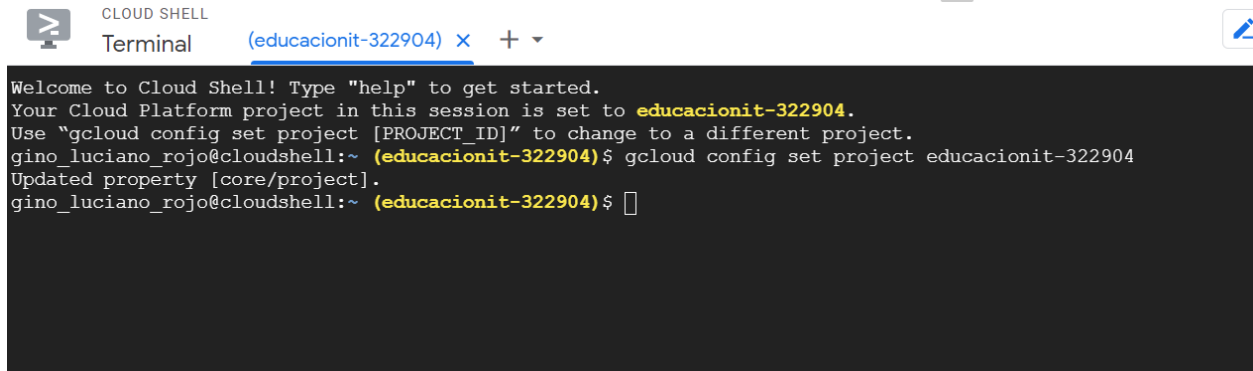


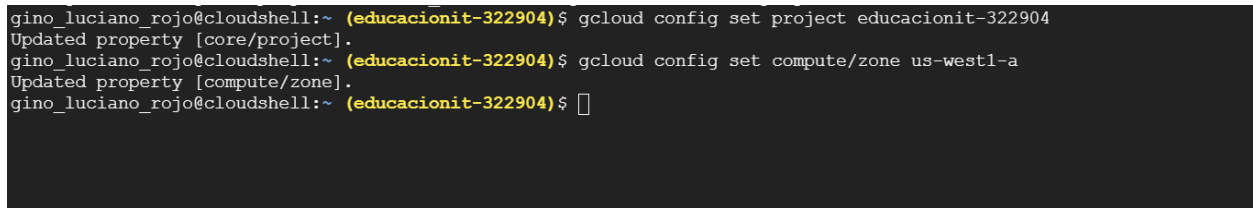
```
gcloud config set project educacionit-322904
```



The screenshot shows a Cloud Shell interface with a terminal window titled "(educacionit-322904)". The terminal output displays a welcome message and the successful execution of the command to set the project to "educacionit-322904".

```
Cloud Shell
Terminal (educacionit-322904) x + v
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to educacionit-322904.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
gino_luciano_rojo@cloudshell:~ (educacionit-322904)$ gcloud config set project educacionit-322904
Updated property [core/project].
gino_luciano_rojo@cloudshell:~ (educacionit-322904)$
```

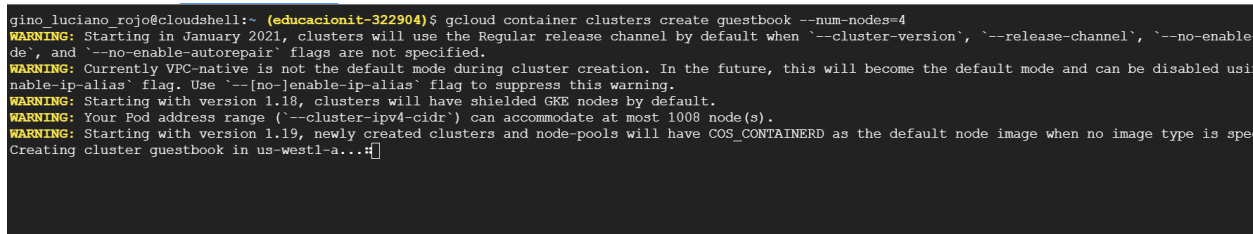
```
gcloud config set compute/zone us-west1-a
```



The screenshot shows the terminal window continuing from the previous command. The command to set the compute zone to "us-west1-a" has been executed successfully.

```
gino_luciano_rojo@cloudshell:~ (educacionit-322904)$ gcloud config set project educacionit-322904
Updated property [core/project].
gino_luciano_rojo@cloudshell:~ (educacionit-322904)$ gcloud config set compute/zone us-west1-a
Updated property [compute/zone].
gino_luciano_rojo@cloudshell:~ (educacionit-322904)$
```

```
gcloud container clusters create guestbook --num-nodes=4
```



The screenshot shows the terminal window with the command to create a container cluster. The output includes several warnings about release channels, VPC-native mode, shielded nodes, and node image types, followed by the start of cluster creation.

```
gino_luciano_rojo@cloudshell:~ (educacionit-322904)$ gcloud container clusters create guestbook --num-nodes=4
WARNING: Starting in January 2021, clusters will use the Regular release channel by default when '--cluster-version', '--release-channel', '--no-enable-de', and '--no-enable-autorepair' flags are not specified.
WARNING: Currently VPC-native is not the default mode during cluster creation. In the future, this will become the default mode and can be disabled using the '--no-enable-ip-alias' flag. Use '--[no-]enable-ip-alias' flag to suppress this warning.
WARNING: Starting with version 1.18, clusters will have shielded GKE nodes by default.
WARNING: Your Pod address range ('--cluster-ipv4-cidr') can accommodate at most 1008 node(s).
WARNING: Starting with version 1.19, newly created clusters and node-pools will have COS_CONTAINERD as the default node image when no image type is specified.
Creating cluster guestbook in us-west1-a...#
```

Cluster

```
Cloud Shell Terminal (educacionit-322904) x +
WARNING: Starting in January 2021, clusters will use the Regular release channel by default when '--cluster-version', '--release-channel', '--no-enable-ip-alias', and '--no-enable-autorepair' flags are not specified.
WARNING: Currently VPC-native is not the default mode during cluster creation. In the future, this will become the default mode and can be disabled with the '--enable-ip-alias' flag. Use '--no-enable-ip-alias' flag to suppress this warning.
WARNING: Starting with version 1.18, clusters will have shielded GKE nodes by default.
WARNING: Your Pod address range ('--cluster-ipv4-cidr') can accommodate at most 1008 node(s).
WARNING: Starting with version 1.19, newly created clusters and node-pools will have COS_CONTAINERD as the default node image when no image type is specified.
Creating cluster guestbook in us-west1-a...done.
Created [https://container.googleapis.com/v1/projects/educacionit-322904/zones/us-west1-a/clusters/guestbook].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/_gcloud/us-west1-a/guestbook?project=educacionit-322904
kubeconfig entry generated for guestbook.
NAME      LOCATION  MASTER_VERSION  MASTER_IP      MACHINE_TYPE  NODE_VERSION    NUM_NODES  STATUS
guestbook us-west1-a 1.20.8-gke.900  34.83.221.209  e2-medium    1.20.8-gke.900  4          RUNNING
gino_luciano_rojo@cloudshell:~ (educacionit-322904) $
```

## KubectI get nodes

```
Cloud Shell Terminal (educacionit-322904) x +
gino_luciano_rojo@cloudshell:~ (educacionit-322904) $ kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
gke-guestbook-default-pool-6ela826e-10r3  Ready    <none>   3m56s  v1.20.8-gke.900
gke-guestbook-default-pool-6ela826e-2ttt  Ready    <none>   3m55s  v1.20.8-gke.900
gke-guestbook-default-pool-6ela826e-9k8w  Ready    <none>   3m53s  v1.20.8-gke.900
gke-guestbook-default-pool-6ela826e-fgb4  Ready    <none>   3m54s  v1.20.8-gke.900
gino_luciano_rojo@cloudshell:~ (educacionit-322904) $
```

## gcloud container clusters list

```
gino_luciano_rojo@cloudshell:~ (educacionit-322904) $ gcloud container clusters list
NAME      LOCATION  MASTER_VERSION  MASTER_IP      MACHINE_TYPE  NODE_VERSION    NUM_NODES  STATUS
guestbook us-west1-a 1.20.8-gke.900  34.83.221.209  e2-medium    1.20.8-gke.900  4          RUNNING
```

## gcloud container clusters describe guestbook

```
gino_luciano_rojo@cloudshell:~ (educacionit-322904) $ gcloud container clusters describe guestbook
addonsConfig:
  gcePersistentDiskCsiDriverConfig:
    enabled: true
  kubernetesDashboard:
    disabled: true
  networkPolicyConfig:
    disabled: true
clusterIpv4Cidr: 10.76.0.0/14
createTime: '2021-08-14T04:48:17+00:00'
currentMasterVersion: 1.20.8-gke.900
```

# Configura el líder de Redis

La aplicación de libro de visita usa Redis para almacenar sus datos. La aplicación escribe sus datos en una instancia del líder de Redis y lee los datos de varias instancias de seguidores de Redis. El primer paso es implementar un líder de Redis.

Primero, clona los manifiestos de muestra:

```
git clone https://github.com/GoogleCloudPlatform/kubernetes-engine-samples
```

```
cd kubernetes-engine-samples/guestbook
```

```
git checkout abbb383
```

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ git checkout abbb383
HEAD is now at abbb383 Replace redis master/slave terminology with leader/follower (#127)
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$
```

Usa el archivo de manifiesto llamado `redis-leader-deployment` para implementar el líder de Redis. En este archivo de manifiesto, se especifica un Deployment de Kubernetes que ejecuta una sola réplica del Pod líder de Redis:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-leader
  labels:
    app: redis
    role: leader
    tier: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
        role: leader
        tier: backend
    spec:
      containers:
        - name: leader
          image: "docker.io/redis:6.0.5"
          resources:
            requests:
              cpu: 100m
```

memory: 100Mi  
ports:  
- containerPort: 6379

guestbook/redis-leader-deployment.yaml

[Ver en GitHub](#)

Kubectl apply -f redis-leader-deployment.yaml

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl apply -f redis-leader-deployment.yaml
deployment.apps/redis-leader created
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
redis-leader  1/1     1            1           13s
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get nodes
NAME                                     STATUS   ROLES    AGE   VERSION
gke-guestbook-default-pool-6e1a826e-10r3  Ready   <none>    14m   v1.20.8-gke.900
gke-guestbook-default-pool-6e1a826e-2ttt  Ready   <none>    14m   v1.20.8-gke.900
gke-guestbook-default-pool-6e1a826e-9k8w  Ready   <none>    14m   v1.20.8-gke.900
gke-guestbook-default-pool-6e1a826e-fgb4  Ready   <none>    14m   v1.20.8-gke.900
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$
```

Kubectl get deployment

Kubectl get nodes

Kubectl get pods

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP            NODE                                EXTERNAL-IP
redis-leader-fb76b4755-xpqxc        1/1     Running   0          117s  10.76.0.5    gke-guestbook-default-pool-6e1a826e-10r3  10.76.0.5
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$
```

Ejecuta el siguiente comando para ver los registros del pod del líder de Redis:

```
kubectl logs deployment/redis-leader
```

```
kubectl logs deployment/redis-leader
```

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl logs deployment/redis-leader
1:C 14 Aug 2021 05:05:38.103 # oOoOoOoOoOoOo Redis is starting oOoOoOoOoOoOo
1:C 14 Aug 2021 05:05:38.103 # Redis version=6.0.5, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 14 Aug 2021 05:05:38.103 # Warning: no config file specified, using the default config. In order to specify a config file use
f
1:M 14 Aug 2021 05:05:38.105 * Running mode=standalone, port=6379.
1:M 14 Aug 2021 05:05:38.105 # Server initialized
1:M 14 Aug 2021 05:05:38.105 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create late
h Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your
n the setting after a reboot. Redis must be restarted after THP is disabled.
1:M 14 Aug 2021 05:05:38.105 * Ready to accept connections
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$
```

Crea el servicio de líder de Redis

La aplicación del libro de visitas necesita comunicarse con el líder de Redis para escribir sus datos. Puedes crear un [Service](#) para poder usar un proxy en el tráfico hacia el pod del líder de Redis.

Un [Service](#) es una abstracción de Kubernetes que define un conjunto lógico de pods y una política para acceder a ellos. El Service es efectivamente un balanceador de cargas con nombre que actúa como proxy para el tráfico hacia uno o más pods. Cuando configuras un Service, describe en qué pods actuar como proxy en función de las etiquetas de los pods.

Mira el archivo de manifiesto redis-leader-service.yaml, en el que se describe un recurso de un Service para el líder de Redis:

```
apiVersion: v1
kind: Service
metadata:
  name: redis-leader
  labels:
    app: redis
    role: leader
    tier: backend
spec:
  ports:
    - port: 6379
      targetPort: 6379
  selector:
    app: redis
    role: leader
    tier: backend
```

Con este archivo de manifiesto, se crea un servicio llamado redis-leader con un conjunto de selectores de etiquetas. Estas coinciden con el conjunto de etiquetas implementado en el paso anterior. Por lo tanto, este Service enruta el tráfico de la red al pod líder de Redis que creaste en un paso anterior.

En la sección ports del manifiesto, se declara el mapa de un solo puerto. En este caso, el servicio enrutará el tráfico del port: 6379 al targetPort: 6379 de los contenedores que coincidan con las etiquetas selector especificadas. Ten en cuenta que el containerPort del Deployment debe coincidir con el targetPort a fin de enrutar el tráfico al Deployment.

Para iniciar el Service del líder de Redis, ejecuta lo siguiente:

```
kubectl apply -f redis-leader-service.yaml
```

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904) $ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes    ClusterIP     10.79.240.1   <none>         443/TCP    30m
redis-leader   ClusterIP     10.79.244.39  <none>         6379/TCP   7m35s
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904) $
```

## Configura los seguidores de Redis

Si bien el líder de Redis es un solo Pod, puedes hacer que tenga alta disponibilidad y cumplir con las demandas de tráfico si agregas algunos seguidores de Redis, o réplicas.

Observa el archivo de manifiesto `redis-follower-deployment.yaml`, en el que se describe una implementación para los pods de los seguidores de Redis:

guestbook/redis-follower-deployment.yaml

[Ver en GitHub](#)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-follower
  labels:
    app: redis
    role: follower
    tier: backend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
        role: follower
        tier: backend
```

```
spec:
  containers:
  - name: follower
    image: gcr.io/google_samples/gb-redis-follower:v2
    resources:
      requests:
        cpu: 100m
        memory: 100Mi
    ports:
    - containerPort: 6379
```

Para crear el Deployment de seguidores de Redis, ejecuta lo siguiente:

```
kubectl apply -f redis-follower-deployment.yaml
```

Luego, `kubectl get pods` para ver el output

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get pods --watch
NAME                                READY   STATUS    RESTARTS   AGE
redis-follower-dddfbdcc9-p72wm      1/1     Running   0           44s
redis-follower-dddfbdcc9-zvwdx      1/1     Running   0           44s
redis-leader-fb76b4755-xpqxc        1/1     Running   0           18m
```

Y luego

```
kubectl logs deployment/redis-follower
```



ica 10.76.3.4:6379 asks for synchronization

1:M 14 Aug 2021 05:23:13.158 \* Full resync requested by replica 10.76.3.4:6379

1:M 14 Aug 2021 05:23:13.158 \* Starting BGSAVE for SYNC with target: disk

1:M 14 Aug 2021 05:23:13.159 \* Background saving started by pid 21

21:C 14 Aug 2021 05:23:13.163 \* DB saved on disk

21:C 14 Aug 2021 05:23:13.164 \* RDB: 0 MB of memory used by copy-on-write

1:M 14 Aug 2021 05:23:13.217 \* Background saving terminated with success

1:M 14 Aug 2021 05:23:13.218 \* Synchronization with replica 10.76.3.4:6379 s

## Crea el servicio de seguidores de Redis

La aplicación de libro de visitas debe comunicarse con los seguidores de Redis para leer los datos. Para que los seguidores de Redis sean detectables, debes configurar otro Service.

redis-follower-service.yaml define la configuración de Service para los seguidores de Redis:

guestbook/redis-follower-service.yaml

[Ver en GitHub](#)

```
apiVersion: v1
kind: Service
metadata:
  name: redis-follower
  labels:
    app: redis
    role: follower
    tier: backend
spec:
  ports:
    # the port that this service should serve on
    - port: 6379
  selector:
    app: redis
    role: follower
    tier: backend
```

Con este archivo, se define un Service llamado `redis-follower` que se ejecuta en el puerto 6379. Ten en cuenta que el campo selector del Service coincide con los Pods seguidores de Redis creados en el paso anterior.

Para crear el servicio `redis-follower`, ejecuta el siguiente comando:

```
kubectl apply -f redis-follower-service.yaml
```

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl apply -f redis-follower-service.yaml
service/redis-follower created
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$
```

Kubectl get service

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get service
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes          ClusterIP   10.79.240.1    <none>         443/TCP    39m
redis-follower      ClusterIP   10.79.247.248  <none>         6379/TCP   42s
redis-leader        ClusterIP   10.79.244.39   <none>         6379/TCP    16m
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$
```

## Configura el frontend web del libro de visitas

Ahora que el almacenamiento de Redis del libro de visitas está listo para usarlo, inicia sus servidores web. Al igual que los seguidores de Redis, el frontend se implementa mediante un Deployment de Kubernetes.

La aplicación de libro de visitas usa un frontend PHP. Se configura para comunicarse con los Services de los seguidores o del líder de Redis, según si la solicitud es de lectura o de escritura. El frontend expone una interfaz JSON y entrega una UX basada en jQuery-Ajax.

Visualiza el archivo de manifiesto `frontend-deployment.yaml` que describe el Deployment para el servidor web del libro de visitas:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: guestbook
      tier: frontend
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v5
        env:
        - name: GET_HOSTS_FROM
          value: "dns"
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
      ports:
      - containerPort: 80
```

Para crear el Deployment del frontend web del libro de visitas, ejecuta el siguiente comando:

```
kubectl apply -f frontend-deployment.yaml
```

```
kubectl apply -f frontend-deployment.yaml
```

## Kubectl get deployment --watch

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl apply -f frontend-deployment.yaml
deployment.apps/frontend created
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
frontend      0/3     3            0           9s
redis-follower 2/2     2            2           10m
redis-leader  1/1     1            1           28m
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get deployment --watch
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
frontend      2/3     3            2           29s
redis-follower 2/2     2            2           10m
redis-leader  1/1     1            1           28m
□
```

Consulta la lista de etiquetas que identifican el frontend web para verificar que las tres réplicas estén en ejecución:

```
kubectl get pods -l app=guestbook -l tier=frontend
```

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get pods -l app=guestbook -l tier=frontend
NAME                                READY   STATUS    RESTARTS   AGE
frontend-85595f5bf9-2dpcz          1/1     Running   0          3m8s
frontend-85595f5bf9-6mv4t          1/1     Running   0          3m8s
frontend-85595f5bf9-hj8bc          1/1     Running   0          3m8s
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ □
```

## Kubectl get pods

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
frontend-85595f5bf9-2dpcz          1/1     Running   0          3m36s
frontend-85595f5bf9-6mv4t          1/1     Running   0          3m36s
frontend-85595f5bf9-hj8bc          1/1     Running   0          3m36s
redis-follower-dddfbdcc9-p72wm     1/1     Running   0          14m
redis-follower-dddfbdcc9-zvwdx     1/1     Running   0          14m
redis-leader-fb76b4755-xpqxc       1/1     Running   0          31m
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ □
```

En el archivo de manifiesto, se especifica la variable de entorno `GET_HOSTS_FROM=dns`. Cuando proporcionas la configuración a la aplicación del frontend web del libro de visitas, la aplicación del frontend usa los nombres de host `redis-follower` y `redis-leader` para realizar una búsqueda de DNS. Mediante la búsqueda de DNS, se obtienen las direcciones IP de los Service

respectivos que creaste en los pasos anteriores. Este concepto se llama descubrimiento de servicios de DNS.

```
- name: GET_HOSTS_FROM
  value: "dns"
```

## Expón el frontend en una dirección IP externa

Los Service redis-follower y redis-leader que creaste en los pasos anteriores solo son accesibles en el clúster de GKE, porque el tipo predeterminado para un Service es [ClusterIP](#). Con ClusterIP, se proporciona una sola dirección IP para el conjunto de Pods al que se direcciona el Service. Solo se puede acceder a esta dirección IP en el clúster.

Sin embargo, el Service del frontend web del libro de visitas debe ser visible de manera externa. Es decir, quieres que un cliente pueda solicitar el Service desde fuera del clúster de GKE. Para realizar esta solicitud, puedes [especificar type: LoadBalancer o type: NodePort](#) en la configuración del Service, según tus requisitos. En este ejemplo, usarás type: LoadBalancer. El archivo de manifiesto frontend-service.yaml en el que se especifica esta configuración tiene el siguiente aspecto:

guestbook/frontend-service.yaml

[Ver en GitHub](#)

```
apiVersion: v1
kind: Service
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  type: LoadBalancer
  ports:
    # the port that this service should serve on
    - port: 80
  selector:
    app: guestbook
    tier: frontend
```

Cuando se crea el servicio frontend, GKE crea un [balanceador de cargas](#) y una dirección IP externa. Ten en cuenta que estos recursos están [sujetos a facturación](#). En la declaración del puerto en la sección ports, se especifica port: 80, pero no se especifica targetPort. Cuando omites la propiedad targetPort, su valor predeterminado es el valor del campo port. En este caso, este Service enruta el tráfico externo del puerto 80 al puerto 80 de los contenedores del Deployment frontend.

Para crear el Service, ejecuta el siguiente comando:

```
kubectl apply -f frontend-service.yaml
```

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl apply -f frontend-service.yaml
service/frontend created
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get svc
```

Kubectl get svc --watch hasta que se nos brinda la IP Externa del Backend

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get svc --watch
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
frontend      LoadBalancer  10.79.244.180  <pending>      80:30150/TCP     17s
kubernetes    ClusterIP     10.79.240.1    <none>         443/TCP          51m
redis-follower ClusterIP     10.79.247.248  <none>         6379/TCP         12m
redis-leader   ClusterIP     10.79.244.39   <none>         6379/TCP         28m
frontend      LoadBalancer  10.79.244.180  34.127.99.125  80:30150/TCP     78s
█
```

## Visita el sitio web del libro de visitas

Para acceder al Service del libro de visitas, ejecuta el siguiente comando a fin de buscar la IP externa del Service que configuraste:

kubectl get service frontend

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get svc --watch
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
frontend      LoadBalancer  10.79.244.180  <pending>      80:30150/TCP     17s
kubernetes    ClusterIP     10.79.240.1    <none>         443/TCP          51m
redis-follower ClusterIP     10.79.247.248  <none>         6379/TCP         12m
redis-leader   ClusterIP     10.79.244.39   <none>         6379/TCP         28m
frontend      LoadBalancer  10.79.244.180  34.127.99.125  80:30150/TCP     78s
^Cgino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get service frontend
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
frontend      LoadBalancer  10.79.244.180  34.127.99.125  80:30150/TCP     2m34s
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ █
```

Ahora, accede a la dirección IP externa de tu FrontEnd mediante el Load Balancer, usando un navegador

The screenshot shows two side-by-side windows. The left window is the Google Cloud Platform console for the project 'educacionit'. It displays project information and a terminal window. The terminal shows the command to create the frontend service and two tables listing the resources created.

**Terminal Output:**

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/  
service/frontend created  
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/  
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP  
frontend      LoadBalancer  10.79.244.180  <pending>  
kubernetes     ClusterIP      10.79.240.1    <none>  
redis-follower ClusterIP      10.79.247.248  <none>  
redis-leader   ClusterIP      10.79.244.39   <none>  
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/  
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP  
frontend      LoadBalancer  10.79.244.180  <pending>  
kubernetes     ClusterIP      10.79.240.1    <none>  
redis-follower ClusterIP      10.79.247.248  <none>  
redis-leader   ClusterIP      10.79.244.39   <none>  
frontend      LoadBalancer  10.79.244.180  34.127.99.125  
^Cgino_luciano_rojo@cloudshell:~/kubernetes-engine-sample  
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP  
frontend      LoadBalancer  10.79.244.180  34.127.99.125  
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/
```

The right window is a web browser showing the 'Guestbook' application. It has a text input field labeled 'Messages' and a 'Submit' button. Below the input, it displays the message: 'Tu Primera Implementacion de Kubernetes Funciona! Gino Luciano Rojo, EducacionIT'.

Funciona!

# Guestbook

Tu Primera Implementacion de Kubernetes Funciona! Gino Luciano Rojo, EducacionIT  
Kubernetes-EducacionIT

Prueba agregar algunas entradas del libro de visitas. Para ello, escribe un mensaje y haz clic en **Enviar** (Submit). El mensaje que escribiste aparece en el frontend. Este mensaje indica que los datos se agregaron a Redis de forma correcta a través de los Service que creaste antes.

## Cómo escalar verticalmente el frontend web

Supongamos que tu aplicación de libro de visitas lleva un tiempo en ejecución y, de un momento a otro, se llena de publicidad. Entonces, decides que sería una buena idea agregar más servidores web a tu frontend. Esto es fácil de hacer, ya que tus servidores están definidos como un servicio que usa un controlador de implementación.

Para escalar verticalmente la cantidad de pods de frontend a cinco, ejecuta el siguiente comando:

```
kubectl scale deployment frontend --replicas=5
```



La configuración del Deployment se actualizará para indicar que debería haber cinco réplicas en ejecución. El Deployment ajusta la cantidad de pods que ejecuta para que coincidan con la configuración. Para verificar la cantidad de réplicas, ejecuta el siguiente comando:

Kubectl get pods --watch

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get pods --watch
NAME                                READY   STATUS    RESTARTS   AGE
frontend-85595f5bf9-2dpcz          1/1     Running   0           21m
frontend-85595f5bf9-6mv4t          1/1     Running   0           21m
frontend-85595f5bf9-hj8bc          1/1     Running   0           21m
frontend-85595f5bf9-rmxgq          1/1     Running   0           40s
frontend-85595f5bf9-s7bh8          1/1     Running   0           40s
redis-follower-dddfbdcc9-p72wm      1/1     Running   0           31m
redis-follower-dddfbdcc9-zvwdx      1/1     Running   0           31m
redis-leader-fb76b4755-xpqxc        1/1     Running   0           49m
□
```

Listo! Ya hemos finalizado nuestro Laboratorio, ahora, resta evitar costos innecesarios

## Realiza una limpieza

Para evitar que se apliquen cargos a tu cuenta de Google Cloud por los recursos usados en este instructivo para Kubernetes por EducacionIT, borra el proyecto que contiene los recursos o conserva el proyecto y borra los recursos individuales.

1. **Borra el Service:** En este paso, se desasigna el balanceador de cargas de Cloud creado para el Service de frontend:

kubectl delete service frontend

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl delete service frontend
service "frontend" deleted
□
```

**Espera hasta que se haya borrado el balanceador de cargas aprovisionado para el Service de frontend:** El balanceador de cargas se borra de forma asíncrona en segundo plano cuando

ejecutas `kubect1 delete`. Mira el resultado del siguiente comando y espera hasta que el balanceador de cargas se haya borrado:

Se borro el LoadBalancer

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl delete service frontend
service "frontend" deleted
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
kubernetes           ClusterIP   10.79.240.1      <none>        443/TCP           67m
redis-follower        ClusterIP   10.79.247.248    <none>        6379/TCP          28m
redis-leader          ClusterIP   10.79.244.39     <none>        6379/TCP          44m
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ Ya no esta el LB!
```

`gcloud compute forwarding-rules list`

```
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$ gcloud compute forwarding-rules list
Listed 0 items.
gino_luciano_rojo@cloudshell:~/kubernetes-engine-samples/guestbook (educacionit-322904)$
```

Y por ultimo, eliminamos el Cluster guestbook

`gcloud container clusters delete guestbook`

```
gino_luciano_rojo@cloudshell:~/kubernetes-
The following clusters will be deleted.
- [guestbook] in [us-west1-a]

Do you want to continue (Y/n)? y

Deleting cluster guestbook....
```

