

sãocarlos .monks

Case Técnico

DevOps

Ivan Costa de Sousa

São Paulo, 28 de Julho de 2023

Sumário

1	O Case	1
2	Objetivo	1
3	Elaboração do Case	1
3.1	Docker	1
3.2	Kubernetes	2
3.2.1	NameSpace	2
3.2.2	Deployment	2
3.2.3	Service	2
3.2.4	Horizontal Pod Autoscaler	2
3.2.5	Ingress	3
3.3	NGINX ingress controller	3
3.4	Google Cloud Platform (GCP)	3
3.5	Terraform	3
3.5.1	Provider e Version	4
3.5.2	Virtual Private Cloud (VPC)	4
3.5.3	Host	4
3.5.4	Cluster GKE	4
4	Como reproduzir o SETUP	5
4.1	Google Cloud Platform (GCP)	5
4.2	Terraform	5
4.3	Google Cloud Platform (GCP)	5
4.4	Deletar o projeto do cluster	7
5	Resultados Obtidos	7
5.1	Projeto Criado	7
5.2	Instâncias VMs	7
5.3	VPC e Route	8

5.4	Host	10
5.5	Cluster Kubernetes	11
5.6	Objetos Kubernetes	11
5.7	Acesso	12

6	Conclusão	13
----------	------------------	-----------

1 O Case

Imagine que você é um engenheiro de Infraestrutura trabalhando em uma empresa de tecnologia. Sua equipe está desenvolvendo uma nova aplicação que será hospedada no Google Cloud Platform (GCP). A aplicação em questão precisa ser escalável e a segurança é um fator crítico, então, foi decidido que a infraestrutura será baseada em um Cluster Kubernetes privado no GKE (Google Kubernetes Engine), e você será responsável por configurar essa infraestrutura.

Você deve criar um projeto no GCP, e, neste projeto, criar um Cluster de Kubernetes privado, operando em uma VPC (Virtual Private Cloud), com controle do tráfego de ingresso feito pelo NGINX Ingress Controller. Para validar a configuração, você deve hospedar uma aplicação de “Hello, World!” ou similar no Cluster criado e a disponibilizar para a internet através do NGINX Ingress Controller. Essa aplicação pode utilizar de alguma imagem disponível no DockerHub ou outro Registro de Contêineres com imagens públicas, bem como ser desenvolvida por você.

2 Objetivo

O objetivo deste projeto é construir, utilizando a ferramenta Terraform e recursos da Google Cloud Platform (GCP), um Cluster Kubernetes Privado (GKE), uma Virtual Private Cloud (VPC) e inserir uma aplicação simples, Hello Monks, inserida no repositório DockerHub, neste Cluster e expor essa aplicação de forma segura utilizando o NGINX ingress controller.

3 Elaboração do Case

Essa seção está distribuída conforme as ferramentas utilizadas no projeto.

Os arquivos utilizados nessa seção podem ser acessados no endereço GitHub: IvanC-Sousa/mediamonks.

3.1 Docker

Para a elaboração do Case foi escolhida a opção de construir a própria imagem do Docker, a aplicação Hello Monks, esta aplicação foi construída utilizando NodeJs.

Sendo assim, para a construção dessa imagem Docker, foi elaborado o DockerFile contendo as instruções para a construção da Imagem, instruções tais como a imagem base, o “repositório de trabalho” caso seja feito o SSH no container, a cópia dos arquivos da aplicação e os comandos para inicializar o NodeJs e a aplicação.

A imagem está disponível no repositório do DockerHub (Imagem: DockerHub/IvanCSousa), nomeada com a tag da seguinte forma **ivancsousa/mediamonks:0.12**, Note que no repositório existem as tags 0.1 e 0.12 mudando somente a porta exposta do container e da aplicação

Os documentos mencionados nessa seção estão localizados no repositório: GitHub/Docker.

3.2 Kubernetes

O código para a construção dos objetos Kubernetes está distribuído em dois arquivos YAML, um denominado de `monks.yaml` no qual contém os objetos Service, Deployment, Namespace e HorizontalPodAutoscaler (`monks.yaml`), e o arquivo nomeado de `ingress-monks.yaml` que contém o objeto Ingress que em conjunto com NGINX ingress controller será utilizado para expor a aplicação de forma segura. (`ingressmonks.yaml`).

Os arquivos mencionados nessa seção podem ser vistos no Github do projeto [GitHub/k8s](#).

Essa seção está dividida conforme os objetos Kubernetes utilizados no Cluster GKE.

3.2.1 Namespace

O Namespace, nomeado como `monks-ns`, é utilizado para criar um ambiente isolado para os objetos da aplicação. Os objetos descritos a seguir estão isolados pelo namespace `monks-ns` com exceção apenas para os objetos do NGINX ingress controller que estão no namespace default do cluster.

3.2.2 Deployment

O Deployment construído e nomeado como `deploy-monks` tem um estado desejado com duas réplicas de pods, `pod-monk`, utilizando uma imagem do repositório DockerHub, **ivanc-sousa/mediamonks:0.12**, como descrito na seção Docker. A porta utilizada pelo contêiner é a 3000, conforme construído na imagem.

3.2.3 Service

O objeto service construído e nomeado como `svc-monks`, foi configurado com o type `cluster-ip` para isolar melhor os objetos da aplicação, sendo que a exposição e o loadbalancear da aplicação é feita de forma segura pelo ingress e Nginx ingress Controller, as portas configuradas são port do Service como 8080 e a port target para pods como 3000.

3.2.4 Horizontal Pod Autoscaler

No objeto HorizontalPodAutoscaler e nomeado como `monks-hpa` é utilizado para escalar a aplicação e foi construído com base em uma métrica comum e nativa do uso da CPU dos contêineres, a métrica foi baseada pela escolha das máquinas utilizadas no Cluster, como será descrito nesse relatório na seção Terraform, e também pelo tipo de aplicação construída. Dessa forma, quanto maior o uso desse recurso mais pods devem ser executados para comportar e processar esse volume e uma redução de pods conforme a redução do uso do recurso. O autoscaling é baseado na CPU com o target de 80% de uso, podendo escalar no máximo em dez o número de pods.

3.2.5 Ingress

O objeto ingress, nomeado como ingress, tem como objetivo aplicar regras para o tráfego de entrada da aplicação, baseado no host disponível pelo service do Nginx Ingress Controller, e fazer o tráfego de entrada para o service da aplicação, essa é a maneira mais segura de expor a aplicação em uma rede privada. Na construção do objeto é possível notar no arquivo YAML a utilização das annotations que o ingress é baseado em nginx e que não utiliza um redirect para serviços de certificações tais como cert-manager. O host utilizando para a conexão é ip externo do service do ingress controller, as regras são baseadas em path, que nesse caso é própria raiz do endereço, como a aplicação utiliza apenas um service facilita no momento dos testes.

3.3 NGINX ingress controller

O NGINX-ingress-controller tem como objetivo expor a aplicação para fora da rede privada, utilizando as regras estabelecidas pelo ingress da aplicação, encaminhando o tráfego das requisições para o service da aplicação. No NGINX-ingress-controller também é feito o balanceamento de carga (LoadBalancer) pelo seu service. A instalação foi feita utilizando o Helm, por esse motivo não existem arquivos YAML referentes aos seus objetos Kubernetes, foi utilizado o repositório do ingress (<https://kubernetes.github.io/ingress-nginx>), sendo assim foi implantado e executando o Chart e criado os objetos do repositório no cluster, sendo o namespace default utilizado para isso.

3.4 Google Cloud Platform (GCP)

O projeto foi nomeado como MonksProject, utilizando o usuário ivansousa.mediamonks. No projeto foram habilitadas as APIs: Computer Engine, Kubernetes Engine e Identity-Aware Proxy (IAP), nas quais foram utilizadas respectivamente para as construções das Vms, construção do cluster Kubernetes e utilização de serviços de tunelamento. Depois desse processo foi criada uma service account para a utilização do Terraform, chamada sa-terraform, com o seguinte registro sa-terraform@monkscluster.iam.gserviceaccount.com que será utilizada para a construções dos recursos GCP, foi adicionada a esse service account a rule de Project Edit.

Com essas configurações descritas acima foi possível construir uma rede VPC privada com suas regras, uma VM host para a conexão no cluster GKE e um Cluster Kubernetes privado utilizando o Terraform.

3.5 Terraform

O Terraform foi utilizado para as construções dos recursos GCP que são necessários para a solução do Case, se tratando de uma ferramenta IaC amplamente utilizada no mercado. Os arquivos de códigos em Terraforms foram separados e construídos conforme as implementações dos recursos dentro do GCP, essa seção está dividida entre os recursos construídos e os arquivos Terraforms, sendo arquivos de código Terraform separados em: vpc, version, route, provider, cluster, host e os arquivos de Outputs e terraform.tfvars de variáveis.

Os arquivos descritos nessa sessão estão disponíveis no GitHub do projeto: [media-monks/Terraform](#).

3.5.1 Provider e Version

Esses dois arquivos de código Versions.tf e Provider.tf estão associados aos recursos e fonte de dados Terraform que foram utilizados, acionando os recursos referentes ao provedor, no caso deste Case o provider é o GCP, também estão presentes no arquivo informações da credencial e o nome do projeto utilizado para a construção dos recursos GCP.

3.5.2 Virtual Private Cloud (VPC)

Os arquivos de código IaC associados a VPC são: vpc.tf e route.tf, o primeiro está associado a criação da vpc em si, já o segundo está associado as configurações de rota e configuração a rede NAT que será utilizada para comunicação externa do Cluster.

Para a construção da VPC (Virtual Private Cloud) foram utilizados os recursos terraform `google_compute_network`, `google_compute_subnetwork`, e `google_compute_firewall` para configurar uma rede privada, ou seja, somente conexões dentro do próprio cluster são permitidas, com uma subrede configurada e com regras de Firewall para conexões internas. Dentro da definição da subnet é definido o range de ips para o Cluster.

3.5.3 Host

Esse arquivo de código foi utilizado para a construção de uma VM (Host) que foi utilizada como forma de conexão ao cluster, e o arquivo de código é o host.tf, por se tratar de uma cluster privado conexões externas não são permitidas, para isso é necessário construir uma conexão de uma máquina host que será atribuída ao firewall e ao cluster como uma conexão segura. Os recursos Terraform utilizados são: `google_compute_address` `google_compute_instance`, no primeiro recurso é definido o IP, tipo de endereço, região, rede e sub-rede, esse recurso é utilizado para a construção do host, o segundo recurso está associado a VM em si, onde atributos como tipo de máquina, tipo de disco e a interface de rede utilizada. É importante frisar que o host precisa estar na mesma região e vpc do Cluster Kubernetes.

3.5.4 Cluster GKE

O arquivo de código Terraform associados a construção do Cluster Kubernetes é o cluster.tf, os recursos Terraform `google_container_cluster` e `google_container_node_pool` foram utilizados na criação do Cluster, o primeiro recurso é referente a construção do Cluster em si, então nesse bloco são definidos atributos tais como a localização em que o cluster será construído, a mesma localização definida na VPC, a rede e a subrede em que o cluster está alocado que também foi configurada na vpc, o número inicial de nós sendo um, esses nós são privados e sem endpoint externo de conexão com o servidor, o range de ips alocados que foram definidos antes na criação da VPC e outros atributos não listados, já o segundo recurso está associado ao pool de nodes, no caso desse projeto é definido somente um pool de nodes, nesse

recurso é definido a que cluster o pool de nodes pertence, o mínimo e o máximo de nodes, o tipo de máquina utilizada, o tipo e o tamanho do disco, quantidade de nodes utilizado e outros recursos não listados, é importante salientar que nesse case foram utilizadas VM's preemptivas afim de diminuir custos.

4 Como reproduzir o SETUP

Essa seção tem como objetivo ser um guia de como reproduzir o projeto em um ambiente próprio, e será dividida em etapas, é recomendado que seja feito os downloads das pastas "K8s" e "Terraform" disponíveis no repositório, sendo necessário a utilização dos arquivos presentes nas pastas do projeto GitHub/mediamonks. Um pré-requisito é ter o Terraform e o Git instalados.

4.1 Google Cloud Platform (GCP)

1. Inicialmente é necessário criar um projeto com o nome escolhido e coletar o seu "Id". Doc
2. Para executar os próximos passos é necessário ativar as seguintes APIs: Kubernetes Engine(GKE), Compute Engine (CE) e Identity-Aware Proxy (IAP). Doc
3. Agora para acesso do Terraform ao projeto é necessário criar uma Service Account para o Terraform e aplica a role de Project Edit. Doc
4. Gere uma *Key.json* para a Service Account criada no item anterior e faça o download da mesma para a pasta Terraform Doc
5. Aplique a role de **iap.tunnel.user** no seu usuário Doc.

4.2 Terraform

O Git instalado é um pré-requisito para o Terraform possa acessar e fazer downloads dos módulos necessários. No diretório "Terraform":

1. Altere o id do projeto coletado anteriormente no arquivo de variáveis Terraform.tfvars.
2. No diretório Terraform execute os seguintes comandos:
\$ terraform init
\$ terraform plan
\$ terraform apply
obs.: Espere o processo terminar, isso pode demorar um pouco

4.3 Google Cloud Platform (GCP)

Para executar os próximos passos certifique-se que o processo anterior terminou sem erros.

1. Acesse o Host criado no recurso Compute Engine, como nomeado no arquivo host.tf
2. Utilize o Cloud Shell da VM Host
3. Baixe o kubectl na Instância VM
\$ sudo apt-get install kubectl
4. É necessário fazer a autenticação para isso utilize o comando
\$ gcloud auth login
5. Caso seja necessário instale o SDK de autenticação do GKE
\$ sudo apt-get install google-cloud-sdk-gke-gcloud-auth-plugin
6. O tunelamento/jump para o Cluster GKE pode ser feito
"ex: gcloud container clusters get-credentials monks-cluster --zone us-central1-c --project monksproject"

Ainda no Cloud Shell da instância VM e conectado ao cluster, execute os itens a seguir para a implementação dos objetos do NGINX ingress Controller:

1. Instale o HELM
\$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
\$ chmod 700 get_helm.sh
\$./get_helm.sh
2. Faça o download do repositório e aplique o Chart do Ingress Nginx Controller
\$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
\$ helm repo update
\$ helm install monks ingress-nginx/ingress-nginx
3. Certifique-se de que o Nginx ingress Controller foi instalado e anote o EXTERNAL-IP do service do ingress-nginx-controller
\$ kubectl get pod
\$ kubectl get svc monks-ingress-nginx-controller

Agora para aplicar os objetos Kubernetes da aplicação é necessário fazer o downloads dos arquivos monks.yaml e ingress_monks.yaml na VM Host e seguir os próximos passos:

1. Aplique os objetos Kubernetes do arquivo monks.yaml
\$ kubectl apply -f monks.yaml
2. Verifique o Status dos Objetos
\$ kubectl get all -n monks-ns
3. Para aplicar o Ingress é necessário mudar o Host no arquivo ingress_monks.yaml para o endereço EXTERNAL-IP copiado no item anterior
obs.: Utilize o nip.io - ex: "34.122.88.204.nip.io"
\$ kubectl apply -f ingress_monks.yaml

4. Verifique os status dos objetos
\$ kubectl get all -n monks-ns

Em seu Browser digite o endereço utilizado no Host - ex: "34.122.88.204.nip.io" e verifique o acesso.

4.4 Deletar o projeto do cluster

Caso queira excluir o projeto siga os próximos passos:

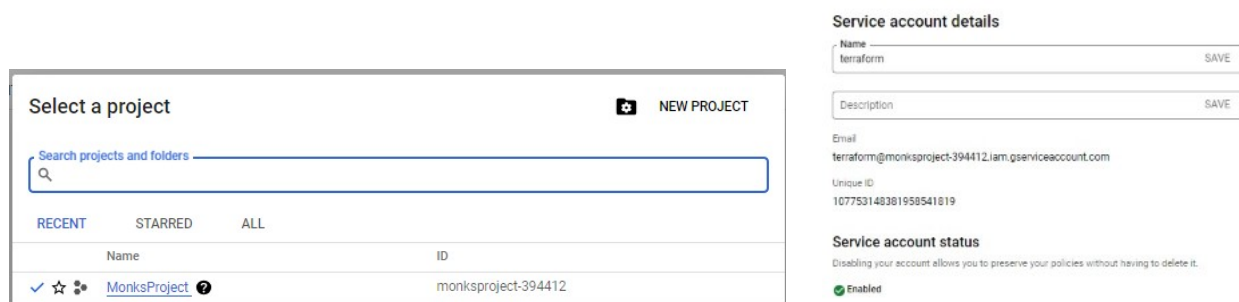
1. Na pasta do Terraform execute o seguinte comando
\$ terraform destroy
obs. Espere o processo terminar, isso pode demorar um pouco
2. No console do GCP entre em Faturamento e desative o projeto criado no campo "ações"
3. No console do GCP entre em IAM e Administração, Configurações e encerre o projeto.

5 Resultados Obtidos

5.1 Projeto Criado

A imagem 1 a seguir correspondem aos primeiros passos, ou seja, a construção do projeto e a Service Account Terraform criada.

Figura 1: Projeto e Service Account



5.2 Instâncias VMs

A imagem 2 a seguir corresponde as instâncias criadas no projeto, as duas primeiras com prefixo "gke" correspondem aos nodes configurados no arquivo de código Terraform, cluster.tf, nesse caso de ambiente teste foram configuradas duas instâncias na mesma zona, porém o ideal seria criar as instâncias em zonas diferentes da mesma localização para aumentar a resiliência do cluster, já a terceira corresponde ao VM Host que é utilizada para conexão com o Cluster Kubernetes privado. Pode ser observado na imagem o campo "Network" das instâncias que

estas pertencem a mesma VPC, monks-vpc, também é possível observar que todas estão na mesma zona, como visto no campo "Zone".

Figura 2: Instâncias VMs

VM instances											
Filter Enter property name or value											
<input type="checkbox"/>	<input type="checkbox"/>	Status	Name ↑	Zone	Creation time	Machine type	Preserved state size	Internal IP	External IP	Network	Connect
<input type="checkbox"/>	<input type="checkbox"/>	✓	gke-monks-cluster-monks-cluster-pool-27685795-020p	us-central1-c	Jul 30, 2023, 4:08:18 PM UTC-03:00	e2-small	0 GB	10.0.0.3 (nic0)		monks-vpc	gke-monks-cluster-c3722c14 SSH
<input type="checkbox"/>	<input type="checkbox"/>	✓	gke-monks-cluster-monks-cluster-pool-27685795-g669	us-central1-c	Jul 30, 2023, 4:08:18 PM UTC-03:00	e2-small	0 GB	10.0.0.4 (nic0)		monks-vpc	gke-monks-cluster-c3722c14 SSH
<input type="checkbox"/>	<input type="checkbox"/>	✓	monks-host	us-central1-c	Jul 30, 2023, 3:56:44 PM UTC-03:00	e2-small	0 GB	10.0.0.7 (nic0)		monks-vpc	SSH

No campo "Internal IP" verificamos que todas estão dentro do mesmo range, como configurado na subnet, host e cluster.

5.3 VPC e Route

As duas imagens 3 e 4 a seguir correspondem a Virtual Private Cloud (VPC) e a Subnet respectivamente, que foram criadas conforme o arquivo de código "vpc.tr", citados anteriormente. É possível observar na imagem 2 a subnet "monks-subnet".

Figura 3: VPC monks-vpc

monks-vpc									
Subnet creation mode									
Custom subnets									
Dynamic routing mode									
Regional									
VPC network ULA internal IPv6 range									
Disabled									
DNS server policy									
None									
Maximum transmission unit									
1460									
SUBNETS STATIC INTERNAL IP ADDRESSES FIREWALLS ROUTES VPC NETWORK PEERING PRIVATE SERVICE CONNECTION									
ADD SUBNET FLOW LOGS									
Private Google Access is in effect (even though it has not been enabled manually) when Cloud NAT is enabled for the primary IP range of the subnetwork. Learn more									
Filter Enter property name or value									
<input type="checkbox"/>	Name ↑	Region	Stack Type	Internal IP ranges	External IP ranges	Secondary IPv4 ranges	Gateway	Private Google Access	Flow logs
<input type="checkbox"/>	monks-subnet	us-central1	IPv4	10.0.0.0/24	None	10.12.0.0/21, 10.11.0.0/21	10.0.0.1	On	Off

Na imagem a seguir 4 a seguir é possível observar no campo "ip-range" o range associada a sub-rede utilizada pelo cluster, também é possível observar os range secundários a separação onde serão alocados os pods e o services, isso é importante para aumentar resiliência e monitoramento do cluster, por fim no campo de "Private Google Access" está "on", ou seja, corresponde as características de segurança podendo ser acessível somente por elementos internos.

Figura 4: Subnet monks-subnet

monks-subnet

VPC Network
[monks-vpc](#)

Region
us-central1

IP stack type
IPv4 (single-stack)

IP ranges

IP range	IP version	Access type
10.0.0.0/24	IPv4	Internal

Secondary IPv4 ranges ?

Subnet range name	Secondary IPv4 range
gke-monks-cluster-pods-c3722c14	10.11.0.0/21
gke-monks-cluster-services-c3722c14	10.12.0.0/21

Gateway
10.0.0.1

Private Google Access
On

Por fim nessa seção, a imagem 5 a seguir corresponde as configurações da route e NAT, utilizadas pelo cluster para conexões externa ao cluster.

Figura 5: Route - NAT

nat-router

Network [monks-vpc](#)

Region [us-central1](#)

Interconnect encryption Unencrypted

Google ASN

Advertised route configuration

BGP sessions will advertise these routes if no other configuration is specified

Advertisement mode
Custom

Advertise all available subnets
No

Advertised IP ranges

[SUBNETS](#) [CUSTOM IP RANGES](#)

This router does not advertise any subnets

Cloud NAT gateways

ADD CLOUD NAT GATEWAY

Gateway name	Status
cloud-nat-wyjkz	Running

[EQUIVALENT REST](#)

Esse passo é muito importante, pois com ele é possível utilizarmos imagens e repositórios que estão fora do cluster. Na imagem é possível observar o campo "Network" denotada como "monks-vpc" e a mesma "Region" da VCP.

5.4 Host

A seguir é possível observar na imagem 6, configurações utilizadas no host, tais como tipo de máquina utilizada, o disco e zona, conforme configurado no arquivo de código Terraform host.tf descrito anteriormente.

Figura 6: Host Instância

DETAILS

OBSERVABILITY

OS INFO

SCREENSHOT

Basic information

Name

monks-host

Instance Id

226137070713375875

Description

None

Type

Instance

Status

Running

Creation time

Jul 30, 2023, 3:56:44 PM UTC-03:00

Zone

us-central1-c

Instance template

None

In use by

None

Reservations

Automatically choose (default)

Labels

None

Tags

Deletion protection

Disabled

Confidential VM service

Disabled

Preserved state size

0 GB

Machine configuration

Machine type

e2-small

CPU platform

AMD Rome

Minimum CPU platform

None

Architecture

x86_64

vCPUs to core ratio

Custom visible cores

Display device

Disabled

GPUs

None

Networking

Public DNS PTR Record

None

Total egress bandwidth tier

NIC type

Firewalls

HTTP traffic

Off

monks-host

EDIT

RESET

CREATE MACHINE IMAGE

CREATE SIMILAR

START / RESUME

OPERATIONS

DETAILS

OBSERVABILITY

OS INFO

SCREENSHOT

Firewalls

HTTP traffic

Off

HTTPS traffic

Off

Allow Load Balancer Health checks

Off

Network tags

None

Network interfaces

Name	Network	Subnetwork	Primary internal IP address	Alias IP ranges	IP stack type	External IP address	Network
nic0	monks-vpc	monks-subnet	10.0.0.7		IPv4	None	-

Storage


Boot disk


Name	Image	Interface type	Size (GB)	Device name	Type	Architecture	Encryption	Mode	Wh
monks-host	debian-11-bullseye-v20230711	SCSI	10	persistent-disk-0	Standard persistent disk	x86_64	Google-managed	Boot, read/write	Del

5.5 Cluster Kubernetes

A seguir é possível observar na imagem 7, configurações utilizadas no Cluster, conforme configurado no arquivo de código Terraform "cluster.tf" descrito anteriormente.

Figura 7: Cluster Kubernetes Privado (GKE)

Cluster basics		
Name	monks-cluster	🔒
Location type	Zonal	🔒
Control plane zone	us-central1-c	🔒
Default node zones 	us-central1-c	✎
Release channel	Regular channel	✎ UPGRADE AVAILABLE
Version	1.27.2-gke.1200	
Total size	2	ⓘ
External endpoint	Disabled	🔒
Internal endpoint	10.13.0.2 Show cluster certificate	🔒

Networking		
Private cluster	Enabled	🔒
Default SNAT	Enabled	✎
Control plane address range	10.13.0.0/28	🔒
Control plane global access	Disabled	✎
VPC peering	gke-n2530c018ffab66304fc-ecee-1ca4-peer	🔒
Network	monks-vpc	🔒
Subnet	monks-subnet	🔒
Stack type	IPv4	✎
VPC-native traffic routing	Enabled	🔒
Cluster Pod IPv4 range (default)	10.11.0.0/21	🔒
Cluster Pod IPv4 ranges (additional) 	None	✎
Maximum pods per node	110	🔒
IPv4 service range	10.12.0.0/21	🔒
Intranode visibility	Disabled	✎
HTTP Load Balancing	Enabled	✎
Subsetting for L4 Internal Load Balancers	Disabled	✎
Control plane authorized networks	net1 (10.0.0.7/32)	✎

Os campos "External endpoint" e "Private Cluster" estão respectivamente configurados como "Disabled" e "Enabled", ou seja, configurações de um cluster privado, também é importante destacar os campos "Network", "Subnet" e Control Plane Authorized Network, campos que mostram que o cluster está utilizando as configurações feitas anteriormente.

5.6 Objetos Kubernetes

Nessa seção, a figura 8 a seguir corresponde aos objetos Kubernetes da aplicação conforme os arquivos .YAML descritos anteriormente.

Figura 8: Cluster Kubernetes Terminal

```

ivansousa_mediamonks@monks-host:~$ kubectl get no
NAME                                STATUS    ROLES    AGE    VERSION
gke-monks-cluster-monks-cluster-pool-27685795-020p  Ready    <none>   26m    v1.27.2-gke.1200
gke-monks-cluster-monks-cluster-pool-27685795-g669  Ready    <none>   26m    v1.27.2-gke.1200
ivansousa_mediamonks@monks-host:~$ kubectl get all -n monks-ns
NAME                                READY    STATUS    RESTARTS   AGE
pod/deploy-monks-77594789b4-5vncq  1/1      Running   0           7m24s
pod/deploy-monks-77594789b4-926zf  1/1      Running   0           7m20s

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
service/svc-monks                   ClusterIP      10.12.6.21     <none>         8080/TCP   14m

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/deploy-monks        2/2      2              2             14m

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/deploy-monks-65579b857b  0          0          0        14m
replicaset.apps/deploy-monks-77594789b4  2          2          2        7m24s

NAME                                REFERENCE                TARGETS    MINPODS    MAXPODS    REPLICAS    AGE
horizontalpodautoscaler.autoscaling/monks-hpa  Deployment/deploy-monks  <unknown>/80%  1          10         2            14m

ivansousa_mediamonks@monks-host:~$ kubectl get ingress -A
NAMESPACE   NAME      CLASS    HOSTS                ADDRESS          PORTS    AGE
monks-ns    ingress  <none>   104.154.63.119.nip.io  104.154.63.119  80       9m36s
ivansousa_mediamonks@monks-host:~$ kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
nginx-ingress-ingress-nginx-controller-9cbdodd9-rkcdv  1/1      Running   0           11m
ivansousa_mediamonks@monks-host:~$ kubectl get svc
NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes                         ClusterIP      10.12.0.1     <none>         443/TCP    35m
nginx-ingress-ingress-nginx-controller  LoadBalancer  10.12.4.23    104.154.63.119  80:31886/TCP,443:30577/TCP  12m
nginx-ingress-ingress-nginx-controller-admission  ClusterIP      10.12.7.116   <none>         443/TCP    12m
ivansousa_mediamonks@monks-host:~$

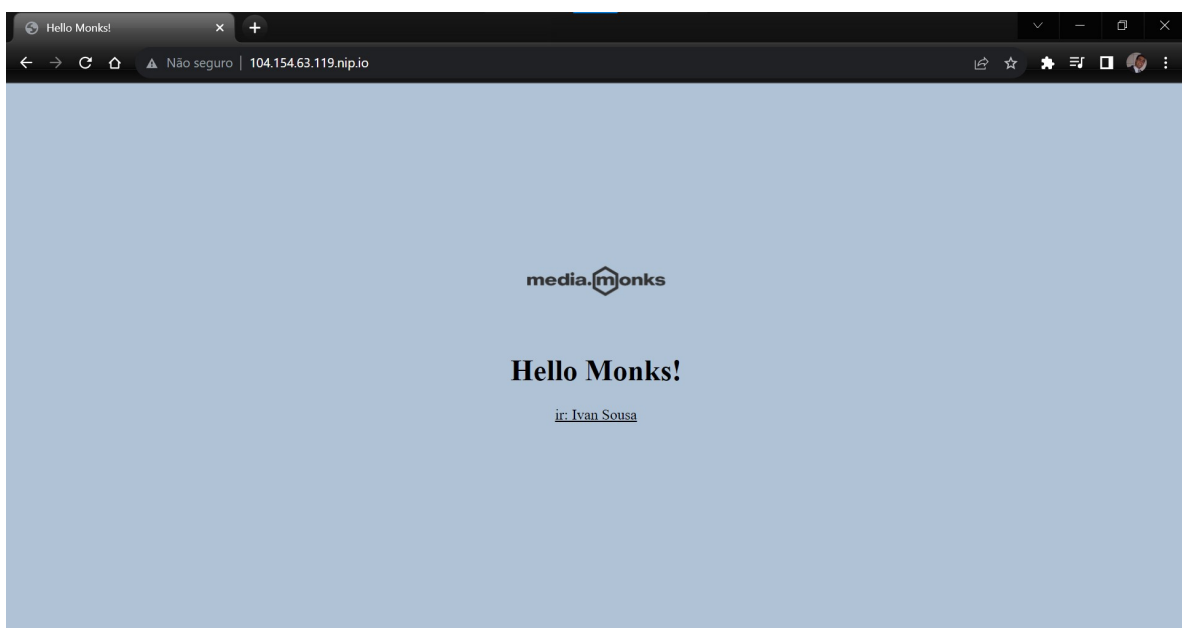
```

É possível observar os resultados gerados conforme solicitado, os nodes do cluster como visto anteriormente, o "deployment" e os objetos gerenciados por ele, pod e replicaset, o Horizontal Pod Autoscaler e o Ingress sendo esses últimos mencionados pertencentes ao namespace "monks-ns". Por fim os objetos Kubernetes pertencente ao NGIX-Ingress-Controller, o pod, o service clusterIP e o service do tipo LoadBalancer com seu External-IP.

5.7 Acesso

Por fim a imagem 10 que corresponde ao acesso a aplicação via Browser.

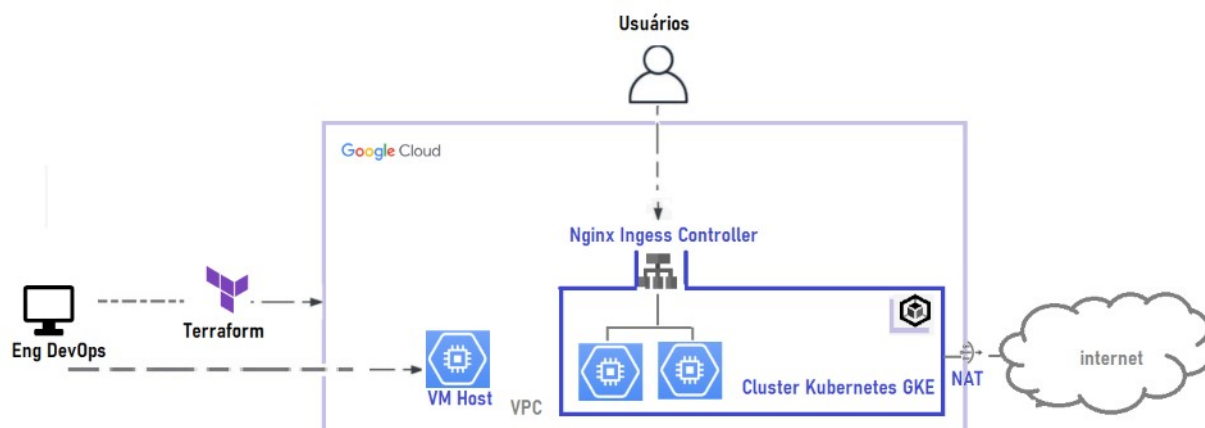
Figura 9: Acesso a aplicação Hello-Monks



6 Conclusão

O Projeto foi concluído conforme proposto no case e descrito no objetivo, a figura a seguir demonstra em partes a construção final do projeto utilizando Recursos Google Cluster Plataforma (GCP), Cluster Kubernetes Privado (GKE), objetos Kubernetes e recursos Terraform.

Figura 10: Cluster Kubernetes Privado



Para a elaboração do projeto foram utilizados além das documentações oficiais do Terraform, Google Cloud, Kubernetes e Nginx ingress controller, o material disponível no Case NGINX Ingress Controller no GKE ((@AMEER00, 2018)) e Como provisionar um Cluster Privado do GKE com Terraform ((K, 2023)).

Referências

@AMEER00. Ingress with nginx controller on google kubernetes engine. Community tutorials Google Cloud, 2018. Disponível em: <https://cloud.google.com/community/tutorials/nginx-ingress-gke>.

K, S. Gcp-terraform to deploy private gke cluster. Google Cloud Community, 2023. Disponível em: <https://medium.com/google-cloud/gcp-terraform-to-deploy-private-gke-cluster-bebb225aa7be>.