DESARROLLO ACTIVIDAD MICROSERVICIOS – KUBERNETES.

Para el desarrollo de esta actividad, primero deben hacer fork al siguiente repositorio (Recuerden que se hace desmarcando el recuadro o checkbox, pues deben copiar todas las ramas). También es necesario tener instalado y ejecutando el Docker Desktop, teniendo habilitado el motor de Kubernetes; de lo contrario, no funcionará el despliegue ni la ejecución de los servicios.

1. Una vez forkeado el repositorio, clónenlo en su equipo. Dentro del directorio donde lo clonaron, abran una sesión de Visual Studio Code (Pueden hacerlo, abriendo una terminal en el directorio y ejecutando el comando **code .**). En la sesión de Visual Studio Code, abren una terminal y crean una rama por estudiante - servicio, usando el comando ***git checkout -b 1926462-CartService,*** donde "1926462" corresponde al código de cada uno y "CartService" al servicio que se le asignó. Para saber qué servicio tiene, fíjese en la siguiente tabla:

Nombre	Servicio	Puerto	Nombre Rama
Juan Manuel	cart_service	5101	CartService
Miguel	payment_service	5102	PaymentService
Yenny	inventory_service	5103	InventoryService
Jean Paul	order_service	5104	OrderService
David	shipping_service	5105	ShippingService
Erika	billing_service	5106	BillingService
Alejandro	notification_service	5107	NotificationService
Juan Pablo	loyalty_service	5108	LoyaltyService

2. Una vez creada la rama, dentro de la carpeta "services" del repositorio, crean una carpeta con el nombre "1926462_cart_service", donde cambian su código y nombre del servicio según su caso. Luego, se accede al siguiente directorio de Google Drive, donde hay dos carpetas, una de servicios y otra para el despliegue en Kubernetes. Ingresan a la carpeta de servicios y buscan el directorio correspondiente a su servicio, descargan su contenido y lo guardan en la carpeta creada anteriormente dentro del repositorio (Procuren estudiar el contenido de las funciones definidas aquí, porque es probable que pregunten cuáles son las solicitudes que se le hacen a cada servicio. Es más, si quieren, cambian los nombres de las solicitudes, pues son las mismas en cada servicio, pero eso implicaría hacer cambios dentro del orquestador – Ustedes se harían cargo de eso).

3. El siguiente paso consiste en crear el yaml de despliegue con Kubernetes, para esto, ingresamos a la carpeta de “k8s” dentro del directorio de Drive que les mencioné antes y buscan el archivo relacionado a su servicio para copiarlo dentro de la carpeta “k8s” del repositorio.

4. Lo siguiente que deben hacer, es abrir el archivo orchestrator/app.py del repositorio, dentro, en la línea 6 aproximadamente, hay unos corchetes donde deben colocar la información relacionada a su servicio:

```
SERVICES = [
  {"name":"cart","url": os.getenv("CART_URL","http://cart-service:5101")},
  {"name":"payment","url": os.getenv("PAYMENT_URL","http://payment-service:5102")},
  {"name":"inventory","url": os.getenv("INVENTORY_URL","http://inventory-service:5103")},
  {"name":"order","url": os.getenv("ORDER_URL","http://order-service:5104")},
  {"name":"shipping","url": os.getenv("SHIPPING_URL","http://shipping-service:5105")},
  {"name":"billing","url": os.getenv("BILLING_URL","http://billing-service:5106")},
  {"name":"notification","url": os.getenv("NOTIF_URL","http://notification-service:5107")},
  {"name":"loyalty","url": os.getenv("LOYALTY_URL","http://loyalty-service:5108")},
]
```

Agreguen solo su servicio. En caso de que haya desorden, procuren mantenerlo al momento de agregar el servicio (Tampoco olviden las comas).

5. Luego de modificar el app.py del orquestador, se abre el archivo docker-compose.yml del repositorio, en él pondrán la información de su servicio para que sea ejecutado tal y como aparece a continuación:

```
cart_service:
 build: ./services/1926462_cart_service
 ports: ["5101:5101"]
 environment:
  - SERVICE_NAME=cart_service
  - PORT=5101
  - FAIL_RATE=0.0
```

Nótese que lo que está marcado en amarillo debe cambiarse según su caso (Miren la tabla de arriba), el “1926462_cart_service” corresponde al directorio donde guardaron su servicio en el punto 2. **IMPORTANTE:** Para el servicio payment_service (Miguel), el FAIL_RATE debe ser del 0.4, no del 0.0. Al igual que en el anterior caso, mantengan el orden, si el servicio del 5103 se actualizó

primero que el 5102, esperaría que al momento que la persona encargada de hacer este último, lo organice y lo coloque en orden dentro de los archivos. Pueden revisar el archivo de docker-compose.yml que está en la carpeta de drive compartida anteriormente, pero deben modificarlo.

6. Con lo anterior, casi se han terminado las modificaciones. Lo siguiente es probar que todo esté funcionando correctamente antes de continuar con ellas:

   - Primero, asegúrense de tener abierto y ejecutando el Docker Desktop, con el motor de Kubernetes corriendo. Luego, dentro de la terminal del Visual Studio Code, y ubicándonos en la raíz del repositorio, usamos el comando ***kubectl get nodes*** para corroborar que efectivamente se está ejecutando el motor de Kubernetes (Debería aparecer un nodo tipo "docker-desktop".

   - Luego se usa el comando ***docker compose up --build*** para levantar los contenedores que haya en el momento (Si ya hay 3 servicios en el repositorio, entonces se levantarán 4 contenedores – Una más por el orquestador). Se pueden revisar los contenedores levantados con ***docker ps*** (Hay qué abrir otro terminal).

   - El siguiente paso consiste en crear el namespace ejecutando el comando ***kubectl apply -f k8s/namespace.yaml*** .

   - Ahora hay qué hacer una modificación dentro del yml hecho en la carpeta k8s del repositorio. Primero se usa el comando ***docker images*** para listar las imágenes que hay en el equipo de Docker y luego hay qué buscar la correspondiente al servicio, en mi caso se llama saga-ecommerce-cart_service (Probablemente las de ustedes se llamen de forma similar). Ahora, en el archivo de su servicio, digamos cart_service_deploy.yaml, en la línea 18 (Aproximadamente), donde se asigna la imagen, hay qué cambiar el contenido que tiene (REPLACE_IMAGE_cart_service), por el nombre de la imagen que acabamos de encontrar (saga-ecommerce-cart_service).

   - Lo siguiente es cambiar dentro de esos archivos, aquellos nombres del servicio donde hay barra baja; es decir, donde haya "cart_service", hay qué cambiarlo a "cart-service" (No aplica para la imagen).

- Finalmente, se aplican todos los pods con el comando ***kubectl apply -f k8s/*** . Con el comando ***kubectl get pods -n saga-ecommerce*** debería verse el estado de los pods que se aplicaron anteriormente (Deberían estar corriendo).

7. El siguiente paso es forwardear el orquestador para probar que los servicios se estén comunicando. Para esto se usa el comando ***kubectl port-forward svc/orchestrator 5000:5000 -n saga-ecommerce*** . Luego también hay qué forwardear el servicio con el comando ***kubectl port-forward svc/cart-service 5101:5101 -n saga-ecommerce*** (En otra terminal y adaptando a su servicio).

- Para revisar que el servicio está funcionando correctamente, se usa el comando ***curl http://localhost:5101/health*** , cambiando el puerto por el de su servicio (Debería aparecer algo como {"service":"cart_service","status":"ok"}. Con esto ya tendríamos el servicio arriba y funcionando.

- Para probar el endpoint do, se hace:

```
curl -X POST http://localhost:5101/do \
  -H "Content-Type: application/json" \
  -d '{"order_id":"orden-test-001","user":"juan","items":[{"sku":"ABC","qty":2}]}'
```

  Y debería aparecer algo como: {"service":"cart_service","status":"ok"}. Tenga en cuenta que esto cambia según el servicio y los endpoints establecidos.

- Para probar el endpoint compensate, se hace:

```
curl -X POST http://localhost:5101/compensate \
  -H "Content-Type: application/json" \
  -d '{"order_id":"orden-test-001"}'
```

  Y debería aparecer algo como: {"service":"cart_service","status":"compensated"}. Tenga en cuenta que esto cambia según el servicio y los endpoints establecidos.

8. A continuación, se prueba desde el orquestador, haciendo:

```
curl -X POST http://localhost:5000/checkout \
```

***-H "Content-Type: application/json" ***
-d '{"user":"juan","items":[{"sku":"ABC123","qty":1}],"order_id":"orden-101"}'

Debe dar algo como: {"order_id":"orden-101","status":"success"}. Esta prueba si se puede realizar con todos los servicios. En el caso de payment_service (Miguel), se debe probar varias veces, pues en alguna debe dar error. Deberían usar este chat para pedir casos de prueba para su servicio.

9. Si hasta aquí todo está bien, se avanza con subir los cambios a su repositorio remoto haciendo ***git add .*** Luego, usen el comando ***git commit -m "Juan Perea - CartService"*** (Cambiando por sus nombres y sus servicios). Finalmente, con ***git push origin 1926462-CartService*** (Cambiando por el nombre de sus ramas) se suben los cambios.
10. Lo último es hacer el PR, manteniendo en el título y la descripción el nombre del commit, es decir ***Juan Perea - CartService*** (Cambiando por sus nombres y servicios).

RECUERDA USAR:

docker compose down
kubectl delete all --all -n saga-ecommerce
kubectl delete namespace saga-ecommerce

PARA PARAR Y ELIMINAR LOS PROCESOS.