfully. This replaced the originally intended AWS Cloud9 environment, which was no longer supported for Kubernetes-based deployments. The process involved setting up a Kubernetes cluster, deploying the Nginx application, and configuring Nagios for monitoring the application's health.

1. **Nginx Deployment:**
   An Nginx server was successfully deployed within the Kubernetes cluster, showcasing the platform's ability to efficiently manage containerized applications.

2. **Nagios Configuration:**
   Nagios was installed and configured to monitor the Nginx pod's availability.

3. **Health Monitoring Verification:**
   The monitoring system was thoroughly tested, with Nagios successfully detecting the Nginx pod's state.