

# Entrega 2 de la práctica 6

## • Objetivo

El objetivo de esta entrega es realizar una prueba de estrés a un cluster de kubernetes previamente desplegado. Para ello se va a crear un contenedor Docker con toda la funcionalidad incorporada. El despliegue del contenedor se va a realizar por medio de un Job de Kubernetes dentro del mismo cluster creado previamente.

## • Contenedor Docker

El contenedor Docker requerido debe de poder ejecutar comando ab, es decir que debe de tener instalada la librería apache2. El fichero dockerfile con la descripción del contenedor es el siguiente:

```
FROM ubuntu:latest

RUN apt-get -y update; \
    apt-get -y upgrade; \
    apt-get -y install apt-utils \
RUN apt install apache2-utils
```

## • Pasos para la construcción del contenedor

El contenedor no se va a construir en local, si no que se va a contruir en la nube de google. De esta forma no es necesario subir la imagen a posteri, si no que la imagen ya se encuentra disponible en el repository del proyecto. El comando a ejecutar es el siguiente:

```
raul@Maquina1:~/Documents/practica6$ gcloud builds submit --tag gcr.io/kubernetes-366509/locust-tasks:latest .
```


El resultado obtenido es el siguiente:

DONE					
ID	STATUS	CREATE_TIME	DURATION	SOURCE	IMAGES
35ead360-56b9-4937-a0c7-def14bc28463	SUCCESS	2022-10-24T16:03:23+00:00	435	gs://kubernetes-366509_cloudbuild/source/1666627401.788653-cbde1ba76f7484188ec1cdf07fc46b2.tgz	gcr.io/kubernetes-366509/locust-tasks (+1 more)

Aparece el contenedor creado en la nube de google

### Kubernetes

Filtro Ingresar el nombre o el valor de la propiedad

Nombre ↑	Nombre de host ?	Visibilidad ?
 <a href="#">locust-tasks</a>	gcr.io	Privado

## • Cluster master

---

Antes de poder desplegar la imagen que se acaba de crear, es necesario desplegar el cluster objetivo de la prueba de rendimiento. Para ello se va a emplear el siguiente fichero .yaml como descripción del cluster:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
spec:
  selector:
    matchLabels:
      run: php-apache
  replicas: 3
  template:
    metadata:
      labels:
        run: php-apache
    spec:
      containers:
        - name: php-apache
          image: k8s.gcr.io/hpa-example
          ports:
            - containerPort: 80
          resources:
            limits:
              cpu: 500m
            requests:
              cpu: 200m
---
apiVersion: v1
kind: Service
metadata:
  name: php-apache
labels:
  run: php-apache
spec:
  ports:
    - port: 80
  selector:
    run: php-apache
```

Además se van a seguir los siguientes pasos para la creación del cluster

## • Creación cluster

Antes de poder desplegar los pods, es necesario crear un cluster con un número de nodos determinado. Para ello se ejecutan los siguientes comandos:

## Cambio de la zona de cómputo

```
raul@Maquina1:~/Documents/practica6$ gcloud config set compute/zone europe-southwest1-a
Updated property [compute/zone].
```

## Creación del cluster

```
raul@Maquina1:~/Documents/practica6$ gcloud container clusters create cluster-master --num-nodes=3
```

Resultado:

```
kubeconfig entry generated for cluster-master.
NAME             LOCATION             MASTER_VERSION  MASTER_IP      MACHINE_TYPE  NODE_VERSION     NUM_NODES  STATUS
cluster-master   europe-southwest1-a  1.22.12-gke.2300 34.175.80.120  e2-medium    1.22.12-gke.2300 3          RUNNING
```

Una vez se ha creado el cluster, se despliegan los pods en los nodos con el siguiente comando

```
raul@Maquina1:~/Documents/practica6$ kubectl apply -f php-apache.yaml
```

Servicios creados:

```
deployment.apps/php-apache created
service/php-apache created
```

- Escalado cluster

Una vez que se ha creado el cluster es necesario configurarlo para que pueda escalar de forma horizontal y vertical, para ello se van a llevar a cabo los siguientes pasos:

- Escalado de pods con HPA

Para que tengan alta disponibilidad los pods

```
kubectl get deployment
```

Resultado que muestra el número de despliegues que se tienen actualmente. Se puede comprobar que todos los pods se encuentran levantados y disponibles

```
raul@Maquina1:~/Documents/practica6$ kubectl get deployment
W1025 18:55:40.408680      2688 gcp.go:119] WARNING: the gcp auth plugin is deprecated in v1.22+, unavailable in v1.26+; use gcloud instead.
To learn more, consult https://cloud.google.com/blog/products/containers-kubernetes/kubectl-auth-changes-in-gke
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
php-apache       3/3     3            3           3m5s
```

```
kubectl get deployment
```

Se aplica HPA

```
kubectl autoscale deployment php-apache --cpu-percent=50 --min="1" --max="3"
```

De esta forma se establece que un pod se replice cuando se supere el 50% de la CPU del nodo en el que se encuentre. Además se establece que como mínimo debe de existir una instancia de ese pod, y como máximo 3. Con el siguiente comando se comprueba que efectivamente los cambios han surtido efecto

```
kubectl get hpa
```

```
raul@Maquina1:~/Documents/practica6$ kubectl get hpa
W1025 18:58:58.967150 2723 gcp.go:119] WARNING: the gcp auth plugin is deprecated in v1.22+, unavailable in v1.26+; use gcloud instead.
To learn more, consult https://cloud.google.com/blog/products/containers-kubernetes/kubectl-auth-changes-in-gke
NAME          REFERENCE          TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
php-apache    Deployment/php-apache  0%/50%   1         3         3         88
```

- Autoescalado del cluster

En esta parte se va a configurar el número de nodos que puede escalar el cluster, y cuando lo debe de hacer. Con el siguiente comando se va a establecer que el cluster va a tener como mínimo un nodo, y como máximo 5. Además se le establece que la forma de decidir cuando escalar es la de por defecto "balanced"

```
gcloud container clusters update cluster-kubernetes --enable-autoscaling --min-nodes=1 --max-nodes=5
```

El cambio ha sido efectivo

```
raul@Maquina1:~/Documents/practica6$ gcloud container clusters update cluster-kubernetes --enable-autoscaling --min-nodes=1 --max-nodes=5
Default change: During creation of nodepools or autoscaling configuration changes for cluster versions greater than 1.24.1-gke.800 a default location policy is applied. For Spot and PVM it defaults to ANY, and for all other VM kinds a BALANCED policy is used. To change the default values use the '--location-policy' flag.
Updating cluster-kubernetes...done.
Updated [https://container.googleapis.com/v1/projects/kubernetes-366509/zones/europe-west4-a/clusters/cluster-kubernetes].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/europe-west4-a/cluster-kubernetes?project=kubernetes-366509
```

- Pods slave

Ahora que ya se ha desplegado el cluster que sirve el contenido web, se va a proceder a crear el cluster que realiza la prueba de rendimiento. Este cluster despliega 5 pods con el contenedor que se ha creado en el primer apartado. El fichero .yaml con la descripción del despliegue es el siguiente:

```
apiVersion: batch/v1beta1
kind: CronJob
```

```
metadata:
name: mycronjob
spec:
schedule: "*/5 * * * *"
jobTemplate:
spec:
parallelism: 5
template:
spec:
containers:
- name: ab
image: gcr.io/kubernetes-366509/locust-tasks:latest
command: ["ab", "-n", "10000", "-c", "10", "http://php-apache/"]
restartPolicy: Never
backoffLimit: 2

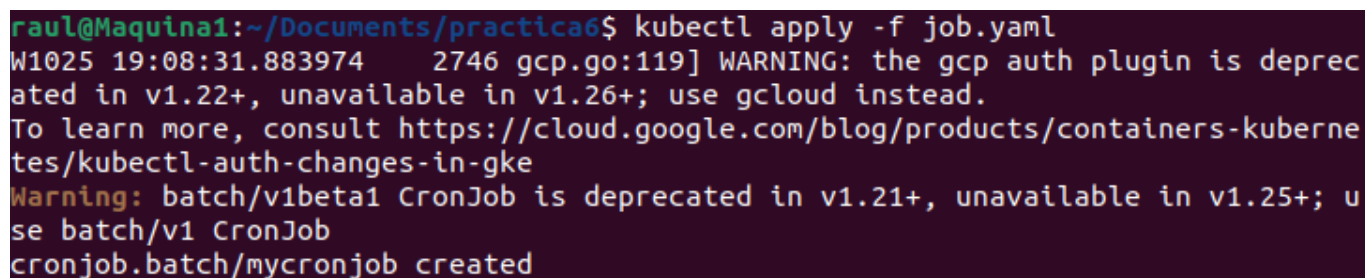
concurrencyPolicy: Allow
```

- Despliegue pods slave

Una vez se tiene el contenedor en el registry de GCE y el fichero del despliegue correctamente configurado, se puede proceder a realizar el despliegue del job. Para ello se ejecuta el siguiente comando

```
kubectl apply -f job.yaml
```

Efectivamente, se ha desplegado el job en el cluster



```
raul@Maquina1:~/Documents/practica6$ kubectl apply -f job.yaml
W1025 19:08:31.883974    2746 gcp.go:119] WARNING: the gcp auth plugin is deprecated in v1.22+, unavailable in v1.26+; use gcloud instead.
To learn more, consult https://cloud.google.com/blog/products/containers-kubernetes/kubectl-auth-changes-in-gke
Warning: batch/v1beta1 CronJob is deprecated in v1.21+, unavailable in v1.25+; use batch/v1 CronJob
cronjob.batch/mycronjob created
```

En la siguiente imagen se puede comprobar como se han levantado los pods, y como efectivamente la prueba ha funcionado correctamente.

```
raul@Maquina1:~/Documents/practica6$ kubectl get pods
W1025 19:34:20.989259 3244 gcp.go:119] WARNING: the gcp auth plugin is deprecated in v1.22+, unavailable in v1.26+; use gcloud instead.
To learn more, consult https://cloud.google.com/blog/products/containers-kubernetes/kubectl-auth-changes-in-gke
NAME                                READY   STATUS    RESTARTS   AGE
mycronjob-27778654-4g72q           1/1     Running   0           21s
mycronjob-27778654-55l97           1/1     Running   0           21s
mycronjob-27778654-hhjwv           1/1     Running   0           21s
mycronjob-27778654-vp6r4           1/1     Running   0           21s
mycronjob-27778654-xn9rh           1/1     Running   0           21s
php-apache-d4cf67d68-kpb4p         1/1     Running   0           41m
```

En la siguiente imagen se puede apreciar como el cluster se ha adaptado a la carga que tenía en el instante de la prueba de carga, ya que el cluster automáticamente ha escalado al número de pods máximo que se le había establecido, es decir 3

```
mycronjob-27778655-mbht2           1/1     Running   0           3s
mycronjob-27778655-q2wt5           1/1     Running   0           3s
mycronjob-27778655-xbj79           1/1     Running   0           3s
php-apache-d4cf67d68-fmfmX         1/1     Running   0           8s
php-apache-d4cf67d68-kpb4p         1/1     Running   0           42m
php-apache-d4cf67d68-tttdg         1/1     Running   0           38s
```

## • Destrucción cluster

---

Cuando se ha terminado la práctica es necesario destruir el cluster creado, para ello se ejecuta el siguiente comando:

```
gcloud container clusters delete cluster-kubernetes --quiet
```