

**UNIVERSIDAD DE GUADALAJARA**  
**CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS**

**Materia:** Computación tolerante a fallas

**Clave:** I7036

**Sección:** D06



**NRC:** 179961

Kubernetes

**Código estudiante:** 221350567

**Alumno:** Rauf Alfonso Hamden Estrada

**Carrera:** Ingeniería en Computación

**Fecha:** 20/11/2023

**Docente:** Michel Emanuel Lopez Franco

2023B

# Kubernetes

## Introduccion

Docker Compose es una herramienta para definir y ejecutar aplicaciones de Docker de varios contenedores. En Compose, se usa un archivo YAML para configurar los servicios de la aplicación. Después, con un solo comando, se crean y se inician todos los servicios de la configuración

## Desarrollo

### ¿Qué es Kubernetes?

Kubernetes es una plataforma de código abierto diseñada para automatizar la implementación, escalado y operación de aplicaciones en contenedores. Permite a los desarrolladores gestionar aplicaciones en contenedores de manera eficiente y proporciona herramientas para la automatización de tareas de implementación, escalado

### ¿Qué es Ingress?

Un Ingress es un recurso que gestiona el acceso externo a los servicios dentro de un clúster. Proporciona una forma de exponer servicios HTTP y HTTPS de manera externa, permitiendo el enrutamiento del tráfico y la configuración de reglas basadas en hosts y rutas. Ingress actúa como un controlador de recursos de capa 7 y permite la configuración centralizada de reglas para el manejo del tráfico.

### ¿Qué es un LoadBalancer?

El Load Balancer puede operar a nivel de red o a nivel de aplicación. Esto permite mejorar la disponibilidad y la capacidad de respuesta de los sistemas, así como escalar horizontalmente para manejar un mayor volumen de tráfico.

**Para poder hacerlo utilizaremos el Docker compose y para ello checamos que lo tengamos instalado**

```
C:\Users\raufa>docker-compose --version
Docker Compose version v2.22.0-desktop.2
C:\Users\raufa>
```

## iniciamos el cluster

```
PS C:\Users\raufa\OneDrive\Escritorio\Programas Python\Cluster\kompose\examples> docker-compose up -d
>>
[+] Running 27/14
✓ frontend 5 layers [██████] 0B/0B Pulled
✓ redis-master 8 layers [██████] 0B/0B Pulled
✓ redis-replica 11 layers [██████] 0B/0B Pulled

[+] Building 0.0s (0/0)
[+] Running 4/4
✓ Network examples_default Created
✓ Container examples-frontend-1 Started
✓ Container examples-redis-replica-1 Started
✓ Container examples-redis-master-1 Started
```

## Checamos que si se hayan inicializado

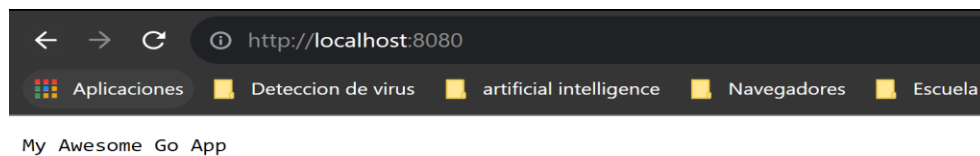
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.3693]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\raufa>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
853f63091b66   registry.k8s.io/redis-slave:v2     "/bin/sh -c /run.sh"    51 seconds ago Up 43 seconds 0.0.0.0:62070-
>6379/tcp     examples-redis-replica-1
a2dd8464bab2   registry.k8s.io/guestbook:v3       "./guestbook"           51 seconds ago Up 44 seconds 0.0.0.0:80->80
/tcp, 3000/tcp examples-frontend-1
1120213af3cc   registry.k8s.io/redis:e2e         "redis-server /etc/r..." 51 seconds ago Up 44 seconds 0.0.0.0:62069-
>6379/tcp     examples-redis-master-1
2eaa87733d80   getting-started                    "docker-entrypoint.s..." 29 minutes ago Up 28 minutes 127.0.0.1:3000
->3000/tcp     loving_bassi

C:\Users\raufa>
```

CONTAINER ID	IMAGE NAMES	COMMAND	CREATED	STATUS	PORTS
853f63091b66	registry.k8s.io/redis-slave:v2	"/bin/sh -c /run.sh"	24 minutes ago	Up 24 minutes	0.0.0.0:62070->6379/tcp
a2dd8464bab2	registry.k8s.io/guestbook:v3	"/guestbook"	24 minutes ago	Up 24 minutes	0.0.0.0:80->80/tcp, 3000/tcp
1120213af3cc	registry.k8s.io/redis:e2e	"redis-server /etc/r..."	24 minutes ago	Up 24 minutes	0.0.0.0:62069->6379/tcp
2eaa87733d80	getting-started	"docker-entrypoint.s..."	53 minutes ago	Up 53 minutes	127.0.0.1:3000->3000/tcp
	loving_bassi				

## Mostramos lo que nos creamos en el puerto 8080



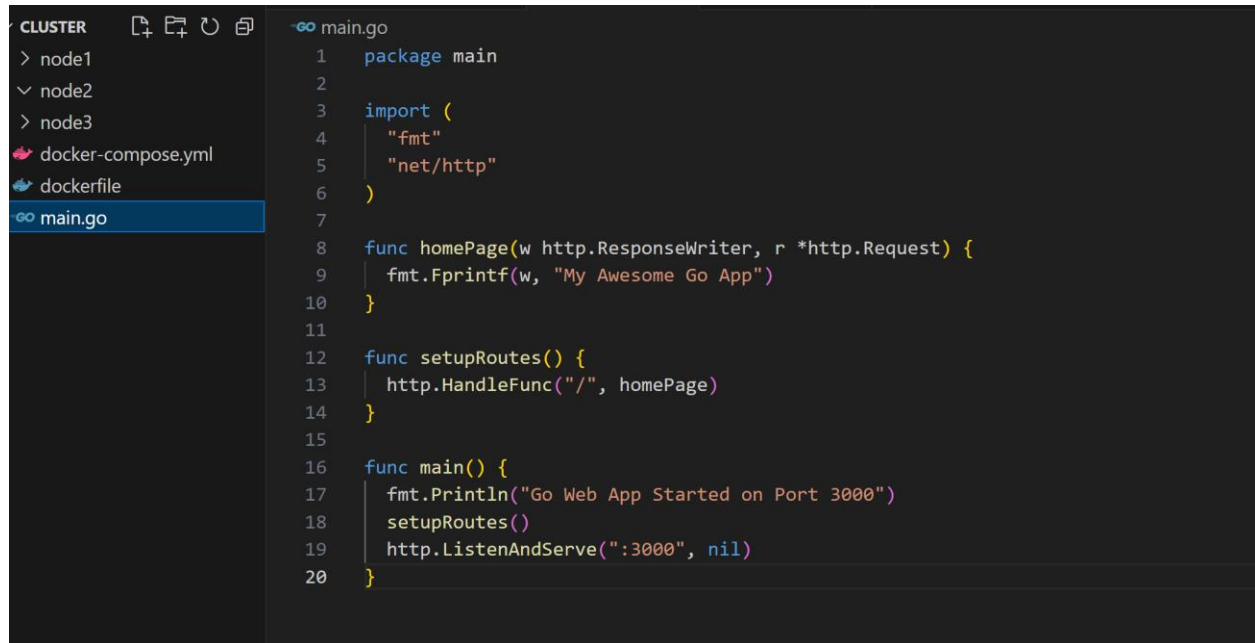
## Mostramos el Yml

```

! deployment.yml
1  ---
2  apiVersion: apps/v1
3  kind: Deployment
4  metadata:
5    name: go-web-app
6  spec:
7    replicas: 5
8    selector:
9      matchLabels:
10       name: go-web-app
11   template:
12     metadata:
13       labels:
14         name: go-web-app
15     spec:
16       containers:
17       - name: application
18         image: sammy/go-web-app
19         imagePullPolicy: IfNotPresent
20         ports:
21         - containerPort: 3000

```

## Main.go



```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6 )
7
8 func homePage(w http.ResponseWriter, r *http.Request) {
9     fmt.Fprintf(w, "My Awesome Go App")
10 }
11
12 func setupRoutes() {
13     http.HandleFunc("/", homePage)
14 }
15
16 func main() {
17     fmt.Println("Go Web App Started on Port 3000")
18     setupRoutes()
19     http.ListenAndServe(":3000", nil)
20 }
```

## Conclusión

Docker Compose simplifica la orquestación de aplicaciones ya que permite la definición y gestión de servicios, redes y volúmenes en un archivo YAML. Esto facilita la creación, configuración y ejecución de aplicaciones multi-contenedor. La utilización de Docker compose para la creación de kubernetes no fue difícil al checar la documentación y videos que fueron dados por el profesor