

Reporte de Proyecto 1 - versión 2

Módulo Docker-Kubernetes

DATAHACK

Profesores:

Docker / Edwin Delgado Paredes

Kubernetes / Cristabel Talavera

Estudiante: Inti Luna Avilés

Objetivos.....	3
Diseño.....	3
Datos y reporte en repositorio:.....	5
Creacion Imagen.....	5
Mongodb.....	5
NET.....	5
Creación entorno local.....	9
Configuración.....	9
Pruebas.....	11
Creación entorno Kubernetes.....	14
Configuración.....	14
Namespace.....	14
Secretos.....	15
Volumenes.....	15
Realizar Deployment.....	16
Pruebas.....	18
Variables de entorno se pasan correctamente.....	18
Prueba de persistencia.....	19
Correr proyecto.....	21
Problemas encontrados.....	23
Recursos.....	24
Conceptos claves.....	24
Enlaces.....	24

Objetivos

Crear entorno local con docker para crear una aplicación .NET que se conecta a una base de datos mongodb.

Crear entorno kubernetes para crear deployments con aplicación .NET que conecta una base de datos mongodb.

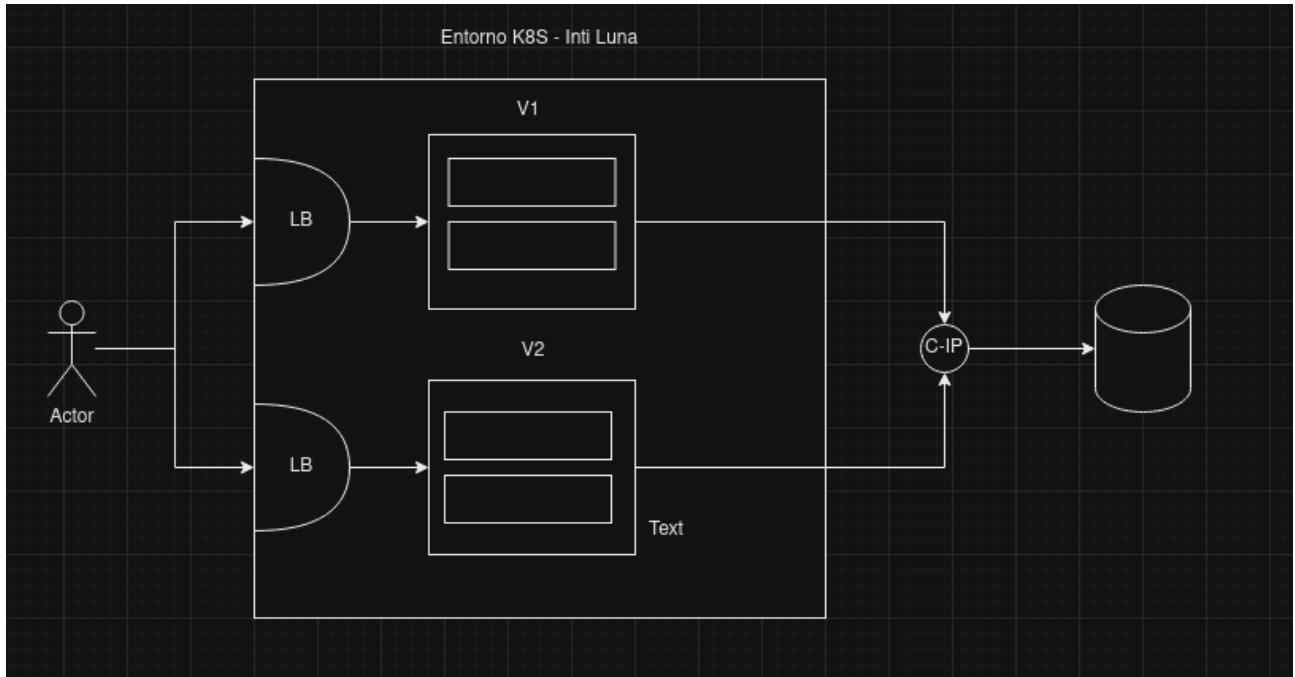
Diseño

Entorno local:

Se tiene una base de datos y dos versiones de front-end Netapi usando las imágenes correspondientes.

Entorno Kubernetes:

Se tiene una base de datos y dos versiones de front-end. Cada versión de front-end está en un deployment diferente y tiene dos réplicas. La conexión hacia el front-end están manejadas por un balanceador de carga y la conexión a mongodb es del tipo Cluster-IP de manera que solo se puede conectar internamente. Esta configuración está bien para la lectura pero requiere mayor investigación y trabajo para optimizar la escritura desde múltiples front-end. Se muestra un diagrama a manera de resumen:



Datos y reporte en repositorio:

<https://github.com/intiluna/docker-kubernetes-practica>

Creacion Imagen

Mongodb

Fichero dockerfile

```
-----  
# From image for Linux OS  
FROM mongo:4.0.7-xenial  
# Add maintainer contact information  
MAINTAINER inti luna <inti.luna.aviles@gmail.com>  
# Add label  
LABEL project="1_Docker_Kubernetes"  
# Default command to run MongoDB  
CMD ["mongod"]  
-----
```

Se crea imagen

>docker build -t mongodhackinti:1.0 .

Se hace tag (paso previo para subir dockerhub)

>docker tag mongodhackinti:1.0 lunadocker0198/mongodhackinti:1.0

Se sube a dockerhub

>docker push lunadocker0198/mongodhackinti:1.0

Se descarga

>docker pull lunadocker0198/mongodhackinti:1.0

Se ejecuta (run) contenedor con especificaciones

>docker run -m 512m --memory-reservation=512m --cpus=1 -d --name mongodb_proyecto
-e "Entorno=inti_luna" -v volume-proyecto1:/data lunadocker0198/mongodhackinti:1.0

Se inspecciona y se guarda archivo

>docker inspect 53726 >> inspect_output_test_container_mongodhackinti_v1.txt

Cumple con volume, variables de entorno y label

Ingresa a contenedor

>docker exec -it ID### /bin/bash

NET

Para la imagen .NET se hacen algunos cambios:

1. Dockerfile.

Se modifica dockerfile para usar csproj disponible.

```
(base) vant@Agile:~/Documents/datahack/Docker-20240304/practica_proyecto_1_original/práctica/comparacion$ diff Dockerfile_original Dockerfile_modificada
5c5
< ENV ASPNETCORE_URLS=http://+:7007
---
> ENV ASPNETCORE_URLS=http://+:7010
14,15c14,16
< COPY ["pr6.csproj", "./"]
< RUN dotnet restore "pr6.csproj"
---
> COPY ["PracticaEval01.csproj", "./"]
> RUN dotnet restore "PracticaEval01.csproj"
>
18c19
< RUN dotnet build "pr6.csproj" -c Release -o /app/build
---
> RUN dotnet build "PracticaEval01.csproj" -c Release -o /app/build
21c22
< RUN dotnet publish "pr6.csproj" -c Release -o /app/publish /p:UseAppHost=false
---
> RUN dotnet publish "PracticaEval01.csproj" -c Release -o /app/publish /p:UseAppHost=false
26c27
< ENTRYPOINT ["dotnet", "pr6.dll"]
---
> ENTRYPOINT ["dotnet", "PracticaEval01.dll"]
(base) vant@Agile:~/Documents/datahack/Docker-20240304/practica_proyecto_1_original/práctica/comparacion$
```

Comentarios:

- Se sustituye “pr6” por “PracticaEval01” para archivos .dll y .csproj
- Se usa puerto 7007 para imagen versión 1 y 7010 para imagen versión 2 de manera que nos podemos conectar a ambos a la vez.

La versión 1 modificada es:

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0-focal AS base
WORKDIR /app
EXPOSE 5000

ENV ASPNETCORE_URLS=http://+:7007

# Creates a non-root user with an explicit UID and adds permission to access
the /app folder
# For more info, please refer to
https://aka.ms/vscode-docker-dotnet-configure-containers
RUN adduser -u 5678 --disabled-password --gecos "" appuser && chown -R
appuser /app
USER appuser

FROM mcr.microsoft.com/dotnet/sdk:6.0-focal AS build
WORKDIR /src
COPY ["PracticaEval01.csproj", "./"]
RUN dotnet restore "PracticaEval01.csproj"

COPY . .
WORKDIR "/src/."
RUN dotnet build "PracticaEval01.csproj" -c Release -o /app/build
```

```
FROM build AS publish
RUN dotnet publish "PracticaEval01.csproj" -c Release -o /app/publish
/p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "PracticaEval01.dll"]
```

2. PracticaController.Cs

Se modifica dockerfile para usar csproj disponible:

```
PracticaController_v1.cs
(base) vant@Agile:~/Documents/datahack/Docker-20240304/practica_proyecto_1_original/práctica/comparacion$ diff PracticaController_original.cs PracticaController_v1.cs
14c14
<         var dbClient = new MongoClient("mongodb://127.0.0.1:27017");
---
>         var dbClient = new MongoClient(Environment.GetEnvironmentVariable("CONFIG_MONGODB_URL"));
34c34
< }
\ No newline at end of file
---
> }
(base) vant@Agile:~/Documents/datahack/Docker-20240304/practica_proyecto_1_original/práctica/comparacion$
```

La modificación consiste en cambiar la URL de manera que la extrae de la variable de ambiente "CONFIG_MONGODB_URL". Con este cambio, no necesitamos crear una nueva imagen cada vez que se cambie un parámetro de configuración como el puerto o nombre de base de datos, o datos de acceso (usuario+contraseña).

Se logra crear imagen

```
>docker build -t imagenet:1.0 .
```

Se sube imagen (tag y luego push) para versión 1

```
>docker tag imagenet:1.0 lunadocker0198/imagenet:1.0
```

```
>docker push lunadocker0198/imagenet:1.0
```

Se sube imagen (tag y luego push) para versión 2

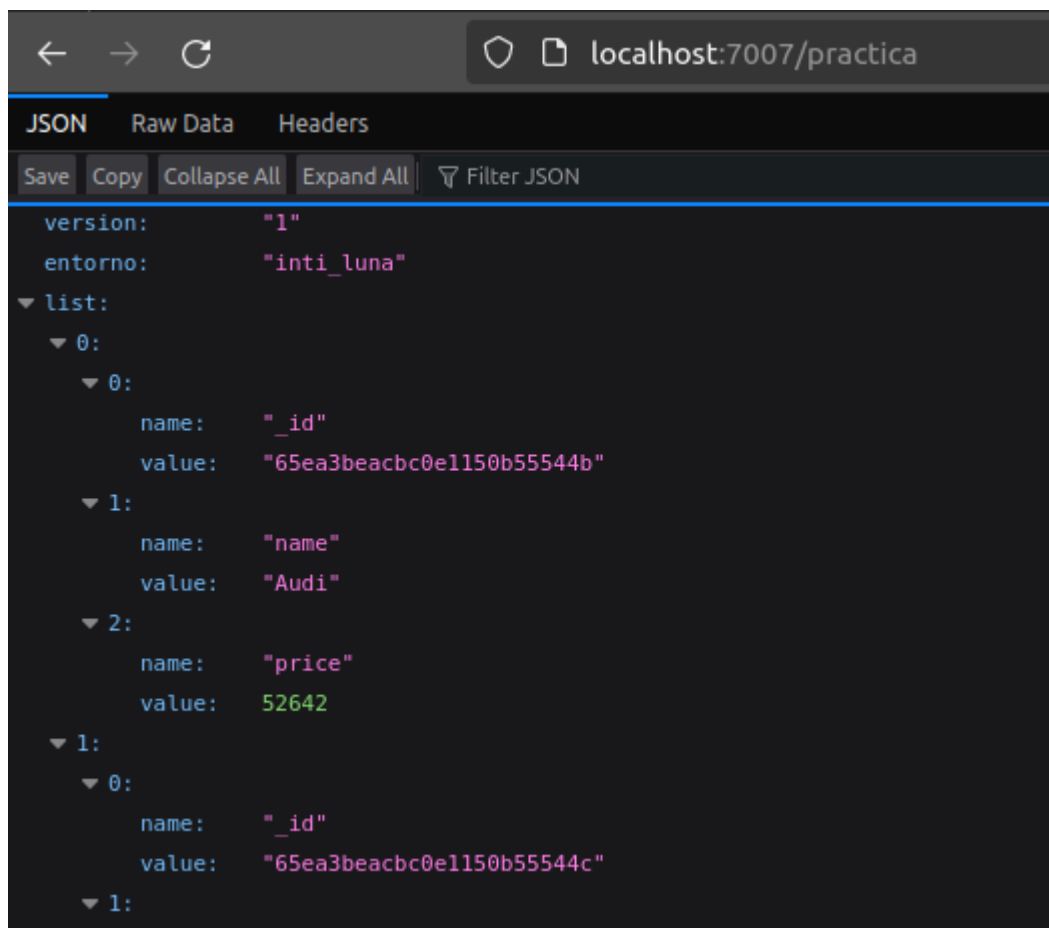
```
>docker tag imagenet:2.0 lunadocker0198/imagenet:2.0
```

```
>docker push lunadocker0198/imagenet:2.0
```

Se corre contenedor

```
>docker run -d --name netapi imagenet:1.0
```

Se comprueba resultado después de ingresar datos a la base de datos



Creación entorno local

Configuración

Se usa docker compose para configurar los servicios de base de datos mongodb y front-end .NET.

Archivo docker-compose.yaml

```
version: "3.8"

services:
  mongo:
    image: lunadocker0198/mongodhackinti:1.0
    #image: mongo:4.0.7-xenial
    restart: always
    ports:
      - "27017:27017"
    deploy:
      resources:
        limits:
          cpus: '4'
          memory: 5120M
        reservations:
          cpus: '4'
          memory: 5120M
    volumes:
      - mongodata:/data/db
      - mongoconfig:/data/configdb
    environment:
      - Entorno=inti_luna
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=adminpw
      - MONGO_INITDB_DATABASE=test
    command: mongod

  netapiv1:
    image: lunadocker0198/imagenet:1.0
    ports:
      - "7007:7007"
    deploy:
      resources:
        limits:
          cpus: '2'
```

```
    memory: 512M
    reservations:
      cpus: '2'
      memory: 512M
volumes:
  - mongodata:/data/db
  - mongoconfig:/data/configdb
environment:
  - Entorno=inti_luna
  - MONGO_INITDB_DATABASE=test
  - MONGO_INITDB_ROOT_USERNAME=admin
  - MONGO_INITDB_ROOT_PASSWORD=adminpw
  - CONFIG_MONGODB_URL=mongodb://admin:adminpw@mongo:27017/
depends_on:
  - mongo
```

```
netapiv2:
  image: lunadocker0198/imagenet:2.0
```

```
ports:
  - "7010:7010"
```

```
deploy:
  resources:
    limits:
      cpus: '2'
      memory: 512M
    reservations:
      cpus: '2'
      memory: 512M
```

```
volumes:
  - mongodata:/data/db
  - mongoconfig:/data/configdb
environment:
  - Entorno=inti_luna
  - MONGO_INITDB_DATABASE=test
  - MONGO_INITDB_ROOT_USERNAME=admin
  - MONGO_INITDB_ROOT_PASSWORD=adminpw
  - CONFIG_MONGODB_URL=mongodb://admin:adminpw@mongo:27017/
depends_on:
  - mongo
```

```
volumes:
  mongodata:
  mongoconfig:
```

Comentarios:

Mongo

Se define servicio de base de datos con imagen creada anteriormente (lunadocker0198/mongodhackinti:1.0), recursos, volúmenes y variables de entorno(nombre del estudiante ("inti_luna"), usuario y clave de base de datos).

Netapi

Se montan v1 y v2 las cuales son idénticas a excepción del puerto por el cual accedemos y la imagen usada (lunadocker0198/imagenet:1.0 y lunadocker0198/imagenet:2.0). En las variables de entorno definimos la variable CONFIG_MONGODB_URL con la URL de conexión a la base de datos (incluyendo usuario y contraseña). Esta variable es la que utiliza el controlador para conectarse a mongodb.

Pruebas

>docker compose up -d

```
(base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$ docker compose up -d
WARN[0000] /home/vant/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local/docker-compose.yaml: 'version' is obsolete
[+] Running 3/4
  ✔ Network definicion_entorno_local_default Created
  ✔ Container definicion_entorno_local-mongo-1 Started
  ✔ Container definicion_entorno_local-netapiv1-1 Started
  ✔ Container definicion_entorno_local-netapiv2-1 Started
(base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$
```

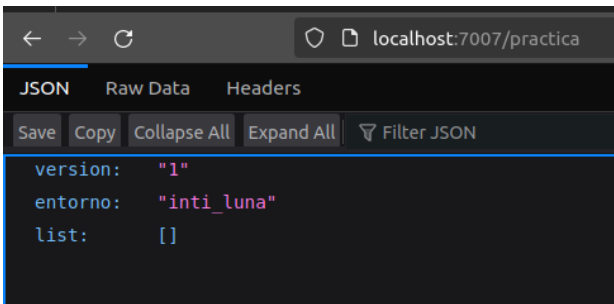
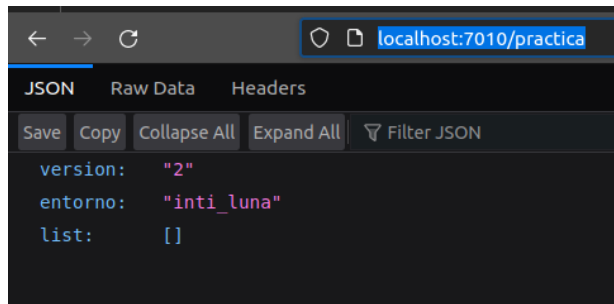
>docker ps -a

```
(base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
0166b178893   lunadocker0198/imagenet:2.0        "dotnet PracticaEval..." 16 minutes ago Up 16 minutes 5000/tcp, 0.0.0.0:7010->7010/tcp, :::7010->7010/tcp   definicion_entorno_local-netapiv2-1
8f6d6ad92c88   lunadocker0198/imagenet:1.0        "dotnet PracticaEval..." 16 minutes ago Up 16 minutes 5000/tcp, 0.0.0.0:7007->7007/tcp, :::7007->7007/tcp   definicion_entorno_local-netapiv1-1
dbacbi9aef7    lunadocker0198/mongodhackinti:1.0  "docker-entrypoint.s..." 16 minutes ago Up 16 minutes 0.0.0.0:27017->27017/tcp, :::27017->27017/tcp         definicion_entorno_local-mongo-1
```

Volúmenes han sido creados

```
(base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$ docker volume ls
DRIVER    VOLUME NAME
local     0ae16c8820413db145cb7c0fade03bd6a50c4de969c951a8adefcf81ee081ce9
local     4d375ea28518c543f4c3cff63e5ee9cb91f6753f8905d7448d589396e95bd75d
local     9a4f14c7711ef84dcd78124929419348bbb6c6daab2350e3e9fd221309ded1d0
local     83f680a2ae80dd244b5506eb991de97f54ee1415bf40e01be9bc666d1e50637f
local     98636d214156e095816c4d4e04aab48b33d9dc0104d6c193295070a1befd3a6d
local     c2e1933394d84a541cda58458d67a2b4ab051ef319cab822aa028b8072861a34
local     definicion_entorno_local_mongoconfig
local     definicion_entorno_local_mongoconfig
```

Revisamos front-end

v1	v2
	

Accedemos a contenedor de mongo e ingresamos datos (usando nombre de contenedor)

```
>docker exec -it definicion_entorno_local-mongo-1 /bin/bash
```

Dentro de contenedor:

```
>mongo
```

```
>use admin
```

```
>db.auth("admin","adminpw")
```

```
>use testdb
```

#Insertamos registros

```
>db.cars.insert({name: "Audi", price: 52642})
```

```
>db.cars.insert({name: "Mercedes", price: 57127})
```

```
>db.cars.insert({name: "Skoda", price: 9000})
```

```
>db.cars.insert({name: "Volvo", price: 29000})
```

```
>db.cars.insert({name: "Bentley", price: 350000})
```

```
>db.cars.insert({name: "Citroen", price: 21000})
```

```
>db.cars.insert({name: "Hummer", price: 41400})
```

```
>db.cars.insert({name: "Volkswagen", price: 21600})
```

```
>db.cars.insert({name: "Seat", price: 2160})
```

#verificamos registros

```
> db.cars.find().pretty()
```

#salimos de sesión mongo y de contenedor

```
>exit
```

```
>exit
```

Captura de proceso:

```
(base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$ docker exec -it definicion_entorno_local-mongo-1 /bin/bash
root@dbacblbaecf7:/# mongo
MongoDB shell version v4.0.7
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("0ae29705-99a6-4554-8b97-f04418b76776") }
MongoDB server version: 4.0.7
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user

> use admin
switched to db admin
> db.auth("admin", "adminpw")
1
> use testdb
switched to db testdb
> db
testdb
> db.cars.insert({name: "Audi", price: 52642})
rs.insert({name: "Mercedes", price: 57127})
db.cars.insert({name: "Skoda", price: 9000})
db.cars.insert({name: "Volvo", price: 29000})
db.cars.insert({name: "Bentley", price: 350000})
db.cars.insert({name: "Citroen", price: 21000})
```

Revisamos front-end netapi v1 y v2

v1

localhost:7007/practica

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
version: "1"
entorno: "inti_luna"
list:
  0:
    0:
      name: "_id"
      value: "65ea3beacbc0e1150b55544b"
    1:
      name: "name"
      value: "Audi"
    2:
      name: "price"
      value: 52642
  1:
    0:
      name: "_id"
      value: "65ea3beacbc0e1150b55544c"
    1:
      name: "name"
      value: "Mercedes"
    2:
      name: "price"
      value: 57127
  2:
    0:
      name: "_id"
      value: "65ea3beacbc0e1150b55544d"
    1:
      name: "name"
      value: "Skoda"
    2:
      name: "price"
      value: 9000
```

v2

localhost:7010/practica

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
version: "2"
entorno: "inti_luna"
list:
  0:
    0:
      name: "_id"
      value: "65ea3beacbc0e1150b55544b"
    1:
      name: "name"
      value: "Audi"
    2:
      name: "price"
      value: 52642
  1:
    0:
      name: "_id"
      value: "65ea3beacbc0e1150b55544c"
    1:
      name: "name"
      value: "Mercedes"
    2:
      name: "price"
      value: 57127
  2:
    0:
      name: "_id"
      value: "65ea3beacbc0e1150b55544d"
    1:
      name: "name"
      value: "Skoda"
    2:
      name: "price"
      value: 9000
```

Apagamos servicios

>docker compose down

```
• (base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$ docker compose down
WARN[0000] /home/vant/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local/docker-compose.yaml: 'version' is obsolete
[+] Running 4/4
✓ Container definicion_entorno_local-netapiv2-1 Removed 0.4s
✓ Container definicion_entorno_local-netapiv1-1 Removed 0.5s
✓ Container definicion_entorno_local-mongo-1 Removed 0.3s
✓ Network definicion_entorno_local_default Removed 0.5s
○ (base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$
```

Se reinicia y se verifica que los datos han persistido.

Creación entorno Kubernetes

>minikube start & status

```
(base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$ minikube start
🐳 minikube v1.32.0 on Ubuntu 22.04
🌟 Using the docker driver based on existing profile
👉 Starting control plane node minikube in cluster minikube
🔄 Pulling base image ...
🔄 Restarting existing docker container for "minikube" ...
🔄 Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
🔗 Configuring bridge CNI (Container Networking Interface) ...
🔍 Verifying Kubernetes components...
   ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: storage-provisioner, default-storageclass
👉 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
(base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
(base) vant@Agile:~/Documents/datahack/Docker-20240304/pro1_entregar/pro1_entregar/definicion_entorno_local$
```

Configuración

Namespace

Creamos namespace para evitar conflictos cuando hay muchas aplicaciones y equipos

>kubectl create namespace proyecto1

```
Error from server (InternalError): error when creating "proyecto1-pod-mongo.yaml": namespaces "proyecto1" not found
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl create namespace proyecto1
namespace/proyecto1 created
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$
```

Verificamos namespaces

>kubectl get namespaces

```
pod/proyecto-mongo created
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl get namespaces
NAME                STATUS    AGE
default              Active    16d
kube-node-lease      Active    16d
kube-public          Active    16d
kube-system          Active    16d
proyecto1            Active    3m19s
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$
```

Secretos

Crear secretos dentro del namespace

```
>kubectl create secret generic my-secret --from-literal=username=admin  
--from-literal=password=adminpw --namespace=proyecto1
```

Revisar secrets

```
>kubectl get secrets --namespace=proyecto1
```

Describir

```
>kubectl describe secrets/my-secret --namespace=proyecto1
```

Mostrar valores (decodificar base64)

Usuario

```
>kubectl get secret my-secret --namespace=proyecto1 -o jsonpath="{.data.username}" |  
base64 --decode
```

Clave

```
>kubectl get secret my-secret --namespace=proyecto1 -o jsonpath="{.data.password}" |  
base64 --decode
```

```
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl create secret generic my-secret --from-literal=username=admin --from-literal=password=adminpw --namespace=proyecto1  
secret/my-secret created  
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl get secrets --namespace=proyecto1  
NAME          TYPE          DATA      AGE  
my-secret      Opaque        2          25s  
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl describe secrets/my-secret --namespace=proyecto1  
Name:         my-secret  
Namespace:    proyecto1  
Labels:       <none>  
Annotations:  <none>  
  
Type: Opaque  
  
Data  
====  
username: 5 bytes  
password: 7 bytes  
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl get secret my-secret --namespace=proyecto1 -o jsonpath="{.data.username}" | base64 --decode  
admin(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl get secret my-secret --namespace=proyecto1 -o jsonpath="{.data.password}" | base64 --decode  
adminpw(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$
```

Una mejora ha sido crear un fichero para automatizar más el despliegue. Para ello transformamos valores a base 64:

```
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ echo -n "admin" | base64  
YWRTaW4=  
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ echo -n "adminpw" | base64  
YWRTaW5wdw==  
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$
```

y luego los usamos en fichero:

```
inti-proyecto1 > ! my-secret.yaml > {} data  
io.k8s.api.core.v1.Secret (v1@secret.json)  
1  apiVersion: v1  
2  kind: Secret  
3  metadata:  
4    name: my-secret  
5    namespace: proyecto1  
6  type: Opaque  
7  data:  
8    username: YWRTaW4= # Valor codificado en base64 para el usuario "admin"  
9    password: YWRTaW5wdw== # Valor codificado en base64 para la contraseña "adminpw"  
10
```

Volumenes

1. Definir un PersistentVolume (PV) que representa el almacenamiento
2. Definir un PersistentVolumeClaim (PVC) que es la solicitud de almacenamiento.
3. Ajustar archivo YAML

Definición PersistentVolume (PV)

```
io.k8s.api.core.v1.PersistentVolume (v1@persistentvolume.js
1  apiVersion: v1
2  kind: PersistentVolume
3  metadata:
4    name: mongo-persistent-volume
5    namespace: proyecto1
6  spec:
7    storageClassName: manual
8    capacity:
9      storage: 5Gi
10   accessModes:
11     - ReadWriteOnce
12   hostPath:
13     path: "/mnt/data"
```

Definición PersistentVolumeClaim (PVC)

```
io.k8s.api.core.v1.PersistentVolumeClaim (v1@persistentvolu
1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    name: mongo-persistent-volume-claim
5    namespace: proyecto1
6  spec:
7    accessModes:
8      - ReadWriteOnce
9    resources:
10      requests:
11        storage: 2Gi
```

Después se inicia deployment y se verifica que se ha creado:

-Servicio

-PersistentVolume

-PersistentVolumeClaim

-Deployment

Realizar Deployment

```
>kubectl apply -f proyecto1-kubernetes.yaml
```

```
>kubectl get deployments --namespace=proyecto1
```

```

deployment.apps "p1-mongo-net" deleted
• (base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl apply -f proyecto1-kubernetes.yaml
service/netapi-service created
persistentvolume/mongo-pv created
persistentvolumeclaim/my-pv-claim created
deployment.apps/p1-mongo-net created
○ (base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$

```

Cuando ya se tienen todos los ficheros, se meten en una carpeta y se corren desde la carpeta superior:

```

persistentvolumeclaim "mongodb-pvc" deleted
• (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ kubectl apply -f inti-proyecto1
deployment.apps/mongo-deployment created
persistentvolume/mongo-persistent-volume created
persistentvolumeclaim/mongo-persistent-volume-claim created
service/mongo-service created
deployment.apps/netapi-deployment created
service/netapi-service created

```

Se verifica deployments, pods, pv y pvc están corriendo sin errores:

```

No resources found in default namespace.
• (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ kubectl get deployments.apps --namespace=proyecto1
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
mongo-deployment    1/1     1             1           24s
netapi-deployment   2/2     2             2           24s
• (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ kubectl get pods --namespace=proyecto1
NAME                                READY   STATUS    RESTARTS   AGE
mongo-deployment-5d99dccbbc-bbg4x  1/1     Running   0           37s
netapi-deployment-7c7f99444c-4th6x  1/1     Running   0           37s
netapi-deployment-7c7f99444c-hfxk7  1/1     Running   0           37s
• (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ kubectl get pv --namespace=proyecto1
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                                STORAGECLASS   REASON   AGE
mongo-persistent-volume  5Gi        RWO            Retain           Available      proyecto1/mongo-persistent-volume-claim  manual        58s
pvc-524037cb-15c5-46e7-b239-46f70c6e8e93  2Gi        RWO            Delete           Bound         proyecto1/mongo-persistent-volume-claim  standard      58s
• (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ kubectl get pvc --namespace=proyecto1
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mongo-persistent-volume-claim  Bound   pvc-524037cb-15c5-46e7-b239-46f70c6e8e93  2Gi        RWO            standard       58s
○ (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$

```

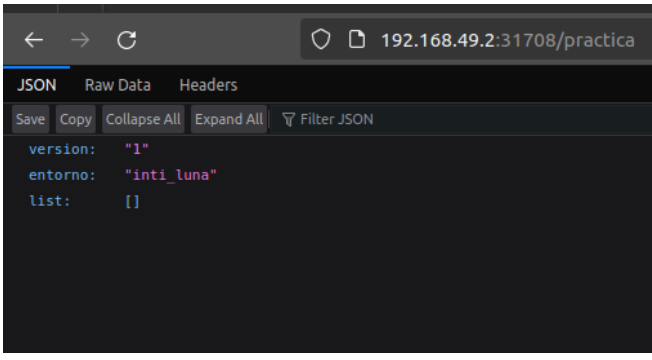
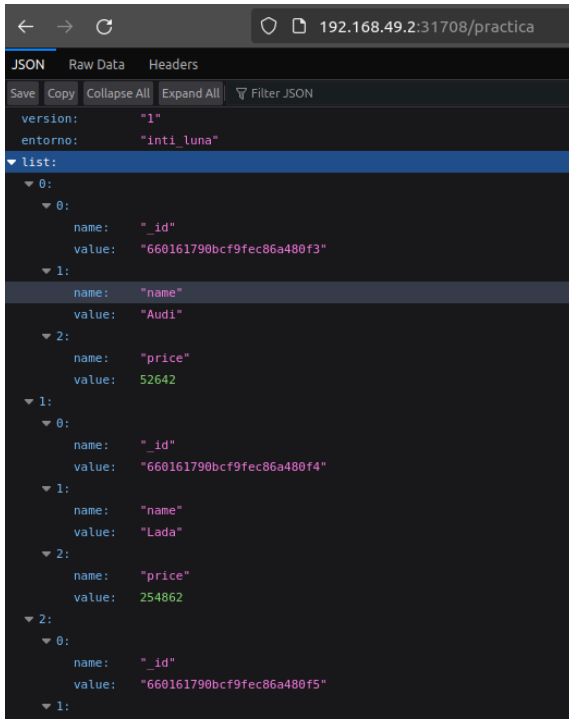
Se obtienen los puertos para conexión

```

• (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ kubectl get services --namespace=proyecto1
NAME                TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
mongo-service       ClusterIP     10.97.178.146   <none>            27017/TCP        5m15s
netapi-service       LoadBalancer 10.100.194.126  <pending>        7007:31139/TCP   5m15s
• (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ minikube ip
192.168.49.2
○ (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$

```

192.168.49.2:31139/practica

Antes de insertar	Después de insertar
	

Pruebas

Variables de entorno se pasan correctamente

>kubectl exec p1-mongo-net-96595f6bb-fvf8n --namespace=proyecto1 -- printenv

```

(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl exec p1-mongo-net-96595f6bb-fvf8n --namespace=proyecto1 -- printenv
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=p1-mongo-net-96595f6bb-fvf8n
Entorno=inti_luna
MONGO_INITDB_ROOT_USERNAME=admin
MONGO_INITDB_ROOT_PASSWORD=adminpw
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
GOSU_VERSION=1.10
JSYAML_VERSION=3.10.0
GPG_KEYS=9DA31620334BD75D9DCB49F368818C72E52529D4
MONGO_PACKAGE=mongodb-org
MONGO_REPO=repo.mongodb.org
MONGO_MAJOR=4.0
MONGO_VERSION=4.0.7
HOME=/root
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$

```

Se verifica a nivel de contenedor ya que no estaba la variable de conexión a mongodb:

>kubectl exec -it p1-mongo-net-67f4fc866b-vwb85 -c netapi-container

--namespace=proyecto1 -- printenv

>kubectl exec -it p1-mongo-net-67f4fc866b-vwb85 -c mongo-container

--namespace=proyecto1 -- printenv

```

(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl exec -it p1-mongo-net-67f4fc866b-vwb85 -c netapi-container --namespace=proyecto1 -- printenv
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=p1-mongo-net-67f4fc866b-vwb85
TERM=xterm
MONGO_INITDB_ROOT_USERNAME=admin
MONGO_INITDB_ROOT_PASSWORD=adminpw
CONFIG_MONGODB_URL=mongodb://admin:adminpw@mongo:27017/
Entorno=inti_luna
MONGO_INITDB_DATABASE=test
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
ASPNETCORE_URLS=http://*:7807
DOTNET_RUNNING_IN_CONTAINER=true
DOTNET_VERSION=6.0.27
ASPNET_VERSION=6.0.27
HOME=/home/appuser

(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$ kubectl exec -it p1-mongo-net-67f4fc866b-vwb85 -c mongo-container --namespace=proyecto1 -- printenv
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=p1-mongo-net-67f4fc866b-vwb85
TERM=xterm
Entorno=inti_luna
MONGO_INITDB_ROOT_USERNAME=admin
MONGO_INITDB_ROOT_PASSWORD=adminpw
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
GOSU_VERSION=1.10
JSYAML_VERSION=3.10.0
GPG_KEYS=9DA31620334BD75D9DCB49F368818C72E52529D4
MONGO_PACKAGE=mongodb-org
MONGO_REPO=repo.mongodb.org
MONGO_MAJOR=4.0
MONGO_VERSION=4.0.7
HOME=/root
(base) vant@Agile:~/Documents/datahack/macpro_bk/Kubernetes$

```

Prueba de persistencia

>kubectl exec -it mongo-deployment-5d99dccbbc-tt5v9 --namespace=proyecto1 mongo

(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects/inti-proyecto1\$ kubectl exec -it

mongo-deployment-5d99dccbbc-vr29z -- /bin/bash

root@mongo-deployment-5d99dccbbc-vr29z:/# mongo

MongoDB shell version v4.0.7

connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("821205bb-81a1-4fa4-84a7-0543cdeae783") }

MongoDB server version: 4.0.7

Welcome to the MongoDB shell.

For interactive help, type "help".

For more comprehensive documentation, see

<http://docs.mongodb.org/>

Questions? Try the support group

<http://groups.google.com/group/mongodb-user>

> use admin

switched to db admin

> db.auth("admin","adminpw")

1

> use testdb

switched to db testdb

> db.cars.insert({name: "Audi", price: 52642})

rs.insert({name: "Mercedes", price: 57127})

db.cars.insert({name: "Skoda", price: 9000})WriteResult({ "nInserted" : 1 })

dbdb.cars.insert({name: "Mercedes", price: 57127})

WriteResult({ "nInserted" : 1 })

> db.cars.insert({name: "Skoda", price: 9000})

WriteResult({ "nInserted" : 1 })

> db.cars.insert({name: "Volvo", price: 29000})

WriteResult({ "nInserted" : 1 })

> db.cars.insert({name: "Bentley", price: 350000})

WriteResult({ "nInserted" : 1 })

> db.cars.insert({name: "Citroen", price: 21000})

WriteResult({ "nInserted" : 1 })

> db.cars.insert({name: "Hummer", price: 41400})

WriteResult({ "nInserted" : 1 })

> db.cars.insert({name: "Volkswagen", price: 21600})

WriteResult({ "nInserted" : 1 })

> db.cars.insert({name: "Seat", price: 2160})

WriteResult({ "nInserted" : 1 })

> db.cars.find().pretty()

{

```

    "_id" : ObjectId("65fd6c0494a7d3117d2ad2fd"),
    "name" : "Audi",
    "price" : 52642
  }
  {
    "_id" : ObjectId("65fd6c0494a7d3117d2ad2fe"),
    "name" : "Mercedes",
    "price" : 57127
  }
  {
    "_id" : ObjectId("65fd6c0494a7d3117d2ad2ff"),
    "name" : "Skoda",
    "price" : 9000
  }
  {
    "_id" : ObjectId("65fd6c0494a7d3117d2ad300"),
    "name" : "Volvo",
    "price" : 29000
  }
  {
    "_id" : ObjectId("65fd6c0494a7d3117d2ad301"),
    "name" : "Bentley",
    "price" : 350000
  }
  {
    "_id" : ObjectId("65fd6c0494a7d3117d2ad302"),
    "name" : "Citroen",
    "price" : 21000
  }
  {
    "_id" : ObjectId("65fd6c0494a7d3117d2ad303"),
    "name" : "Hummer",
    "price" : 41400
  }
  {
    "_id" : ObjectId("65fd6c0494a7d3117d2ad304"),
    "name" : "Volkswagen",
    "price" : 21600
  }
  {
    "_id" : ObjectId("65fd6c0494a7d3117d2ad305"),
    "name" : "Seat",
    "price" : 2160
  }
}

```

```
> exit
```

```
bye
```

```
root@mongo-deployment-5d99dccbbc-vr29z:/# exit
```

```
exit
```

```
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects/inti-proyecto1$ kubectl delete deployments mongo-deployment
```

```
deployment.apps "mongo-deployment" deleted
```

```
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects/inti-proyecto1$ kubectl apply -f mongo-deployment.yaml
```

```
deployment.apps/mongo-deployment created
```

```
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects/inti-proyecto1$ kubectl get deployments.apps
```

```

NAME                READY  UP-TO-DATE  AVAILABLE  AGE
mongo-deployment    1/1    1           1          17s
netapi-deployment    2/2    2           2          26m

```

```
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects/inti-proyecto1$ kubectl get pod
```

```

NAME                READY  STATUS    RESTARTS  AGE
mongo-deployment-5d99dccbbc-jfq9s  1/1    Running   0          25s
netapi-deployment-7c7f99444c-d4lgh  1/1    Running   0          26m
netapi-deployment-7c7f99444c-stfpv  1/1    Running   0          26m

```

```
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects/inti-proyecto1$ kubectl exec -it mongo-deployment-5d99dccbbc-jfq9s -- /bin/bash
```

```
root@mongo-deployment-5d99dccbbc-jfq9s:/# db.cars.find().pretty()
```

```
bash: syntax error near unexpected token `pretty'
```

```
root@mongo-deployment-5d99dccbbc-jfq9s:/# mongo
```

```
MongoDB shell version v4.0.7
```

```
connecting to: mongod://127.0.0.1:27017/?gssapiServiceName=mongod
```

```
Implicit session: session { "id" : UUID("a3190096-3a97-4708-b94f-72d323b2f024") }
```

```
MongoDB server version: 4.0.7
```

```
Welcome to the MongoDB shell.
```

```
For interactive help, type "help".
```

```

For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
> use admin
switched to db admin
> db.auth("admin","adminpw")
1
> use testdb
switched to db testdb
> db.cars.find().pretty()
{
  "_id" : ObjectId("65fd6c0494a7d3117d2ad2fd"),
  "name" : "Audi",
  "price" : 52642
}
{
  "_id" : ObjectId("65fd6c0494a7d3117d2ad2fe"),
  "name" : "Mercedes",
  "price" : 57127
}
{
  "_id" : ObjectId("65fd6c0494a7d3117d2ad2ff"),
  "name" : "Skoda",
  "price" : 9000
}
{
  "_id" : ObjectId("65fd6c0494a7d3117d2ad300"),
  "name" : "Volvo",
  "price" : 29000
}
{
  "_id" : ObjectId("65fd6c0494a7d3117d2ad301"),
  "name" : "Bentley",
  "price" : 350000
}
{
  "_id" : ObjectId("65fd6c0494a7d3117d2ad302"),
  "name" : "Citroen",
  "price" : 21000
}
{
  "_id" : ObjectId("65fd6c0494a7d3117d2ad303"),
  "name" : "Hummer",
  "price" : 41400
}
{
  "_id" : ObjectId("65fd6c0494a7d3117d2ad304"),
  "name" : "Volkswagen",
  "price" : 21600
}
{
  "_id" : ObjectId("65fd6c0494a7d3117d2ad305"),
  "name" : "Seat",
  "price" : 2160
}
> exit
bye
root@mongo-deployment-5d99dccbbc-jfq9s:/# exit
exit

```

Correr proyecto

Corremos desde un nivel superior la carpeta con todos los ficheros

```
>kubectl apply -f inti-proyecto1
```

```
deployment.apps/netapi-deployment-v1 created
● (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ kubectl apply -f inti-proyecto1
deployment.apps/mongo-deployment created
persistentvolume/mongo-persistent-volume unchanged
persistentvolumeclaim/mongo-persistent-volume-claim unchanged
service/mongo-service created
secret/my-secret unchanged
deployment.apps/netapi-deployment-v1 created
deployment.apps/netapi-deployment-v2 created
service/netapi-service-v1 created
service/netapi-service-v2 created
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$
```

Listamos los recursos y revisamos que estén sin problemas

>kubectl get all -n proyecto1

```
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ kubectl get all -n proyecto1
NAME                                READY   STATUS    RESTARTS   AGE
pod/mongo-deployment-5d99dccbbc-dcjdq 1/1     Running   0           28s
pod/netapi-deployment-v1-7596f4bb9-btfnn 1/1     Running   0           28s
pod/netapi-deployment-v1-7596f4bb9-ws7dx 1/1     Running   0           28s
pod/netapi-deployment-v2-7ff7784d75-crncs 1/1     Running   0           28s
pod/netapi-deployment-v2-7ff7784d75-gln77 1/1     Running   0           28s

NAME                                TYPE               CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/mongo-service               ClusterIP          10.109.198.215  <none>           27017/TCP        28s
service/netapi-service-v1           LoadBalancer      10.104.124.112  <pending>        7007:32298/TCP   28s
service/netapi-service-v2           LoadBalancer      10.103.165.127  <pending>        7010:32038/TCP   28s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongo-deployment     1/1     1             1           29s
deployment.apps/netapi-deployment-v1 2/2     2             2           28s
deployment.apps/netapi-deployment-v2 2/2     2             2           28s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/mongo-deployment-5d99dccbbc 1         1         1       28s
replicaset.apps/netapi-deployment-v1-7596f4bb9 2         2         2       28s
replicaset.apps/netapi-deployment-v2-7ff7784d75 2         2         2       28s
```

Obtenemos IP para acceder

>minikube ip

```
(base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ minikube ip
192.168.49.2
```

V1

```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
version: "1"
entorno: "inti_luna"
list:
  0:
    name: "_id"
    value: "660161790bcf9fec86a480f3"
  1:
    name: "name"
    value: "Audi"
  2:
    name: "price"
    value: 52642
  1:
    name: "_id"
    value: "660161790bcf9fec86a480f4"
  1:
    name: "name"
    value: "Lada"
  2:
    name: "price"
    value: 254862
  2:
    name: "_id"
    value: "660161790bcf9fec86a480f5"
  1:
    name: "name"
    value: "Mercedes"
  2:
    name: "price"
    value: 57127
  3:
    name: "_id"
    value: "660161790bcf9fec86a480f6"
```

V2

```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
version: "2"
entorno: "inti_luna"
list:
  0:
    name: "_id"
    value: "660161790bcf9fec86a480f3"
  1:
    name: "name"
    value: "Audi"
  2:
    name: "price"
    value: 52642
  1:
    name: "_id"
    value: "660161790bcf9fec86a480f4"
  1:
    name: "name"
    value: "Lada"
  2:
    name: "price"
    value: 254862
  2:
    name: "_id"
    value: "660161790bcf9fec86a480f5"
  1:
    name: "name"
    value: "Mercedes"
  2:
    name: "price"
    value: 57127
  3:
    name: "_id"
    value: "660161790bcf9fec86a480f6"
```

Y para apagar y eliminar todo:

>kubectl delete all --all -n proyecto1

```
192.168.49.2
• (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$ kubectl delete all --all -n proyecto1
pod "mongo-deployment-5d99dccbbc-dcjdq" deleted
pod "netapi-deployment-v1-7596f4bb9-btfnn" deleted
pod "netapi-deployment-v1-7596f4bb9-ws7dx" deleted
pod "netapi-deployment-v2-7ff7784d75-crncs" deleted
pod "netapi-deployment-v2-7ff7784d75-gln77" deleted
service "mongo-service" deleted
service "netapi-service-v1" deleted
service "netapi-service-v2" deleted
deployment.apps "mongo-deployment" deleted
deployment.apps "netapi-deployment-v1" deleted
deployment.apps "netapi-deployment-v2" deleted
○ (base) vant@Agile:~/Documents/datahack/k8s_handbook_freecode/kubernetes-handbook-projects$
```

Problemas encontrados

Docker-Part

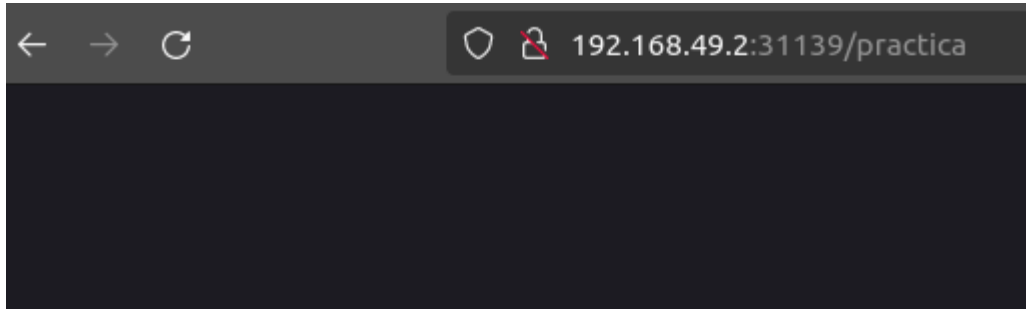
Conexión de front-end a mongo:

1. el URL de conexión que se le pasaba tenía espacios
2. en controller dentro de API .NET tenía un enlace estático sin user:pw por lo que al cambiar con credenciales y dinámico en dependencia de variable de entorno se soluciona.

Acceso con credenciales a mongodb me fue difícil encontrar pero esta solucionado

Kubernetes-part

Se ha probado la conexión pero hay algún error por lo que no se visualiza los datos:



Se ha probado y revisado:

- Imagen funciona bien en docker compose
- Volumenes estan bien configurados y se prueba persistencia
- Puertos estan bien declarados

El problema está en la manera como nos conectamos de un deployment (pod) a otro. En Docker Compose, los contenedores dentro del mismo servicio pueden comunicarse entre sí directamente utilizando los nombres de los servicios definidos en el archivo docker-compose.yml por lo que funciona perfectamente:

`"mongodb://$(MONGO_INITDB_ROOT_USERNAME):$(MONGO_INITDB_ROOT_PASSWORD)@mongo:27017/"`.

Sin embargo, en Kubernetes, los contenedores no se comunican directamente por nombres de contenedor como en Docker Compose. En su lugar, se utilizan servicios para exponer la conectividad entre los contenedores. Como ya hemos creado un "service" anteriormente usamos el nombre del servicio para que su valor dinámico pase a la conexión quedando la variable de conexión de la siguiente manera:

`"mongodb://$(MONGO_INITDB_ROOT_USERNAME):$(MONGO_INITDB_ROOT_PASSWORD)@mongo-service:27017/"`

Recursos

Conceptos claves

Pod

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes

Service

A service in Kubernetes is an abstract way to expose an application running on a set of pods as a network service.

Service is an abstraction which defines a logical set of Pods and a policy by which to access them.

Herramientas

Display network connections

>netstat -tuln

Enlaces

Usa para Docker

https://hub.docker.com/_/mongo

<https://stackoverflow.com/questions/34559557/how-to-enable-authentication-on-mongodb-through-docker>

Usados para Kubernetes

<https://kubernetes.io/docs/reference/kubectl/quick-reference/>

<https://www.freecodecamp.org/news/the-kubernetes-handbook/>