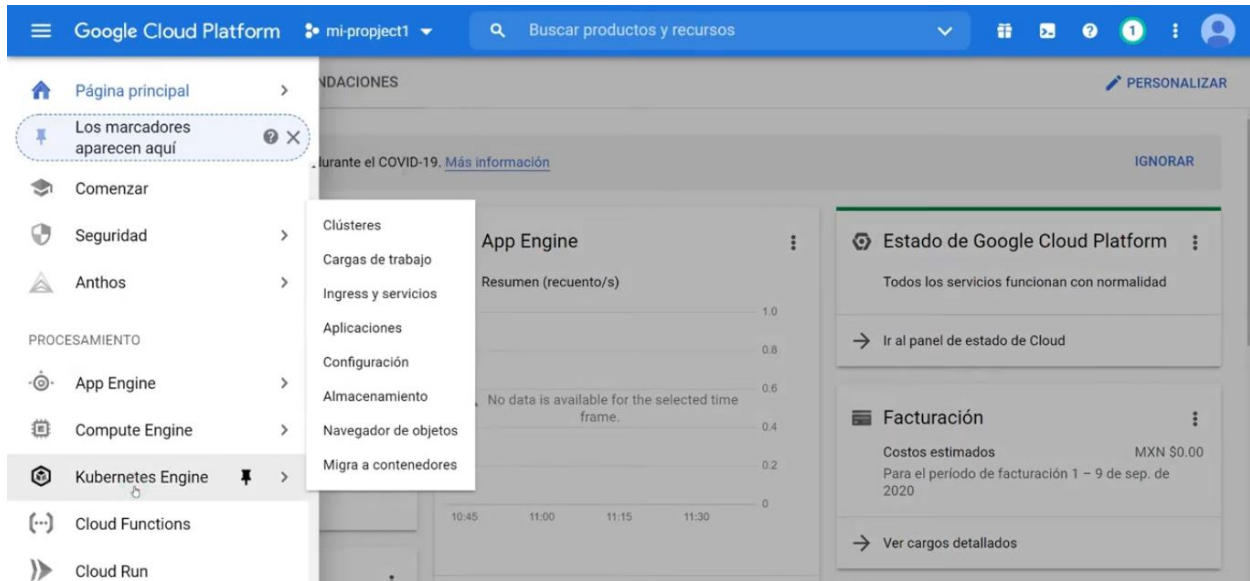


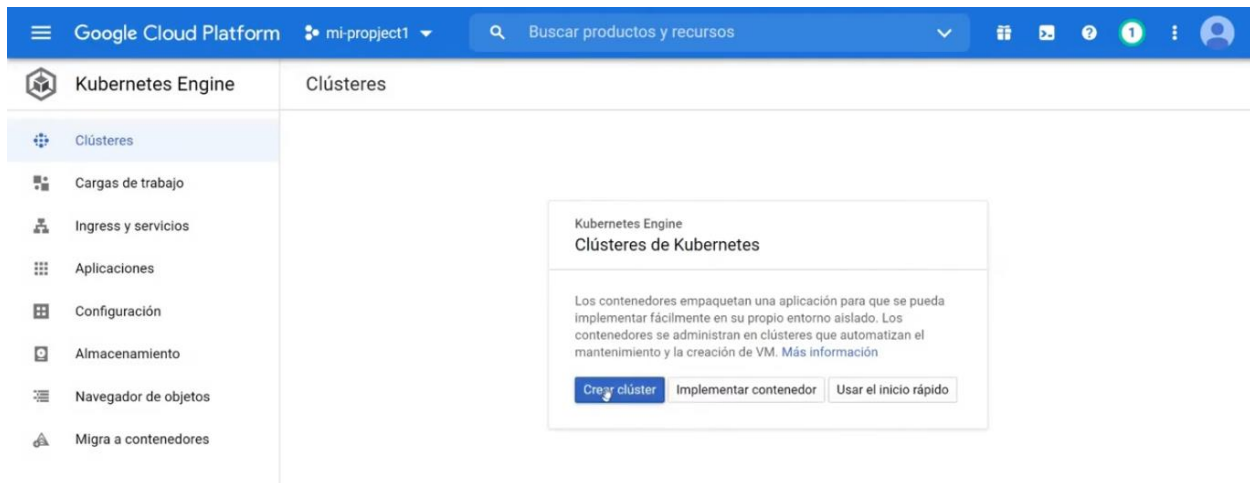
Manual de implementacion Kubernetes

Creación de Clúster en GCP

Primero nos dirigimos a la pagina de Google Cloud Plataform, y en el menú buscamos la opción de clústeres.



Una vez en la sección de kubernetes engine, buscamos la opción crear clúster y lo seleccionamos, estos clusteres permiten alojar los pods que alojaran nuestros contenedores.



Al darle clic en crear, nos aparecerá un formulario donde debemos ingresar el nombre del clúster, lo demás se puede quedar con los valores por defecto

Google Cloud Platform mi-project1 Buscar productos y recursos

← Crea un clúster de Kubernetes + AGREGAR GRUPO DE NODOS QUITAR GRUPO DE NODOS

obtener experiencia

• Aspectos básicos del clúster

GRUPOS DE NODOS

- default-pool

CLÚSTER

- Automatización
- Redes
- Seguridad
- Metadatos
- Características

Nombre
mi-cluster

Tipo de ubicación

☒ Zonal
☐ Regional

Zona
us-central1-c

☐ Especificar las ubicaciones predeterminadas de nodos
Predeterminado actual: us-central1-c

Versión principal

Elige Canal de versiones para obtener actualizaciones de GKE automáticas, a medida que se lancen versiones nuevas. Elige una versión estática para actualizarla más adelante de forma manual. [Más información](#)

☒ Versión estática
☐ Canal de versiones

Versión estática
1.15.12-gke.2 (predeterminado)

CREATE CANCELAR REST o línea de comandos equivalente

Nos dirigimos a la sección de la izquierda donde está la opción default-pool, para determinar la cantidad de nodos, para nuestro caso necesitamos 2 nodos para frontend.

Google Cloud Platform mi-project1 Buscar productos y recursos

← Crea un clúster de Kubernetes + AGREGAR GRUPO DE NODOS QUITAR GRUPO DE NODOS

• Aspectos básicos del clúster

GRUPOS DE NODOS

- default-pool
- Nodos
- Seguridad
- Metadatos

CLÚSTER

- Automatización
- Redes
- Seguridad
- Metadatos
- Características

Detalles del grupo de nodos

El clúster nuevo se creará con al menos un grupo de nodos. Un grupo de nodos es una plantilla para los conjuntos de nodos creados en este clúster. Después de la creación del clúster, se pueden agregar y quitar más grupos de nodos.

Nombre
default-pool

Versión de nodo
1.15.12-gke.2 (versión principal)

Tamaño

Cantidad de nodos *
2 1

El rango de direcciones del pod limita el tamaño máximo del clúster. [Más información](#)

☐ Habilitar ajuste de escala automático

☐ Especificar las ubicaciones de los nodos
Valor predeterminado: us-central1-c

CREATE CANCELAR REST o línea de comandos equivalente

Ahora nos vamos a la opción de nodos, y vamos a elegir el tipo de maquina para nuestro clúster, por temas de costos elegimos la serie E2-micro que contiene 2 CPU virtuales y 1G de memoria. Seleccionamos el botón crear esperamos a que termine de levantar la máquina.

Google Cloud Platform mi-project1 Buscar productos y recursos

← Crea un clúster de Kubernetes AGREGAR GRUPO DE NODOS QUITAR GRUPO DE NODOS

Aspectos básicos del clúster

GRUPOS DE NODOS

- default-pool
 - Nodos
 - Seguridad
 - Metadatos

CLÚSTER

- Automatización
- Redes
- Seguridad
- Metadatos
- Características

Configuración de la máquina

Familia de máquinas

USO GENERAL OPTIMIZADA PARA PROCESAMIENTO MEMORIA OPTIMIZADA

Tipos de máquinas para cargas de trabajo comunes, optimizados en función del costo y la flexibilidad

Serie: E2

Selección de la plataforma de CPU según la disponibilidad

Tipo de máquina: e2-micro (2 CPU virtuales, 1 GB de memoria)

vCPU: 1 núcleo compartido

Memory: 1 GB

PLATAFORMA DE CPU Y GPU

Tipo de disco de arranque: Disco persistente estándar

Tamaño de disco de arranque (GB): 100

CREATE CANCELAR REST o línea de comandos equivalente

Para crear un nuevo clúster únicamente debemos seguir los pasos anteriores, el segundo clúster será para alojar el backen.

Google Cloud Platform mi-project1 Buscar productos y recursos

Kubernetes Engine Clúster...rnetes CREAR CLÚSTER + MOstrar PANEL DE INFORMACIÓN APRENDIZAJE

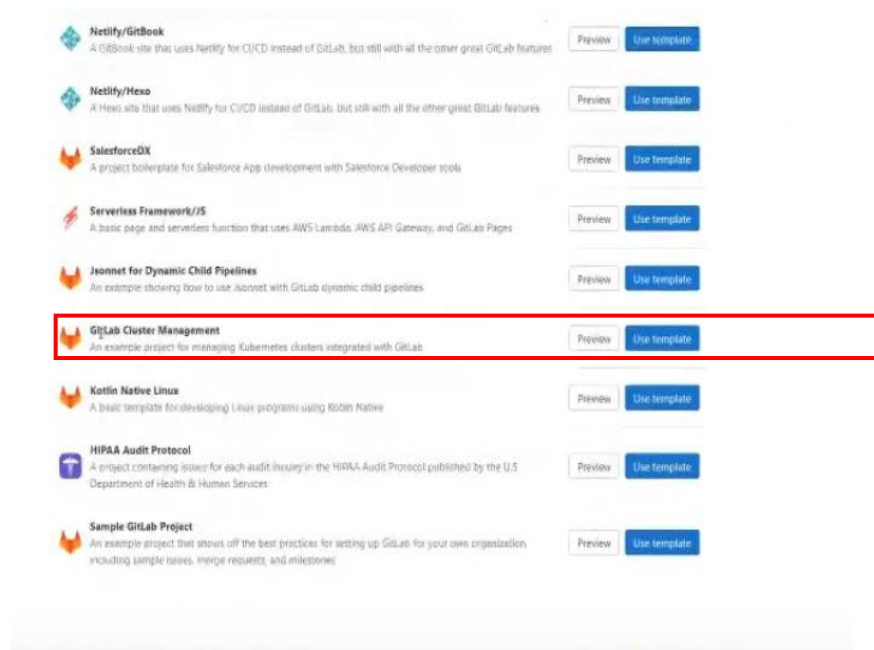
Un clúster de Kubernetes es un grupo administrado de instancias de VM para ejecutar aplicaciones en contenedores. Más información

Filtrar por nombre o etiqueta

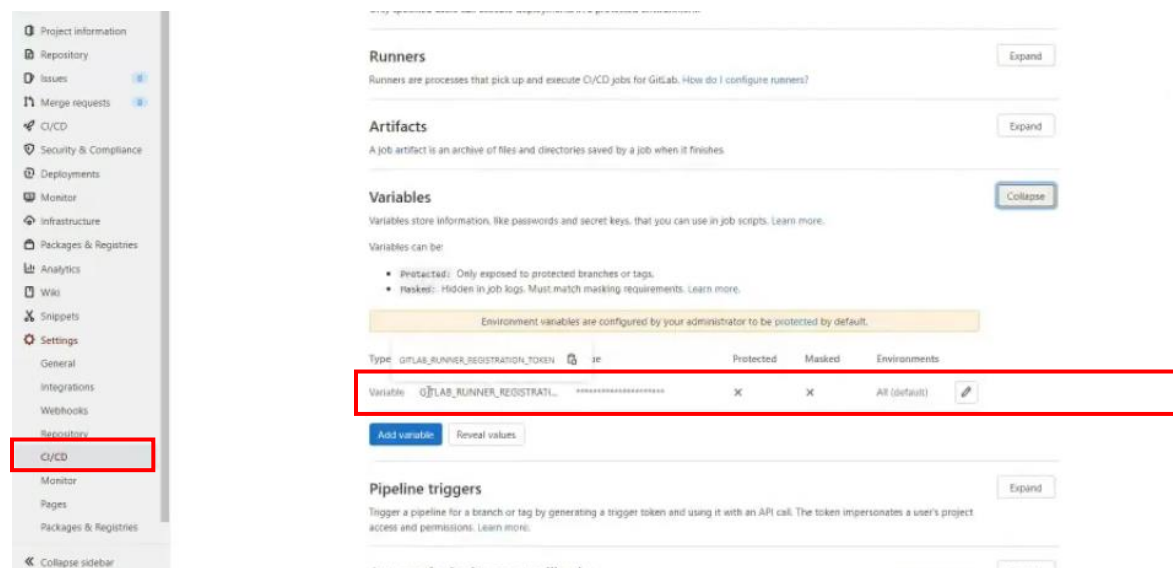
Nombre	Ubicación	Tamaño del clúster	Núcleos totales	Memoria total	Notifications	Etiquetas
mi-cluster	us-central1-c	2	4 CPU virtuales	2.00 GB		Conectar
mi-cluster2	us-central1-c	2	2 CPU virtuales	3.40 GB		Conectar

CLOUD SHELL Terminal (mi-project1) Abrir editor

Agregar la plantilla de clúster manager a nuestro repositorio, esta plantilla se agrega cuando creamos un repositorio a partir de una plantilla desde gitlab o bien podemos agregar la plantilla después de haber creado el repositorio. Esta plantilla incluye archivos de configurados, para que gitlab puede interactuar con un clúster



Agregar el token de registro de gitlab runner, para eso nos dirigimos al menú de la izquierda en la opción Settings -> CI/CD -> Variables



Agregar el archivo helmfile.yaml al repositorio, este archivo es básicamente un gestor de paquetes, donde nosotros le decimos que paquetes queremos que instale. Este archivo es muy útil para cuando realizamos integraciones con CI/CD. En la siguiente imagen se recomiendan instalar los módulos se indican en el recuadro.

```
helmfile.yaml 1.23 KB
1 helmDefaults:
2   atomic: true
3   wait: true
4
5 # ----- IMPORTANT -----
6 # Uncomment the paths below for the applications that you'd like to manage.
7 # By default all the helmfiles have `install:true`. So if you uncomment one of these
8 # helmfiles, the associated application will be tried to be installed or updated.
9 #
10 # You can set `install:false` to either uninstall the app from your cluster, or
11 # keep it uninstalled if you don't have it already installed.
12 #
13 # For more information, reference the Helmfile repository at:
14 # https://github.com/roboll/helmfile
15 # -----
16
17 helmfiles:
18 #
19 - path: applications/cilium/helmfile.yaml
20 - path: applications/ingress/helmfile.yaml
21 - path: applications/cert-manager/helmfile.yaml
22 # - path: applications/cert-manager-1-4/helmfile.yaml
23 # - path: applications/sentry/helmfile.yaml
24 - path: applications/gitlab-runner/helmfile.yaml
25 # - path: applications/elastic-stack/helmfile.yaml
26 - path: applications/prometheus/helmfile.yaml
27 # - path: applications/rudder/helmfile.yaml
28 # - path: applications/fluentd/helmfile.yaml
29 # - path: applications/falco/helmfile.yaml
```

Una vez que tengamos estas configuraciones, debemos instalar lo siguiente en nuestra máquina de desarrollo:

Instalar gitlab-runner

```
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | sudo bash
```

```
sudo apt-get install gitlab-runner
```

Permitir que se ejecute kubectl en nuestro cluster

Instalar docker

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Darle todos los privilegios al usuario de gitlab-runner, para evitar problemas de permisos

```
sudo nano /etc/pam.d/su
```

```
auth [success=ignore default=1] pam_succeed_if.so user = gitlab-runner
```

```
auth sufficient pam_succeed_if.so use_uid user ingroup gitlab-runner
```

```
sudo usermod -aG gitlab-runner ubuntu
```

```
su - gitlab-runner
```

```
auth    [success=ignore default=1] pam_succeed_if.so user = gitlab-runner
auth    sufficient pam_succeed_if.so use_uid user ingroup gitlab-runner
sudo usermod -aG gitlab-runner ubuntu
su - gitlab-runner
sudo usermod -aG gitlab-runner $USER
newgrp gitlab-runner
```

Instalar Google CP y Kubectl, se recomienda hacerlo con el usuario de gitlab-runner.

1. Agrega el URI de distribución del SDK de Cloud como una fuente de paquete:

```
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a
/etc/apt/sources.list.d/google-cloud-sdk.list
```

```
sudo apt-get install apt-transport-https ca-certificates gnupg
```

2. Importar la clave publica de GP

```
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring
/usr/share/keyrings/cloud.google.gpg add -
```

Instalar y actualizar el SDK de cloud

```
sudo apt-get update && sudo apt-get install google-cloud-sdk
```

3. Ejecutar gcloud init

```
gcloud init
```

Instalar kubectl desde el cluster

1.Descargar la ultima version

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Una vez instalado todas las herramientas y modificado los permisos para el usuario gitlab-runner para que pueda ejecutar kubectl, ahora podemos configurar nuestro flujo pipeline en el archivo gitlab-ci.yml

Debemos asegurarnos de que kubectl sea ejecutado por el usuario gitlab-runner y no por el usuario por defecto de Ubuntu, para eso podemos ejecutar el comando whoami.

Flujo del stage de kubernetes

```

#- "feature/decorator"
deploy-job-frontend:
  stage: kubernetes-stage
  only:
    - "main"
    - "develop"
  tags:
    - "ayd2_test"
  before_script:
    - cd ./angular-app
    - docker image rmi selvinlp/serviciofrontend:latest -f
    - docker pull selvinlp/serviciofrontend:latest
  script:
    #Debe estar dentro del directorio angular-app y ya deben estar creadas las imagenes para que funcione.
    - echo "Ejecutando deploy Frontend Kubernetes y LoadBalancer ..."
    - gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project ayd2-327723
    - kubectl get services
    - kubectl delete svc aydrive-front1
    #- kubectl delete svc aydrive-front2
    - kubectl delete deployments load-balancer-front-aydrive

    - kubectl apply -f load-balancer-frontAYD2.yaml
    - kubectl expose deployment load-balancer-front-aydrive --type=LoadBalancer --name=aydrive-front1
    #- kubectl expose deployment load-balancer-front-aydrive --type=LoadBalancer --name=aydrive-front2
    - kubectl get svc
  after_script:
    #- docker-compose push
    - cd ./
    - echo "Terminando deploy frontend ..."
deploy-job-backend:
```

Pasos:

1. Nos movemos al directorio angular-app

```
cd ./angular-app
```

2. Eliminamos todas las imágenes de los contenedores del frontend

```
docker image rmi selvinlp/serviciofrontend:latest -f
```

3. Descargamos las imágenes del frontend

```
docker pull selvinlp/serviciofrontend:latest
```

4. Nos conectamos al cluster que creamos al inicio

```
gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project ayd2-327723
```

6. Eliminamos el servicio y el deployment de nuestro frontend

```
kubectl delete svc aydrive-front1
```

```
kubectl delete deployments load-balancer-front-aydrive
```

7. Creamos el deployment para actualizar los cambios que se hayan hecho

```
kubectl apply -f load-balancer-frontAYD2.yaml
```

8. Volvemos a exponer el deployment del frontend, para que nos proporcione una ip publica

```
kubectl expose deployment load-balancer-front-aydrive --type=LoadBalancer --name=aydrive-front1
```

9. Mostramos la ip publica asignada

```
kubectl get svc
```

Aplicar el mismo procedimiento para el kubernetes-stage del backend

```
84
85 deploy-job-backend:
86   stage: kubernetes-stage
87   only:
88     - "main"
89     - "develop"
90     #- "feature/decorator"
91   tags:
92     - "ayd2_test"
93   before_script:
94     - docker-compose down -v --rm all
95     - docker-compose pull
96     - cd ./backend
97   script:
98     #Debe estar dentro del directorio backend y ya deben estar creadas las imagenes para que funcione.
99     - echo "Ejecutando deploy backend Kubernetes y LoadBalancer ..."
100    - gcloud container clusters get-credentials cluster-2 --zone us-central1-c --project ayd2-327723
101    - kubectl get services
102    - kubectl delete svc aydrive-backend1
103    #- kubectl delete svc aydrive-backend2
104    - kubectl delete deployments load-balancer-backend-aydrive
105
106    - kubectl apply -f load-balancer-backendAYD2.yaml
107    - kubectl expose deployment load-balancer-backend-aydrive --type=LoadBalancer --name=aydrive-backend1 --load-balancer-ip=35.232.111.58
108    #- kubectl expose deployment Load-balancer-backend-aydrive --type=LoadBalancer --name=aydrive-backend1 --load-balancer-ip=34.134.217.79
109    - kubectl get svc
110  after_script:
111    #- docker-compose push
112    - cd ./
113    - echo "Terminando deploy backend ... "
```
