Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Software Avanzado
Primer Semestre 2024
Catedrático:
Ing. Everest Darwin Medinilla Rodríguez
Ing. Marco Tulio Aldana Prillwitz
Tutor académico:
Diego Molina
Ariana Pérez

DOCKER, KUBERNETES, MICROSERVICIOS

Practica 2

Marco Antonio Xocop Roquel
201122934

TECNOLOGÍAS USADAS

Máquinas virtuales:

Se utilizará 1 instancia en Google Cloud con las siguientes características:

- 2 CPUs y 4GB RAM

Microservicio 1

El microservicio se encargará de hacer consultas a Agify enviándole únicamente el nombre.

- Lenguaje: Python
- librería:
    - Flask 3.0.2
    - Flask-Cors 4.0.0

Microservicio 2

El microservicio se encargará de hacer consultas a Genderize enviándole únicamente el nombre.

- Lenguaje: Python
- librería:
    - Flask 3.0.2
    - Flask-Cors 4.0.0

Microservicio 1

El microservicio se encargará de realizar peticiones a los 2 microservicios anteriores.

- Lenguaje: Python
- librería:
    - Flask 3.0.2
    - Flask-Cors 4.0.0

KUBERNETES Y SUS COMPONENTES

KUBERNETES

Kubernetes es una plataforma de código abierto para la automatización, implementación, escalado y administración de aplicaciones en contenedores. A continuación, describo los componentes principales de Kubernetes:

COMPONENTES:

Master Node (nodo maestro):

- **kube-apiserver:** Es el punto de entrada para la API de Kubernetes. Procesa las operaciones de gestión de clúster a través de la API REST.
- **etcd:** Almacena de forma persistente la configuración del clúster y el estado del clúster. Es una base de datos de clave-valor consistente y altamente disponible.
- **kube-scheduler:** Es responsable de asignar trabajos a los nodos del clúster, teniendo en cuenta los recursos disponibles y los requisitos del trabajo.
- **kube-controller-manager:** Un conjunto de controladores que supervisan continuamente el estado del clúster y realizan acciones para mantener el estado deseado. Incluye controladores como el controlador de nodos, el controlador de replicación y el controlador de estado.

Worker Node (nodo de trabajo):

- **kubelet:** Agente que se ejecuta en cada nodo de trabajo y se comunica con el kube-apiserver. Administra los contenedores y sus imágenes, garantizando que estén en el estado deseado.
- **kube-proxy:** Es un proxy de red que refleja las operaciones del servicio a los pods. También realiza el balanceo de carga entre los pods de un servicio.
- **Container Runtime:** El software que se utiliza para ejecutar contenedores. Docker es la opción más común, pero Kubernetes también es compatible con otros como containerd, cri-o, entre otros.

Componentes adicionales:

- **DNS:** Proporciona resolución de nombres para los servicios de Kubernetes.

- **Dashboard:** Interfaz web para la administración y supervisión del clúster.
- **Ingress Controller:** Administra el acceso externo a los servicios del clúster.
- **Autenticación y Autorización:** Mecanismos para autenticar usuarios y autorizar acciones en el clúster.
- **Almacenamiento:** Proveedores de almacenamiento para permitir la persistencia de datos en los pods.

Cada componente desempeña un papel crucial en el funcionamiento general del sistema.

DOCKER Y SUS COMPONENTES

Docker es una plataforma de contenedores de código abierto que facilita la creación, implementación y administración de aplicaciones en contenedores. Aquí están los componentes principales de Docker:

Docker Engine:

- Es el núcleo de Docker que ejecuta y gestiona los contenedores.
- Incluye el demonio dockerd, que se ejecuta en el host y administra los contenedores.
- Proporciona una API que permite a los usuarios interactuar con Docker de varias formas, como a través de la línea de comandos o utilizando herramientas de gestión de contenedores.

Docker Client:

- Es la interfaz de línea de comandos (CLI) que permite a los usuarios interactuar con el demonio Docker. Los usuarios emiten comandos al cliente Docker para crear, ejecutar y administrar contenedores.
- El cliente Docker se comunica con el demonio Docker a través de la API de Docker.

Imágenes de Docker:

- Las imágenes de Docker son plantillas de solo lectura que contienen el sistema de archivos y la configuración necesarios para ejecutar una aplicación en un contenedor.
- Se utilizan para crear contenedores Docker.

- Las imágenes de Docker se pueden crear desde cero utilizando un Dockerfile o pueden descargarse desde un registro de Docker como Docker Hub.

Dockerfile:

- Es un archivo de texto plano que contiene instrucciones que Docker utiliza para construir una imagen de Docker.
- Especifica los pasos necesarios para ensamblar una imagen, como qué base utilizar, qué paquetes instalar y qué comandos ejecutar.
- Proporciona un enfoque reproducible y automatizado para la construcción de imágenes de Docker.

Registro de Docker (Docker Registry):

- Es un repositorio donde se almacenan y se comparten imágenes de Docker.
- Docker Hub es el registro de Docker público predeterminado, donde los desarrolladores pueden compartir y descargar imágenes de Docker.
- Los usuarios también pueden implementar y administrar sus propios registros de Docker para almacenar imágenes privadas o personalizadas.

Docker Compose:

- Es una herramienta que permite definir y ejecutar aplicaciones Docker multicontenedor utilizando un archivo YAML para definir la configuración de la aplicación.
- Facilita la gestión de aplicaciones complejas que constan de varios contenedores Docker.

La combinación de estos componentes proporciona a los desarrolladores y administradores una plataforma flexible y potente para implementar y gestionar aplicaciones en entornos de contenedores.

Microservicios

Microservicio	# de contrato	EndPoint	Descripción
Api_Agify	1	/ agify	Se encargará de obtener información al enviársele un nombre
	2	/live	Muestra un mensaje solo para conocer el estado del servicio
API_Genderize	3	/genderize	Se encargará de obtener información al enviársele un nombre
	4	/live	Muestra un mensaje solo para conocer el estado del servicio
Gateway	5	consulta	Realiza una consulta a la API_Agify y a la API_Genderize con el parámetro de nombre
	6	/live	Muestra un mensaje solo para conocer el estado del servicio
	7	consultasimple	Hace una consulta para conocer si es posible alcanzar a las otras 2 APIS.

HISTORIA DE USUARIO

CONSULTA DE NOMBRE			
ID	1	**USUARIO DESTINO**	Usuario
PRIORIDAD	10	**RIESGO**	Bajo
ESTIMACIÓN	5 puntos	**ENCARGADO DE DESARROLLO**	Marco Antonio Xocop Roquel
El usuario por medio de Postman realizaran una consulta agregando el nombre que desean consultar			

CASO DE USO 1

Consulta de usuario:

Descripción:

El usuario por medio de Postman realizaran una consulta agregando el nombre que desean consultar

Flujo principal:

1. El usuario llena los campos (nombre que busca) para iniciar una consulta desde Postman
2. El usuario envía una consulta.
3. El usuario recibe información proveniente de los dos servicios consultados.

Flujo alternativo:

1. Si el usuario puede recibir un error de conexión

CONSULTA DE PRUEBA			
ID	1	**USUARIO DESTINO**	Técnico
PRIORIDAD	10	**RIESGO**	Bajo
ESTIMACIÓN	5 puntos	**ENCARGADO DE DESARROLLO**	Marco Antonio Xocop Roquel
El usuario rol técnico, se encargará de testear si la aplicación se encuentra en línea, para lo cual harán una consulta simple para que se les devuelva un texto como respuesta.			

CASO DE USO 2

Consulta de usuario:

Descripción:

El usuario rol técnico, se encargará de testear si la aplicación se encuentra en línea, para lo cual harán una consulta simple para que se les devuelva un texto como respuesta.

Flujo principal:

1. El usuario hará una petición desde Postman, será una petición sin parámetros.
2. El usuario envía una consulta.
3. El usuario recibe información proveniente de los dos servicios consultados.

Flujo alternativo:

1. El usuario recibirá un reporte de error

CONTRATOS

ID: 1	**NOMBRE:**	Consultar nombre
PRIORIDAD: Alta	**HISTORIA DE USUARIO:** El usuario por medio de Postman realizaran una consulta agregando el nombre que desean consultar	
ESTIMADO: 4 puntos		
MÓDULO: Agify		

CRITERIOS DE ACEPTACIÓN:

- El sistema validara que se envíe un nombre

RUTA: / agify
MÉTODO: GET
FORMATO DE ENTRADA: JSON
Params:

ATRIBUTO	TIPO	DESCRIPCIÓN
Params	Name	text

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
Name	cadena	Mensaje que contiene el nombre que se buscara.

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	DESCRIPCIÓN
400	Error en dirección de recurso
500	Error interno del servidor

PARÁMETROS DE ENTRADA	BODY:
	{
	name: “”
	}

PARÁMETROS DE SALIDA EXITOSA	HEADER:
	{
	status: 200,
	}
	BODY:
	
	"agify_data": {
	"age": #,
	"count": #,
	"name": " "
	}
	}

PARÁMETROS DE SALIDA FALLIDA	HEADER:
	{
	status: 400,
	}
	BODY:
	{
	mensaje: “”
	}

ID: 2	**NOMBRE:**	Consulta de prueba
PRIORIDAD: Alta **ESTIMADO:** 4 puntos **MÓDULO:** Agify	**HISTORIA DE USUARIO:** El usuario rol técnico, se encargará de testear si la aplicación se encuentra en línea, para lo cual harán una consulta simple para que se les devuelva un texto como respuesta.	

CRITERIOS DE ACEPTACIÓN:

- Ninguna

RUTA: /live
MÉTODO: GET
FORMATO DE ENTRADA: Ninguno

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	**TIPO**	**DESCRIPCIÓN**
Name	cadena	Mensaje que contiene el nombre que se buscara.

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	**DESCRIPCIÓN**
400	Error en dirección de recurso
500	Error interno del servidor

PARÁMETROS DE ENTRADA	Ninguno

PARÁMETROS DE SALIDA EXITOSA	HEADER:
	{
	status: 200,
	{
	"agify_data": {
	"Mensaje": "Hola mundo"
	}
	}

PARÁMETROS DE SALIDA FALLIDA	HEADER:
	{
	status: 400,
	}
	BODY:
	{
	mensaje: “”
	}

ID: 3	**NOMBRE:**	Consultar nombre
PRIORIDAD: Alta	**HISTORIA DE USUARIO:**	
ESTIMADO: 4 puntos	El usuario por medio de Postman realizaran una consulta agregando el nombre que desean consultar	
MÓDULO: Genderize		

CRITERIOS DE ACEPTACIÓN:

- El sistema validara que se envíe un nombre

RUTA: / genderize
MÉTODO: GET
FORMATO DE ENTRADA: JSON
Params:

ATRIBUTO	**TIPO**	**DESCRIPCIÓN**
Params	Name	text

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	**TIPO**	**DESCRIPCIÓN**
Name	cadena	Mensaje que contiene el nombre que se buscara.

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	**DESCRIPCIÓN**
400	Error en dirección de recurso
500	Error interno del servidor

PARÁMETROS DE ENTRADA	BODY:
	{
	name: “”
	}

PARÁMETROS DE SALIDA EXITOSA	HEADER:
	{
	status: 200,
	}
	BODY:
	{
	"genderize_data": {
	"count": “,
	"gender": " ",
	"name": " ",
	"probability": #.#
	}
	}

PARÁMETROS DE SALIDA FALLIDA	HEADER:
	{
	status: 400,
	}
	BODY:
	{
	mensaje: “”

	}
ID: 4	**NOMBRE:** Consulta de prueba
PRIORIDAD: Alta	**HISTORIA DE USUARIO:**
ESTIMADO: 4 puntos	El usuario rol técnico, se encargará de testear si la aplicación se encuentra en línea, para lo cual harán una consulta simple para que se les devuelva un texto como respuesta.
MÓDULO: Genderize	

CRITERIOS DE ACEPTACIÓN:

- Ninguna

RUTA: / live
MÉTODO: GET
FORMATO DE ENTRADA: Params
PARAMS:

ATRIBUTO	TIPO	DESCRIPCIÓN
Params	Name	text

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
Name	cadena	Mensaje que contiene el nombre que se buscara.

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	DESCRIPCIÓN
400	Error en dirección de recurso
500	Error interno del servidor

PARÁMETROS DE ENTRADA	Ninguno

PARÁMETROS DE SALIDA EXITOSA	HEADER:
	{
	status: 200,
	{
	"genderize_data": {
	"Mensaje": "Hola mundo"
	}
	}

PARÁMETROS DE SALIDA FALLIDA	HEADER:
	{
	status: 400,
	}
	BODY:
	{
	mensaje: “”
	}

ID: 5	**NOMBRE:**	Consultar nombre
PRIORIDAD: Alta	**HISTORIA DE USUARIO:**	
ESTIMADO: 5 puntos	El usuario por medio de Postman realizaran una consulta agregando el nombre que desean consultar	
MÓDULO: Gateway		

CRITERIOS DE ACEPTACIÓN:

- El sistema validara que se envíe un nombre

RUTA: / consulta
MÉTODO: GET
FORMATO DE ENTRADA: JSON
Params:

ATRIBUTO	**TIPO**	**DESCRIPCIÓN**
Params	Name	text

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	**TIPO**	**DESCRIPCIÓN**
Name	cadena	Mensaje que contiene el nombre que se buscara.

CÓDIGO DE RESPUESTA FALLIDA:

CÓDIGO	**DESCRIPCIÓN**
400	Error en dirección de recurso
500	Error interno del servidor

PARÁMETROS DE ENTRADA	BODY:
	{
	name: “”
	}

PARÁMETROS DE SALIDA EXITOSA	HEADER:
	{
	status: 200,
	}
	BODY:
	{
	"agify_data": {
	"age": #,
	"count": #,
	"name": " "
	},
	"genderize_data": {
	"count": “,
	"gender": " ",
	"name": " ",
	"probability": #.#
	}
	}

PARÁMETROS DE SALIDA FALLIDA	HEADER:
	{
	status: 400,
	}
	BODY:
	{
	mensaje: “”
	}

<table>
<tr><td>ID: 6</td><td>NOMBRE:</td><td>Consulta de prueba</td></tr>
<tr><td>PRIORIDAD: Medio</td><td colspan="2" rowspan="3">HISTORIA DE USUARIO:
El usuario rol técnico, se encargará de testear si la aplicación se encuentra en línea, para lo cual harán una consulta simple para que se les devuelva un texto como respuesta.</td></tr>
<tr><td>ESTIMADO: 4 puntos</td></tr>
<tr><td>MÓDULO: Gateway</td></tr>
<tr><td colspan="3">CRITERIOS DE ACEPTACIÓN:
• Ninguna

RUTA: / consultasimple
MÉTODO: GET
FORMATO DE ENTRADA: Ninguna</td></tr>
<tr><td colspan="3">FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:
<table>
<tr><th>ATRIBUTO</th><th>TIPO</th><th>DESCRIPCIÓN</th></tr>
<tr><td>Name</td><td>cadena</td><td>Mensaje que contiene el nombre que se buscara.</td></tr>
</table>
</td></tr>
<tr><td colspan="3">CÓDIGO DE RESPUESTA FALLIDA:
<table>
<tr><th>CÓDIGO</th><th>DESCRIPCIÓN</th></tr>
<tr><td>400</td><td>Error en dirección de recurso</td></tr>
<tr><td>500</td><td>Error interno del servidor</td></tr>
</table>
</td></tr>
<tr><td>PARÁMETROS DE ENTRADA</td><td colspan="2">Ninguno</td></tr>
<tr><td>PARÁMETROS DE SALIDA EXITOSA</td><td colspan="2">HEADER:
{
status: 200,
{
"Mensaje": "Hola mundo"
}</td></tr>
<tr><td>PARÁMETROS DE SALIDA FALLIDA</td><td colspan="2">HEADER:
{
status: 400,
}
BODY:
{
mensaje: “”
}</td></tr>
</table>

ID: 7	**NOMBRE:**	Consulta de prueba
PRIORIDAD: Alta **ESTIMADO:** 5 puntos **MÓDULO:** Gateway	**HISTORIA DE USUARIO:** El usuario rol técnico, se encargará de testear si la aplicación se encuentra en línea, para lo cual harán una consulta simple para que se les devuelva un texto como respuesta.	

CRITERIOS DE ACEPTACIÓN:

- Ninguna

RUTA: / consultasimple
MÉTODO: GET
FORMATO DE ENTRADA: Params
PARAMS:

ATRIBUTO	TIPO	DESCRIPCIÓN
Params	Name	text

FORMATO DE SALIDA: JSON
CÓDIGO DE RESPUESTA EXITOSA: HTTP 200
SALIDA:

ATRIBUTO	TIPO	DESCRIPCIÓN
Name	cadena	Mensaje que contiene el nombre que se buscara.

CÓDIGO DE RESPUESTA FALLIDA:

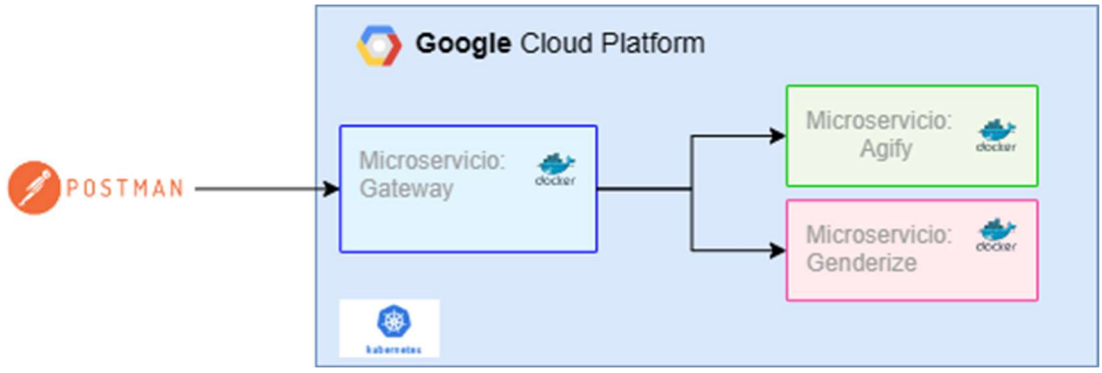
CÓDIGO	DESCRIPCIÓN
400	Error en dirección de recurso
500	Error interno del servidor

PARÁMETROS DE ENTRADA	Ninguno

PARÁMETROS DE SALIDA EXITOSA	HEADER:
	{
	status: 200,
	{
	"agify_data": {
	"Mensaje": "Hola mundo"
	},
	"genderize_data": {
	"Mensaje": "Hola mundo"
	}
	}

PARÁMETROS DE SALIDA FALLIDA	HEADER:
	{
	status: 400,
	}
	BODY:
	{
	mensaje: “”
	}

DIAGRAMA


Google Cloud Platform
POSTMAN
Microservicio:
Gateway
docker
Microservicio:
Agify
docker
Microservicio:
Genderize
docker
kubernetes