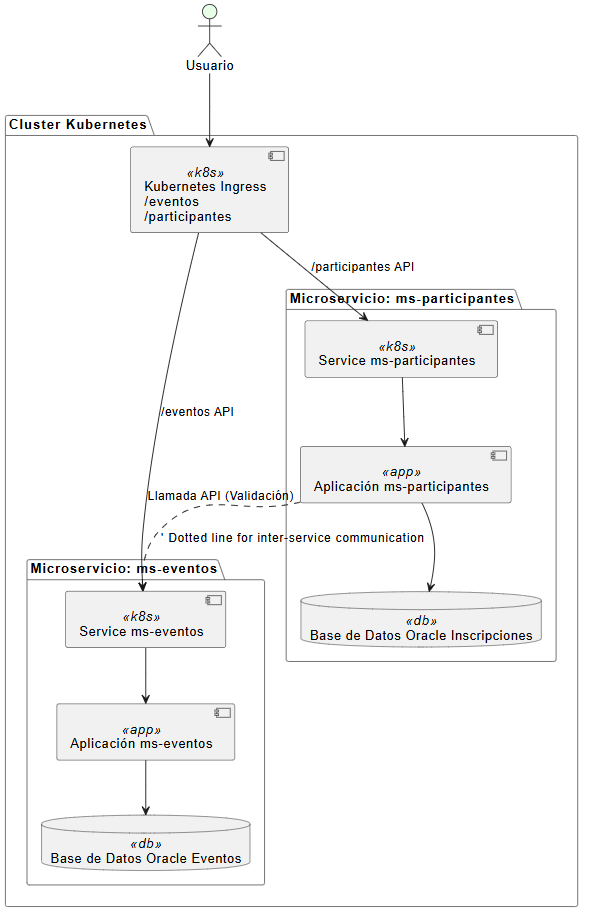
**CURSO Docker y Kubernetes Intermedio | SUNAT**

Integrantes:

* Gerardo Benigno Rondo Gutierrez
* German Alessandro Granados Palomino
* Juan Antonio Balboa Rojas
* Oliver Adrián Sanchez Ascorbe

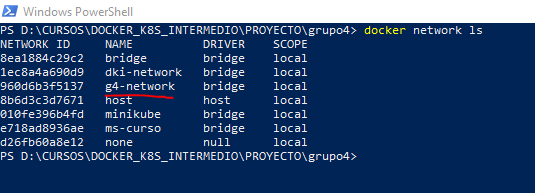
**Diagrama de arquitectura del proyecto**



Pasos a seguir:

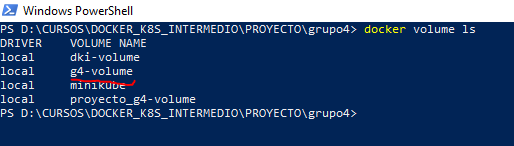
1. Crear network

docker network create --driver bridge g4-network



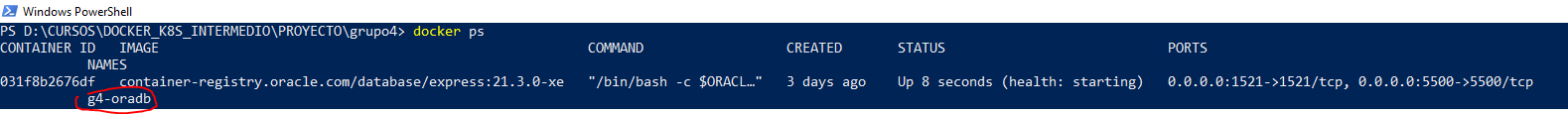
1. Crear volumen

docker volume create g4-volume



1. Crear el contenedor Oracle

docker run -d --name g4-oradb -e ORACLE\_PWD=Netec\_123 --network g4-network --volume g4-volume:/opt/oracle/oradata -p 1521:1521 -p 5500:5500 container-registry.oracle.com/database/express:21.3.0-xe



**Datos de la conexión**

Usuario: sys

Rol: sysdba

Contraseña: Netec\_123

Nombre del Host: localhost

Puerto: 1521

SID: xe

1. Crear usuario de BD

Ejecutar lo siguiente:

**ALTER** **SESSION** **SET** CONTAINER = XEPDB1;

**CREATE** **USER** g4user **IDENTIFIED** **BY** g4password

**DEFAULT** **TABLESPACE** users

**TEMPORARY** **TABLESPACE** temp;

**GRANT** **CONNECT**, RESOURCE **TO** g4user;

**ALTER** **USER** g4user QUOTA UNLIMITED **ON** users;

1. Creación de tablas:

Este punto no es estrictamente necesario, ya que la creación de objetos se realiza al levantar los microservicio.

-- G4USER.EVENTO definition

**CREATE** **TABLE** "G4USER"."EVENTO"

( "ID" **NUMBER**(19,0) GENERATED **BY** **DEFAULT** **AS** **IDENTITY** **MINVALUE** 1 **MAXVALUE** 9999999999999999999999999999 **INCREMENT** **BY** 1 **START** **WITH** 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE **NOT** **NULL** ENABLE,

**"CAPACIDAD"** **NUMBER**(10,0) **NOT** **NULL** ENABLE,

**"DESCRIPCION"** **VARCHAR2**(255 **CHAR**),

**"FECHA"** **TIMESTAMP** (6) **NOT** **NULL** ENABLE,

**"NOMBRE"** **VARCHAR2**(255 **CHAR**) **NOT** **NULL** ENABLE,

**"UBICACION"** **VARCHAR2**(255 **CHAR**),

**PRIMARY** **KEY** (**"ID"**)

**USING** **INDEX** **PCTFREE** 10 INITRANS 2 MAXTRANS 255 **COMPUTE** **STATISTICS**

STORAGE(**INITIAL** 65536 **NEXT** 1048576 MINEXTENTS 1 **MAXEXTENTS** 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL **DEFAULT** FLASH\_CACHE **DEFAULT** CELL\_FLASH\_CACHE **DEFAULT**)

**TABLESPACE** **"USERS"** ENABLE,

**CONSTRAINT** **"UK3GNB3B0HCJYISTT3JEW62CX7X"** **UNIQUE** (**"NOMBRE"**)

**USING** **INDEX** **PCTFREE** 10 INITRANS 2 MAXTRANS 255 **COMPUTE** **STATISTICS**

STORAGE(**INITIAL** 65536 **NEXT** 1048576 MINEXTENTS 1 **MAXEXTENTS** 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL **DEFAULT** FLASH\_CACHE **DEFAULT** CELL\_FLASH\_CACHE **DEFAULT**)

**TABLESPACE** **"USERS"** ENABLE

) SEGMENT CREATION **IMMEDIATE**

**PCTFREE** 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

**NOCOMPRESS** LOGGING

STORAGE(**INITIAL** 65536 **NEXT** 1048576 MINEXTENTS 1 **MAXEXTENTS** 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL **DEFAULT** FLASH\_CACHE **DEFAULT** CELL\_FLASH\_CACHE **DEFAULT**)

**TABLESPACE** **"USERS"**;

-- G4USER.PARTICIPANTE definition

**CREATE** **TABLE** "G4USER"."PARTICIPANTE"

( "ID" **NUMBER**(19,0) GENERATED **BY** **DEFAULT** **AS** **IDENTITY** **MINVALUE** 1 **MAXVALUE** 9999999999999999999999999999 **INCREMENT** **BY** 1 **START** **WITH** 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE **NOT** **NULL** ENABLE,

**"ID\_EVENTO"** **NUMBER**(19,0) **NOT** **NULL** ENABLE,

**"ID\_PARTICIPANTE"** **NUMBER**(19,0) **NOT** **NULL** ENABLE,

**PRIMARY** **KEY** (**"ID"**)

**USING** **INDEX** **PCTFREE** 10 INITRANS 2 MAXTRANS 255 **COMPUTE** **STATISTICS**

STORAGE(**INITIAL** 65536 **NEXT** 1048576 MINEXTENTS 1 **MAXEXTENTS** 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL **DEFAULT** FLASH\_CACHE **DEFAULT** CELL\_FLASH\_CACHE **DEFAULT**)

**TABLESPACE** **"USERS"** ENABLE

) SEGMENT CREATION **IMMEDIATE**

**PCTFREE** 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

**NOCOMPRESS** LOGGING

STORAGE(**INITIAL** 65536 **NEXT** 1048576 MINEXTENTS 1 **MAXEXTENTS** 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL **DEFAULT** FLASH\_CACHE **DEFAULT** CELL\_FLASH\_CACHE **DEFAULT**)

**TABLESPACE** **"USERS"**;

1. Descargar proyectos y repositorios:

**Proyectos principales:**

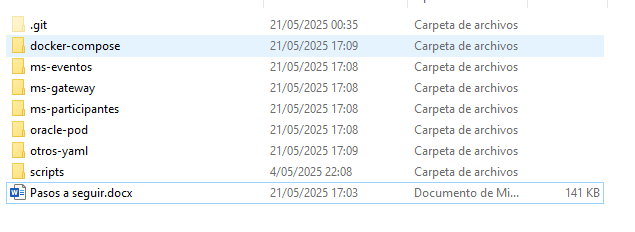
<https://github.com/jbalboar/ms-eventos>

<https://github.com/jbalboar/ms-participantes>

**Proyecto para Spring Cloud Gateway:**

<https://github.com/jbalboar/ms-gateway>

**Proyecto donde se encuentra los distintos yamls y scripts de BD:**



1. Compilar y generar jar

Ubicarse en la ruta de cada proyecto y ejecutar:

.\mvnw clean install -Dmaven.test.skip=true

1. Crear la imagen de ambos proyectos

Ubicarse en la ruta de cada proyecto y ejecutar:

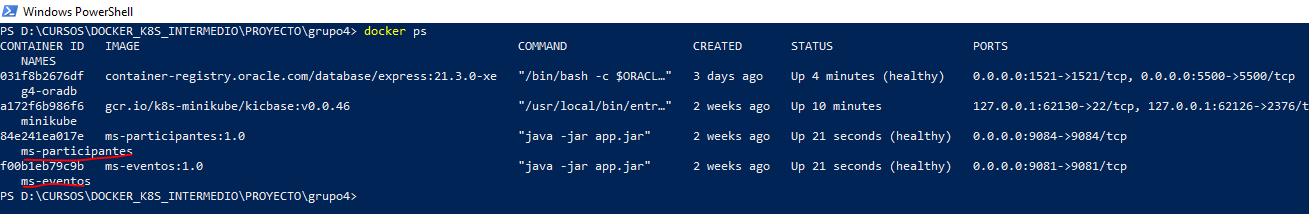
docker build -t ms-eventos:1.0 .

docker build -t ms-participantes:1.0 .

1. Crear y levantar los contenedores

docker run -d --name ms-eventos --network g4-network -p 9081:9081 ms-eventos:1.0

docker run -d --name ms-participantes --network g4-network -p 9081:9081 ms-participantes:1.0



1. Para docker-compose, asegurarse de lo siguiente:

* Detener y eliminar los contenedores:

ms-participantes

ms-eventos

g4-oradb

1. Crear el archivo docker-compose-yml con el siguiente contenido:

<https://github.com/jbalboar/grupo4/tree/main/docker-compose>

services:

  g4-oradb:

    container\_name: g4-oradb

    image: container-registry.oracle.com/database/express:21.3.0-xe

    environment:

      ORACLE\_PWD: g4password

      ORACLE\_SID: XE

      ORACLE\_PDB: XEPDB1

      ORACLE\_CHARACTERSET: AL32UTF8

    ports:

      - "1521:1521"

      - "5500:5500"

    volumes:

      - g4-volume:/opt/oracle/oradata

      - ./scripts:/opt/oracle/scripts/startup

    networks:

      - g4-network

    healthcheck:

      test: ["CMD", "sh", "-c", "echo 'SELECT 1 FROM DUAL;' | sqlplus system/Netec\_123@localhost:1521/XE"]

      interval: 30s

      timeout: 10s

      retries: 5

      start\_period: 60s

  ms-eventos:

    image: ms-eventos:1.0

    container\_name: ms-eventos

    ports:

      - "9081:9081"

    environment:

      USER\_DEMO: Netec

    networks:

      - g4-network

    depends\_on:

      g4-oradb:

        condition: service\_healthy

    healthcheck:

      test: ["CMD", "curl", "-f", "http://127.0.0.1:9081/eventos"]

      interval: 30s

      timeout: 15s

      retries: 5

      start\_period: 30s

  ms-participantes:

    image: ms-participantes:1.0

    container\_name: ms-participantes

    ports:

      - "9084:9084"

    environment:

      MS\_EVENTOS\_URL: http://ms-eventos:9081/eventos

    networks:

      - g4-network

    depends\_on:

      ms-eventos:

        condition: service\_healthy

    healthcheck:

      test: ["CMD", "curl", "-f", "http://127.0.0.1:9084/participantes"]

      interval: 30s

      timeout: 15s

      retries: 5

      start\_period: 30s

volumes:

  g4-volume:

networks:

  g4-network:

    external: true

1. Crear la carpeta “scripts” en la misma ruta donde se ha creado el archivo “docker-compose.yml”. Dentro de la carpeta script crear el archivo “init\_schema.sql” con el siguiente contenido:

ALTER SESSION SET CONTAINER = XEPDB1;

CREATE USER g4user IDENTIFIED BY g4password

DEFAULT TABLESPACE users

TEMPORARY TABLESPACE temp;

GRANT CONNECT, RESOURCE TO g4user;

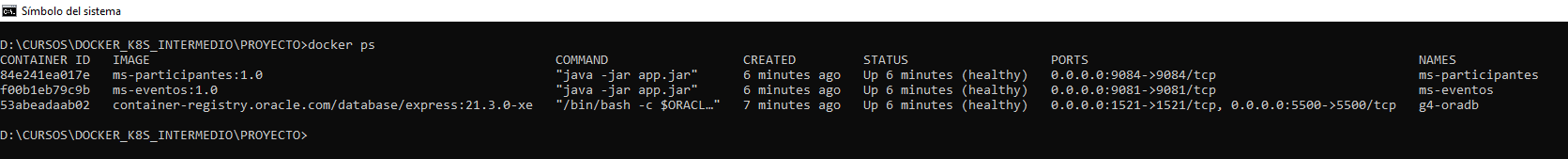
ALTER USER g4user QUOTA UNLIMITED ON users;

ALTER USER g4user QUOTA UNLIMITED ON users;

1. Ejecutar el comando:

docker-compose up -d

1. Verificar que los contenedores se encuentren levantados



1. Pasos para Kubernetes.

**Nota:** Debido a disponibilidad de la máquina virtual, se decidió utiliza minikube instalado localmente.

1. Primero se instala el pod de Oracle

Para ello se ha generado el “oracle-pod.yaml”

<https://github.com/jbalboar/grupo4/tree/main/oracle-pod>

apiVersion: v1

kind: Pod

metadata:

  name: oracle-xe

  labels:

    app: oracle-xe

spec:

  containers:

    - name: oracle-xe

      image: gvenzl/oracle-xe:18.4.0-slim

      ports:

        - containerPort: 1521

      env:

        - name: ORACLE\_PASSWORD

          value: admin

      volumeMounts:

        - mountPath: /opt/oracle/oradata

          name: oracle-data

  volumes:

    - name: oracle-data

      emptyDir: {}

y su service “oracle-service.yaml”:

apiVersion: v1

kind: Service

metadata:

  name: g4-oradb

spec:

  selector:

    app: oracle-xe

  ports:

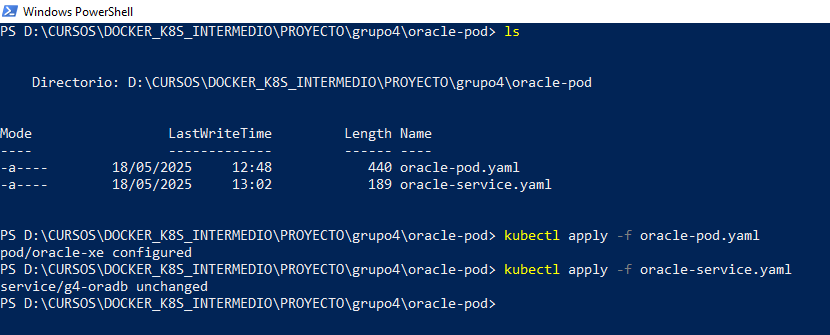
    - name: oracle

      port: 1521

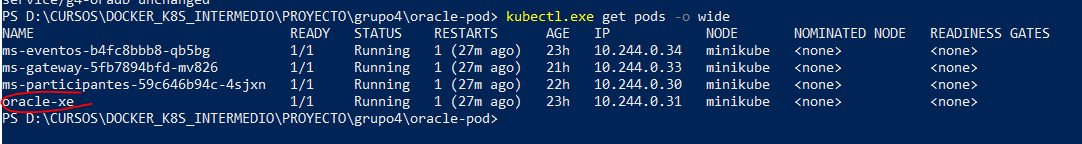
      targetPort: 1521

  type: ClusterIP

Se ejecutan ambos yamls:

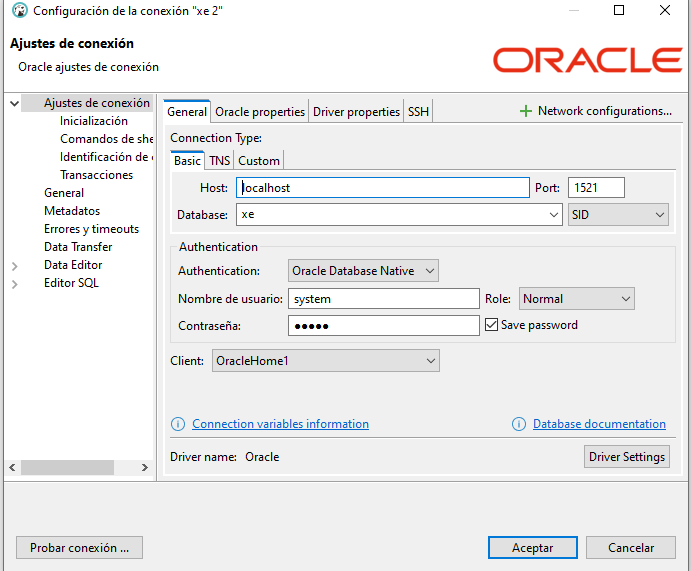


Y luego se verifica el pod creado:



Nos conectamos a la BD:

Clave: admin



Y ejecutamos los scripts de BD:

**ALTER** **SESSION** **SET** CONTAINER = XEPDB1;

**CREATE** **USER** g4user **IDENTIFIED** **BY** g4password

**DEFAULT** **TABLESPACE** users

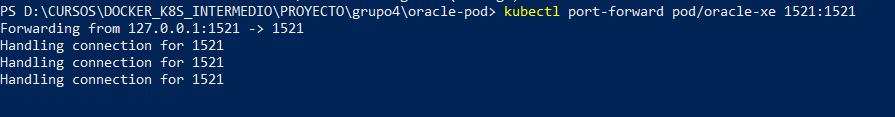
**TEMPORARY** **TABLESPACE** temp;

**GRANT** **CONNECT**, RESOURCE **TO** g4user;

**ALTER** **USER** g4user QUOTA UNLIMITED **ON** users;

Nota: En minikube se debe ejecutar el siguiente comando para exponer la BD de forma local desde el pod:

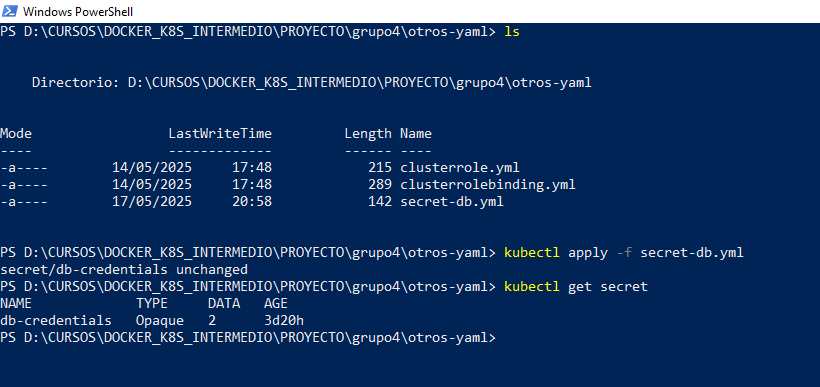
kubectl port-forward pod/oracle-xe 1521:1521



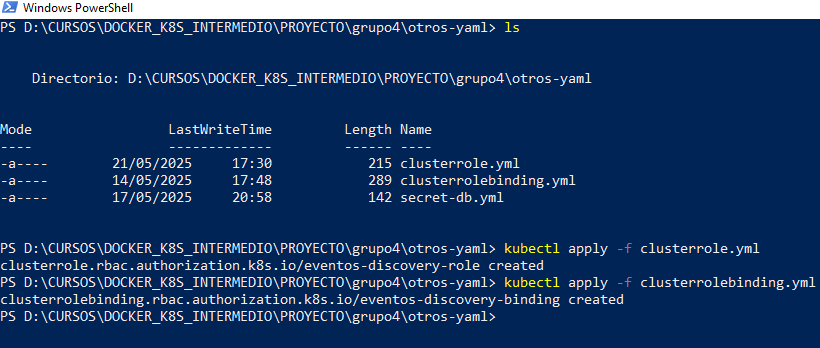
1. Luego procedemos a levantar los yaml en común para ambos micros.

Primero creamos el secret “secret-db.yml”

<https://github.com/jbalboar/grupo4/tree/main/otros-yaml>



Ahora ejecutamos el ClusterRole y ClusterRoleBinding



1. Ahora procederemos a levantar el ms-eventos

Descargar los yaml del git

<https://github.com/jbalboar/grupo4/tree/main/ms-eventos>

Y ejecutarlo en el orden siguiente:

configmap-ms-eventos.yaml

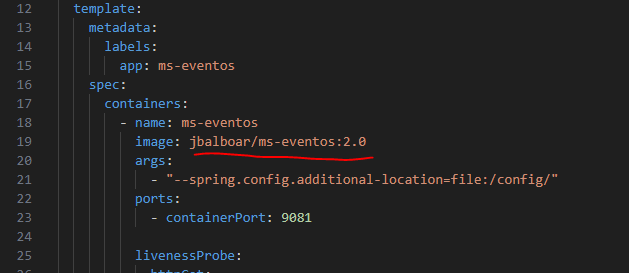
ms-eventos-ingres.yaml

service-ms-eventos.yaml

ms-eventos-deployment.yml

En el archivo ms-eventos-deployment.yml se debe consignar la versión subida al docker hub

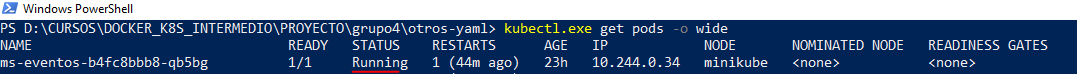
<https://hub.docker.com/repository/docker/jbalboar/ms-eventos/tags/2.0/sha256-2934fd6ded29110041453ce1ed89d670710f00a7fded36a7c11bb501313b3eaa>



Para ejecutar cada archivo se hace con:

kubectl apply -f <el archivo yaml>

Luego de la ejecución el pod debe estar en estado Running



1. Ahora procederemos a levantar el ms-participantes

Descargar los yaml del git

<https://github.com/jbalboar/grupo4/tree/main/ms-participantes>

Y ejecutarlo en el orden siguiente:

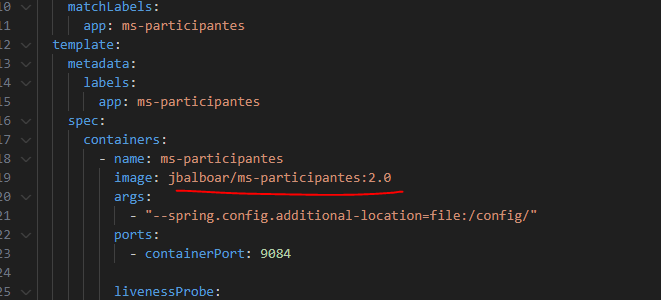
configmap-ms-participantes.yaml

service-ms-participantes.yaml

ms-participantes-deployment.yml

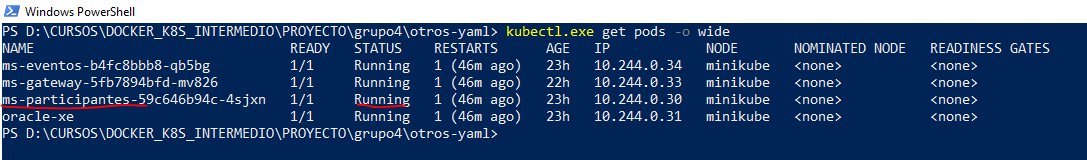
En el archivo ms-participantes-deployment.yml se debe consignar la versión subida al docker hub

<https://hub.docker.com/repository/docker/jbalboar/ms-participantes/tags/2.0/sha256-501aa767832f223072711e56ed3ee89ea13ef74573f1f91b103788295583ad57>



kubectl apply -f <el archivo yaml>

Luego de la ejecución el pod debe estar en estado Running



1. Ahora procederemos a levantar el ms-gateway

Descargar los yaml del git

<https://github.com/jbalboar/grupo4/tree/main/ms-gateway>

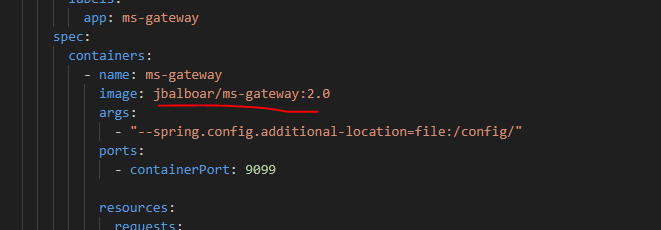
Y ejecutarlo en el orden siguiente:

configmap-ms-gateway.yaml

service-ms-gateway.yaml

ms-gateway-deployment.yml

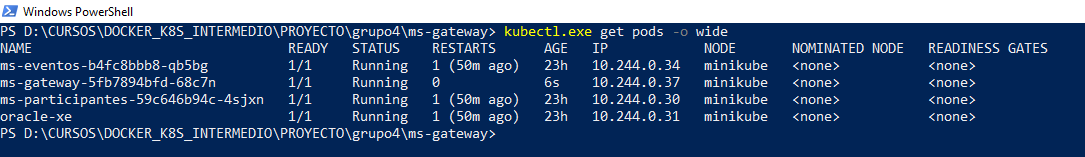
En el archivo ms-gateway-deployment.yml se debe consignar la versión subida al docker hub



<https://hub.docker.com/repository/docker/jbalboar/ms-gateway/tags/2.0/sha256-b049feee3194161760aaadc74481345a43f19f668747f221c6c45e6cc0991702>

kubectl apply -f <el archivo yaml>

Luego de la ejecución el pod debe estar en estado Running



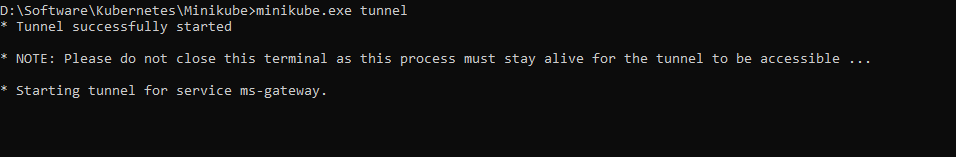
Considerar que ambos los microservicio de eventos y participantes saldrán por el puerto 9099 (según la configuración del Service)



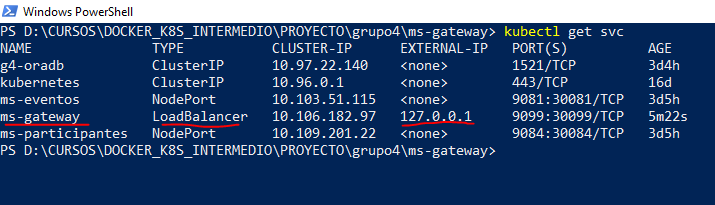
1. Probando los microservicios

En minikube, se debe ejecutar lo siguiente para exponer la IP del LoadBalance:

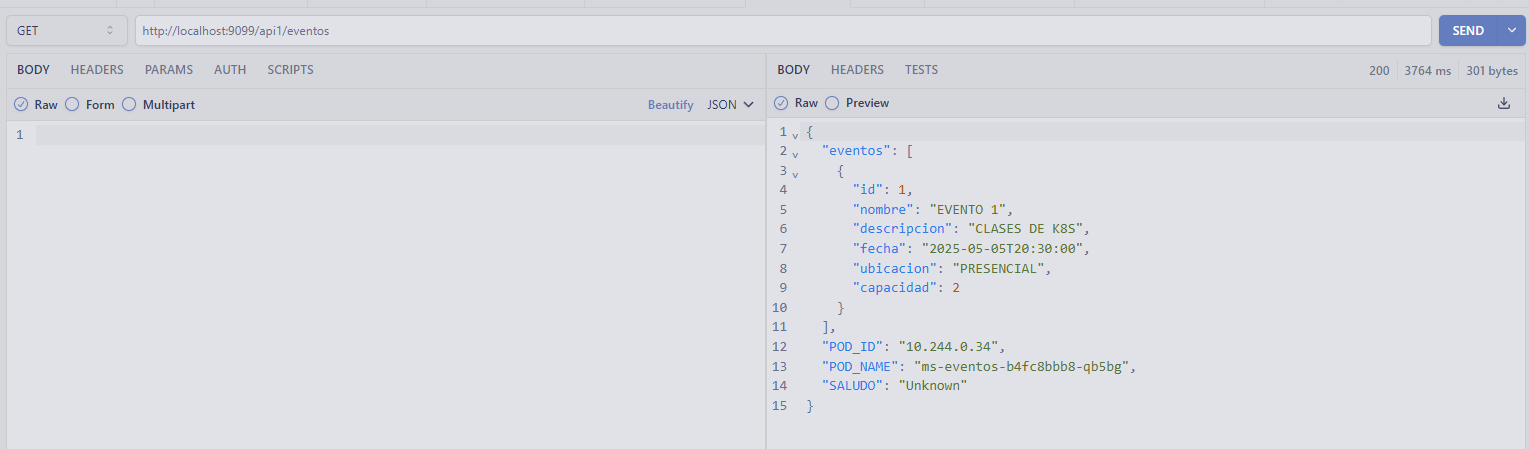
minikube tunnel



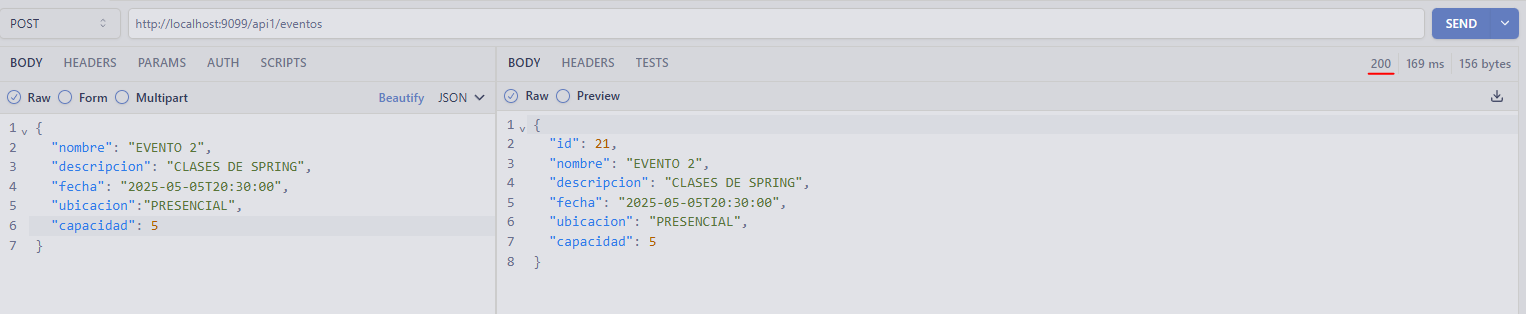
Con esto vemos que se expone el LoadBalancer en el puerto 9099 del localhost



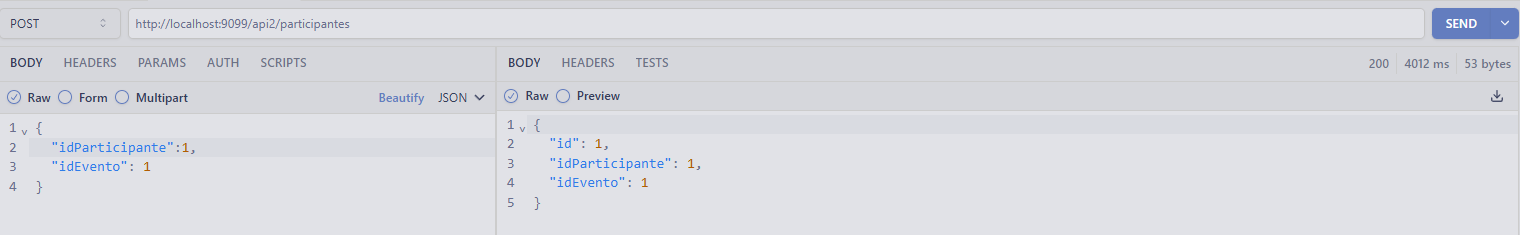
Llamada a GET /api1/eventos:



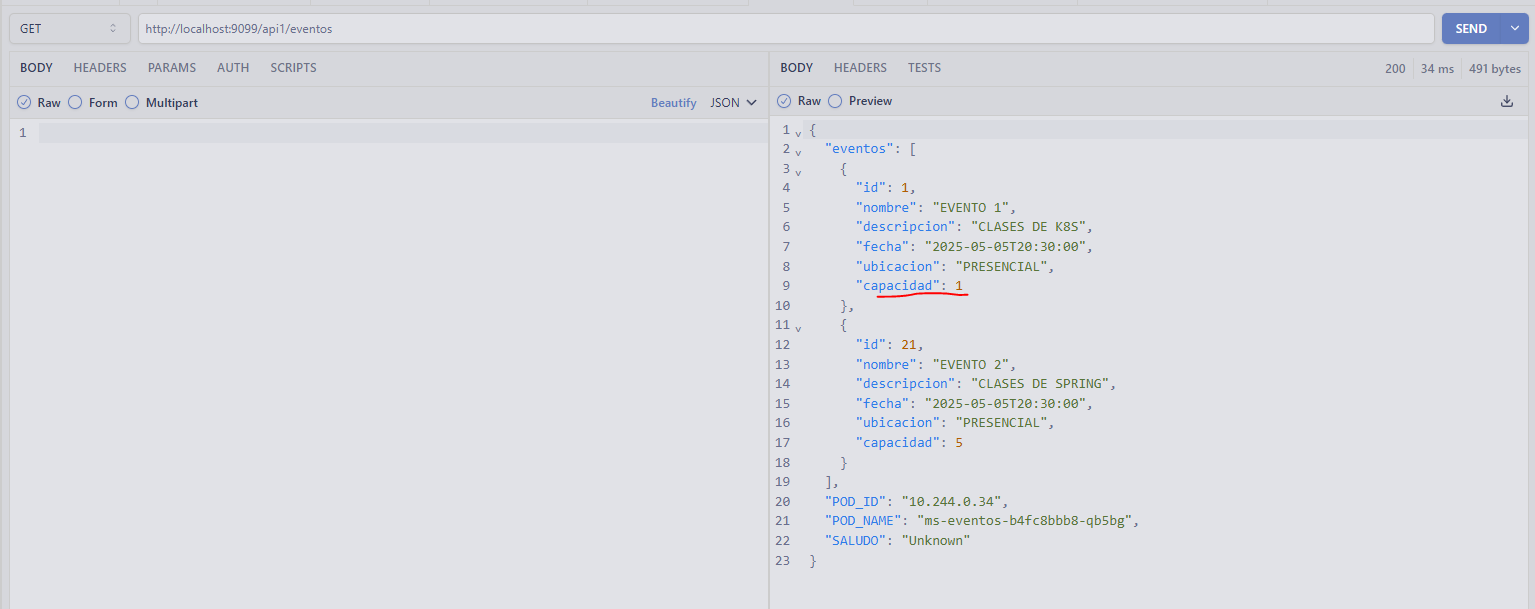
Llamada a POST /api1/eventos:



Guardamos el participante “1” al evento “1” con el POST /api2/participantes



Cuando volvemos a obtener la lista de evento, observamos que la capacidad disminuyó



Cuando consultamos la lista de participantes con el GET /api2/participantes

Observamos que se muestra el participante “1” registrado en el evento “1”

