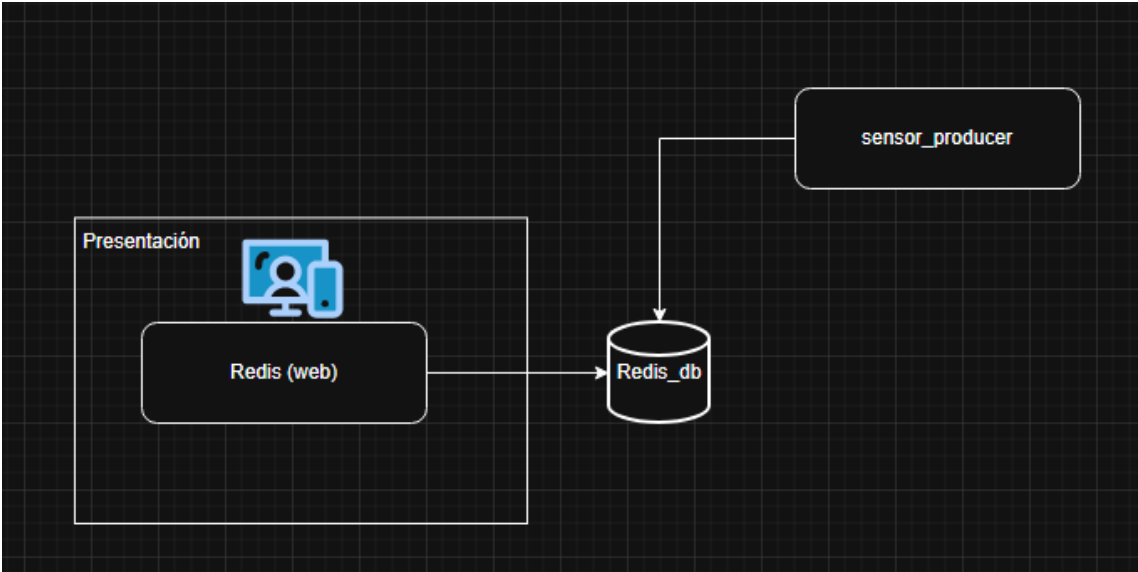


Informe de examen

Realizado por: Andrés Encalada

1. Arquitectura



La arquitectura se basa en un microservicio catalogado como productor, el cual actúa como un sensor lot que envía datos continuamente a la base de datos Redis, posteriormente la interfaz web de redis muestra los datos leídos de la base.

2. Tecnologías utilizadas

	Tecnología	Razón
Base de datos	Redis (redis:alpine)	Elegido por su rapidez, se le configuró persistencia mediante un volumen de datos.
Productor	Python 3.9 alpine	Es una imagen ligera, el script se le inyectó con ConfigMap
Cliente	Redis commander	Se usó esta opción por su compatibilidad con redis

3. Configuración de redis Commander

Llaves configuradas para no requerir configuración

```
- name: REDIS_HOST
  value: "redis-service"
- name: REDIS_PORT
  value: "6379"
- name: REDIS_DB
  value: "0"
- name: REDIS_PASSWORD
  valueFrom:
    secretKeyRef:
      name: redis-secret
      key: redis-password
```

Configuración de acceso (NodePort)

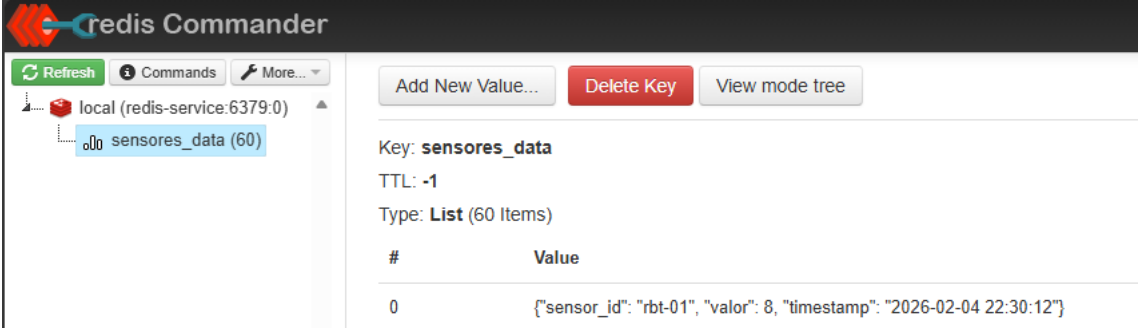
```
apiVersion: v1
kind: Service
metadata:
  name: visor-service
spec:
  type: NodePort
  selector:
    app: redis-commander
  ports:
    - port: 80
      targetPort: 8081
      nodePort: 30050
```

4. Manifiestos de Kubernetes

https://github.com/AndresEncalada/EvaluacionEncaladaA_Kubernetes.git

5. Evidencia de funcionamiento

60 items al iniciar



The screenshot shows the Redis Commander web interface. On the left, a tree view shows a local Redis instance (redis-service:6379:0) with a key named 'sensores_data' containing 60 items. On the right, the details for the 'sensores_data' key are displayed, showing a TTL of -1 and a type of List (60 Items). A table below shows the first item in the list:

#	Value
0	{"sensor_id": "rbt-01", "valor": 8, "timestamp": "2026-02-04 22:30:12"}

Ahora existen 65 items

Redis Commander interface showing a Redis key **sensores_data** with 65 items in a List type. The TTL is -1. The interface includes buttons for Refresh, Commands, More..., Add New Value..., Delete Key, and View mode tree.

#	Value
0	{ "sensor_id": "rbt-01", "valor": 49, "timestamp": "2026-02-04 22:30:27" }

PersistenceVolumeClaim en estado Bound

```
C:\Users\Estudiante\Desktop\Nueva carpeta\examen>kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS	AGE
redis-pvc	Bound	pvc-dd96131d-16bb-47bc-bf54-12e0c0f81f2	1Gi	RWO	standard	<unset>	18m

Borrado y levantamiento automático del pod

```
C:\Users\Estudiante\Desktop\Nueva carpeta>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
redis-commander-7fb4d594c7-q5vcm    1/1     Running   0           3m6s
redis-db-f4559fbf8-qflks            1/1     Running   0           3m6s
sensor-producer-5dc6c49649-xnc57    1/1     Running   0           3m6s

C:\Users\Estudiante\Desktop\Nueva carpeta>kubectl delete pod redis-db-f4559fbf8-qflks
pod "redis-db-f4559fbf8-qflks" deleted from default namespace

C:\Users\Estudiante\Desktop\Nueva carpeta>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
redis-commander-7fb4d594c7-q5vcm    1/1     Running   0           3m37s
redis-db-f4559fbf8-s49js            1/1     Running   0           10s
sensor-producer-5dc6c49649-xnc57    1/1     Running   0           3m37s

C:\Users\Estudiante\Desktop\Nueva carpeta>
```

Items después del borrado del pod

Redis Commander interface showing a Redis key **sensores_data** with 90 items in a List type. The TTL is -1. The interface includes buttons for Refresh, Commands, More..., Add New Value..., Delete Key, and View mode tree.

#	Value
0	{ "sensor_id": "rbt-01", "valor": 97, "timestamp": "2026-02-04 22:31:42" }