


Integrantes Grupo 6:

- Huallpamaita Quispe Washington
- Luque cano William Placido
- Tineo Santamaria Ronald Herliss
- Rey Gonzáles Ana Teresa

1. Repositorio Github del proyecto CRUD:

<https://github.com/rtineo/crud>

 crud PrivateWatch 1

main 1 Branch 0 Tags

Go to file

Add file

Code

arey Se agrega log y se agrega Dockerfile, verificar Readme file c994bd0 · last week 3 Commits

gradle/wrapper	se sube el proyecto	2 weeks ago
src	Se agrega log y se agrega Dockerfile, verificar Readme file	last week
.gitattributes	se sube el proyecto	2 weeks ago
.gitignore	se sube el proyecto	2 weeks ago
Dockerfile	Se agrega log y se agrega Dockerfile, verificar Readme file	last week
Readme.txt	Se agrega log y se agrega Dockerfile, verificar Readme file	last week
build.gradle	se sube el proyecto	2 weeks ago
gradlew	se sube el proyecto	2 weeks ago
gradlew.bat	se sube el proyecto	2 weeks ago
pom.xml	Se agrega log y se agrega Dockerfile, verificar Readme file	last week
settings.gradle	se sube el proyecto	2 weeks ago

2. Dockerfile:

main + Q

Go to file

gradle

src

.gitattributes

.gitignore

Dockerfile

Readme.txt

build.gradle

gradlew

gradlew.bat

pom.xml

settings.gradle

arey Se agrega log y se agrega Dockerfile, verificar Readme file

Code

Blame

14 lines (10 loc) · 298 Bytes

Code 55% faster with GitHub Copilot

```
1 #imagen base de java
2 FROM openjdk:21-jdk-slim
3
4 #directorio de la app
5 WORKDIR /app
6
7 #Copiar el archivo JAR generado en el contenedor
8 COPY target/crud-0.0.1-SNAPSHOT.jar app.jar
9
10 #exponer el puerto que usa la app
11 EXPOSE 9055
12
13 #Comando para ejecutar la aplicacion
14 ENTRYPOINT ["java","-jar", "app.jar"]
```

En powershell ejecutamos la carga de la imagen:

```
docker build -t crud:v1 .
```

Verificamos que exista la Imagen a nivel local:

```
docker images
```

```
PS C:\Users\Netec\crud> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
crud	v1	ae34a80bf63e	45 minutes ago	783MB

Desplegamos la imagen con el puerto 8088:

```
docker run -d -p 8088:9055 --name pf-crud crud:v1
```

Verificamos la escucha de la app desde <http://localhost:8088/productos>

The screenshot displays a REST client interface with two requests and their responses.

Request 1: POST
URL: `http://localhost:8088/productos`
Method: `POST`
Body (JSON):

```
{  "nombre": "cuadernos",  "descripcion": "triple region a4 100 hojas",  "precio": 7.80}
```


Response: `200 OK`, `54 ms`, `249 B`.
Body (JSON):

```
{  "id": 2,  "nombre": "cuadernos",  "descripcion": "triple region a4 100 hojas",  "precio": 7.8}
```

Request 2: GET
URL: `http://localhost:8088/productos`
Method: `GET`
Response: `200 OK`, `16 ms`, `251 B`.
Body (JSON):

```
[  {    "id": 2,    "nombre": "cuadernos",    "descripcion": "triple region a4 100 hojas",    "precio": 7.8  }]
```

- Después de subir la imagen al Docker Hub lo verificamos:

<https://hub.docker.com/r/anndocker2020/crud>

The image shows two screenshots of the Docker Hub interface. The top screenshot is a search results page for 'anndocker2020/crud'. It displays a list of repositories, with 'anndocker2020/crud' at the top, updated 7 days ago. Below it are 'crudapi/crudapi-admin-web' and 'crudfab/appfab'. The bottom screenshot shows the detailed page for 'anndocker2020/crud'. It includes the repository icon, name, and description: 'crud del grupo 6 docker rtineo, arey, whuallpamaita y wluque.' It also shows a 'No overview available' message and a 'Docker Pull Command' box with the command 'docker pull anndocker2020/crud' and a 'Copy' button.

- La imagen del crud podrá descargarse desde el DockerHub:
`docker pull anndocker2020/crud`

- Creamos el namespace.
Todos
`kubectrl create namespace pfinal`

```
MINGW64:/c/proyecto/crud
Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get ingress
NAME          CLASS      HOSTS          ADDRESS          PORTS    AGE
crud-ingress  <none>     crud.midominio.com  crud.midominio.com  80      101s

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl apply -f crud-ingress.yaml
ingress.networking.k8s.io/crud-ingress configured

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get ingress
NAME          CLASS      HOSTS          ADDRESS          PORTS    AGE
crud-ingress  crud       crud.midominio.com  crud.midominio.com  80      6m4s

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get pods -n ingress-crud
No resources found in ingress-crud namespace.

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl create namespace pfinal
namespace/pfinal created

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$
```

6. Archivos YAML de Kubernetes Deployment:
Se crea el archivo crud-deployment.yaml con la siguiente instrucción:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: crud-deployment
  namespace: pfinal
  labels:
    app: crud
spec:
  replicas: 2 # Configuramos el número de réplicas.
  selector:
    matchLabels:
      app: crud
  template:
    metadata:
      labels:
        app: crud
    spec:
      containers:
        - name: crud
          image: anndocker2020/crud:v1 # Imagen del contenedor.
          ports:
            - containerPort: 9055 # Puerto expuesto dentro del
contenedor.
```

Ejecutamos el comando para realizar el deployment
kubectl apply -f crud-deployment.yaml

Al realizar el deployment se tiene la siguiente imagen, donde se observa que se encuentra estado desplegado sin problemas desde la imagen “anndocker2020/crud:v1” en el puerto 9055 los **dos pods de réplica**.

“crud deployment”

```
MINGW64/c/proyecto/crud
Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl delete -f crud-service.yaml
service "crud-service" deleted

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl delete -f deploygrupo6.yaml
deployment.apps "crud" deleted

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP    42d

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc -f pfinal
error: the path "pfinal" does not exist

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc -n pfinal
No resources found in pfinal namespace.

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl apply -f crud-deployment.yaml
deployment.apps/crud-deployment created

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc -n pfinal
No resources found in pfinal namespace.

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get pod -n pfinal
NAME                                READY   STATUS    RESTARTS   AGE
crud-deployment-6747bf5697-6v164    1/1     Running   0          26s
crud-deployment-6747bf5697-n9gdq    1/1     Running   0          26s

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ |
```

Nota: en caso de que el estado no cambie a Running se sugiere ejecutar los siguientes comandos para reiniciar el calico.

kubectl rollout restart daemonset/calico-node -n kube-system

kubectl rollout restart deployment/calico-kube-controllers -n kube-system

7. Archivos YAML de Kubernetes Service:

Para crear el Service en Kubernetes, seguimos estos pasos:

- Creamos el archivo crud-service.yaml con la siguiente configuración:

```
apiVersion: v1
kind: Service
metadata:
  name: crud-service
  namespace: pfinal
spec:
  selector:
    app: crud # Selector que coincide con las labels del
Deployment
  ports:
    - port: 8080 # Puerto del servicio
      targetPort: 9055 # Puerto del contenedor
    type: ClusterIP # Tipo de servicio
```

- Aplicamos la configuración del Service con el siguiente comando:

```
kubectl apply -f crud-service.yaml
```

```
MINGW64:/c/proyecto/crud
specialized-daemonset-bq6z2 1/1 Running 0 46h

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
crud-6747bf5697-hxldf              1/1    Running   0          20d
crud-6747bf5697-zpbv7              1/1    Running   0          20d
hostpath-pod                       1/1    Running   0          2d
mi-aplicacion-5bbb7d7cbf-5hfvc     1/1    Running   0          20d
mi-aplicacion-5bbb7d7cbf-pvc7f     1/1    Running   0          20d
mi-aplicacion-5bbb7d7cbf-vj879     1/1    Running   0          20d
my-pod                             1/1    Running   0          20d
nginx-test                         1/1    Running   0          22d
node-pod                           1/1    Running   0          16d
pvc-pod                            1/1    Running   2 (96m ago) 2d
specialized-daemonset-5ldln        1/1    Running   0          46h
specialized-daemonset-bq6z2        1/1    Running   0          46h

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl apply -f crud-service.yaml
service/crud-service created
```

- Verificamos que el servicio se ha creado correctamente:

```
kubectl get services -n pfinal
```

```
MINGW64:/c/proyecto/crud
Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc -f pfinal
error: the path "pfinal" does not exist

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc -n pfinal
No resources found in pfinal namespace.

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl apply -f crud-deployment.yaml
deployment.apps/crud-deployment created

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc -n pfinal
No resources found in pfinal namespace.

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get pod -n pfinal
NAME                                READY   STATUS    RESTARTS   AGE
crud-deployment-6747bf5697-6vl64    1/1    Running   0          26s
crud-deployment-6747bf5697-n9gdq    1/1    Running   0          26s

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl edit svc ingress-nginx-controller -n ingress-nginx
Edit cancelled, no changes made.

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl apply -f crud-service.yaml
service/crud-service created

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc -n pfinal
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
crud-service ClusterIP   10.106.48.172 <none>        8080/TCP   11s

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$
```

- El servicio crud-service ahora expone internamente el puerto 80 del clúster y redirecciona el tráfico al puerto 9055 de los pods seleccionados con la etiqueta app: crud.

8. Archivos YAML de Kubernetes Ingress:

Para exponer nuestra aplicación al exterior a través de un Ingress, seguimos estos pasos:

- Creamos el archivo crud-ingress.yaml con la siguiente configuración:

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: crud-ingress
  namespace: pfinal
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: / # Reescritura
de ruta
spec:
  ingressClassName: nginx
  rules:
    - host: crud.midominio.com # Hostname para acceder
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: crud-service
                port:
                  number: 8080

```

- Aplicamos la configuración del Ingress:

```
kubectl apply -f crud-ingress.yaml
```

The screenshot shows a terminal window with the following content:

```

MINGW64/c/proyecto/crud
crud-6747bf5697-zpbv7      1/1      Running    0          20d
hostpath-pod              1/1      Running    0          2d1h
mi-aplicacion-5bbb7d7cbf-5hfvc 1/1      Running    0          20d
mi-aplicacion-5bbb7d7cbf-pvc7f 1/1      Running    0          20d
mi-aplicacion-5bbb7d7cbf-vj879 1/1      Running    0          20d
my-pod                    1/1      Running    0          20d
nginx-test                1/1      Running    0          22d
node-pod                  1/1      Running    0          16d
pvc-pod                   1/1      Running    2 (105m ago) 2d1h
specialized-daemonset-5ldln 1/1      Running    0          47h
specialized-daemonset-bq6z2 1/1      Running    0          47h

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE
crud-service  ClusterIP     10.99.244.149 <none>        80/TCP           3m4s
kubernetes    ClusterIP     10.96.0.1     <none>        443/TCP          42d

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl apply -f crud-ingress.yaml
ingress.networking.k8s.io/crud-ingress created

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$

```

- Verificamos que el Ingress se ha creado correctamente:

```
kubectl get ingress -n pfinal
```

- Es importante destacar que para que el Ingress funcione correctamente, necesitamos:

- o Un controlador de Ingress instalado en el clúster (como NGINX Ingress Controller)

- Configurar nuestro DNS para que crud.midominio.com apunte a la dirección IP del controlador de Ingress
- Alternativamente, para pruebas locales, podemos modificar el archivo /etc/hosts para agregar una entrada que apunte a la IP del Ingress Controller

9. Pruebas de invocación por el ingress

Editamos en caliente el ingress-controller se cambia el Type: LoadBalancer -> Type: NodePort

```
kubectl edit svc ingress-nginx-controller -n ingress-nginx
```

```

MINGW64:/c/proyecto/crud
$ kubectl apply -f crud-deployment.yaml
deployment.apps/crud-deployment created

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc -n pfinal
No resources found in pfinal namespace.

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get pod -n pfinal
NAME                                READY   STATUS    RESTARTS   AGE
crud-deployment-6747bf5697-6vl64    1/1     Running   0           26s
crud-deployment-6747bf5697-n9gdq    1/1     Running   0           26s

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl edit svc ingress-nginx-controller -n ingress-nginx
Edit cancelled, no changes made.

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl apply -f crud-service.yaml
service/crud-service created

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get svc -n pfinal
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
crud-service  ClusterIP   10.106.48.172 <none>        8080/TCP    11s

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl apply -f crud-ingress.yaml
ingress.networking.k8s.io/crud-ingress created

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get ingress -n pfinal
NAME      CLASS    HOSTS                ADDRESS        PORTS    AGE
crud-ingress  nginx    crud.midominio.com   10.111.50.74   80       13s

Netec@labRBWRZK MINGW64 /c/proyecto/crud (main)
$ |

```

Verificamos que ahora si indica NodePort

```
kubectl describe svc ingress-nginx-controller -n ingress-nginx
```



```
MINGW64/c/proyecto/crud

Netec@labR8wRZK MINGW64 /c/proyecto/crud (main)
$ kubectl get ingress -n pfinal
NAME          CLASS  HOSTS          ADDRESS        PORTS   AGE
crud-ingress  nginx  crud.midominio.com  10.111.50.74   80      13s

Netec@labR8wRZK MINGW64 /c/proyecto/crud (main)
$ kubectl describe svc ingress-nginx-controller -n ingress-nginx
Name:          ingress-nginx-controller
Namespace:     ingress-nginx
Labels:        app.kubernetes.io/component=controller
               app.kubernetes.io/instance=ingress-nginx
               app.kubernetes.io/name=ingress-nginx
               app.kubernetes.io/part-of=ingress-nginx
               app.kubernetes.io/version=1.12.1
Annotations:   <none>
Selector:      app.kubernetes.io/component=controller,app.kubernetes.io/instance=ingress-nginx
Type:          NodePort
IP Family Policy: SingleStack
IP Families:   IPv4
IP:            10.111.50.74
IPs:           10.111.50.74
Port:          http 80/TCP
TargetPort:    http/TCP
NodePort:      http 32171/TCP
Endpoints:     10.244.235.154:80
Port:          https 443/TCP
TargetPort:    https/TCP
NodePort:      https 30707/TCP
Endpoints:     10.244.235.154:443
Session Affinity: None
External Traffic Policy: Local
Events:        <none>

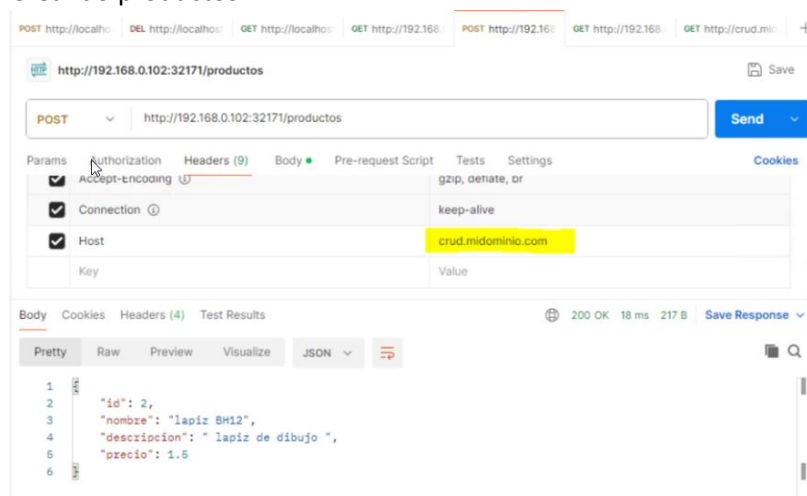
Netec@labR8wRZK MINGW64 /c/proyecto/crud (main)
$
```

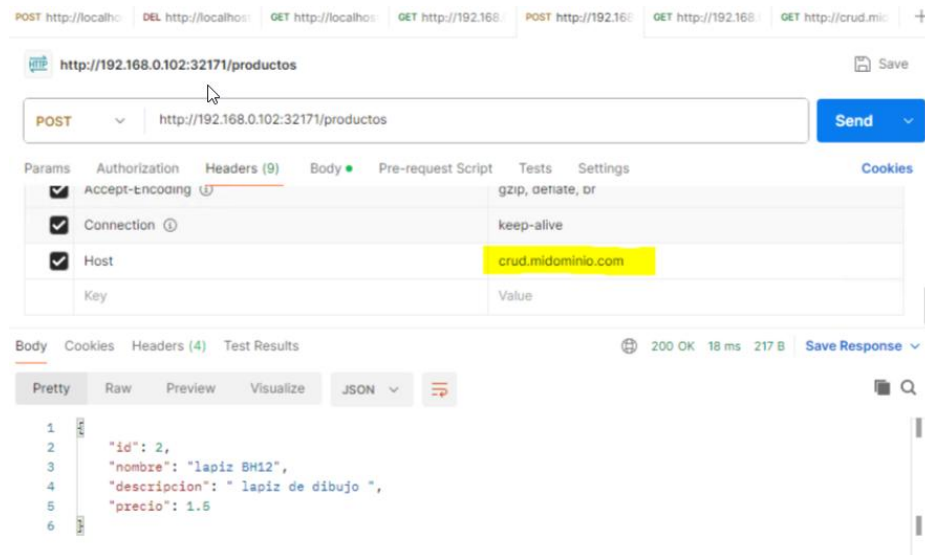
Pruebas desde el postman

El IP: 192.168.0.102 corresponde con el IP del HOST worker1 y el puerto del ingress controller.

<http://192.168.0.102:32171/productos>

- Creando productos





- Consultando productos

