# example-books-microservices

Questo progetto ha come obbiettivo quello di creare un microservizio di esempio da utilizzare per mostrare le operazioni necessarie per pubblicare su *KUBERNETES*.

## Tecnologie utilizzate

Il progetto prevede l'utilizzo delle seguenti tecnologie:

- nodejs
- express
- docker
- kubernetes

## Come utilizzare il progetto

Per utilizzare il progetto si devono seguire i seguenti passi:

- Download del progetto da github
- Eseguire la build Docker
- Effettuare il deploy su *kubernetes*
- Creare il servizio su *kubernetes*
- Verificare il funzionamento

### Download del progetto da github

Per effettuare il download del progetto è possibile eseguire il comando `git clone`

```
git clone https://github.com/course-kubernetes/example-microservices.git
```

### Eseguire la build Docker

Una volta scaricato il progetto è possibile eseguirlo in locale tramite il comando

```
    node index.js
```

ed anche generare l'immagine ***docker*** del progetto

```
docker build -t example-books-microservice:1.1 .
```

In questo modo viene generato il container del progetto che è possibile eseguire direttamente

```
docker run -p 3000:3000 example-books-microservice:1.1
```

o distribuire l'immagine in un orchestratore come *KUBERNETES*.

### Effettuare il deploy su *kubernetes*

Per distribuire il container del progetto su *KUBERNETES* è possibile procedere in diversi modi.

Innanzi tutto è possibile

- farlo direttamente da terminale se ci si sta appoggiando ad un REGISTRY DOCKER

```
    kubectl run example-books-deployment --image=lucasalzone/example-books-microservice:1.1.0
```

> NOTE: Documentazione Kubectl

- creare i file di configurazione YAML per il deploy ed applicarlo su kubernetes

```
    kubectl apply -f example-books-deploy.yaml
```

> NOTE: Documentazione Deployment

Nel secondo caso è possibile usare l'opzione `imagePullPolicy: Never` per dire di non effettuare il pull dell'immagine, ma di cercarla direttamente nell'engine locale.

**Il file YAML**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example-books-microservice
  namespace: course-kubernetes
spec:
  replicas: 1
  selector:
    matchLabels:
      app: example-books-microservice
  template:
    metadata:
      labels:
        app: example-books-microservice
    spec:
      containers:
        - name: example-books-microservice
          image: example-books-microservice:1.1
          imagePullPolicy: Never
          ports:
            - containerPort: 3000
```

- ***apiVersion***: Indica la versione dell'API utilizzata.
- ***kind***: Deve essere impostato su "Deployment"
- ***metadata.name***: Il nome del deployment.
- ***spec.replicas***: Il numero desiderato di repliche del pod.
- ***spec.selector.matchLabels***: Il selettore
- ***spec.template.spec.containers.name***: Il nome del container.
- ***spec.template.spec.containers.image***: Il nome dell'immagine del container.
- ***spec.template.spec.containers.ports.containerPort***: La porta del container da esporre.

Questo file YAML definisce un deployment che crea un singolo ***pod*** con un container basato sull'immagine `example-books-microservice:1.1` ed espone la porta 3000 del container.

### Creare il servizio su *kubernetes*

Per poter vedere il deployment dall'esterno, bisogna creare un servizio. I servizi possono essere di diverso tipo:

- ***ClusterIP***: E' visibile solo all'interno del kluster ed ha un proprio indirizzo IP interno al cluster.
- ***NodePort***: Assegna una porta statica dell'host su cui sta girando il cluster
- ***LoadBalancer***: Espone il servizio tramite un bilanciamento del carico di rete esterno.
- ***ExternalName***: Permette di mappare un servizio Kubernetes a un nome di dominio esterno.
- ***Ingress***: L'Ingress è un oggetto Kubernetes che gestisce il traffico HTTP e HTTPS esterno al cluster verso i servizi all'interno del cluster.

Anche per creare il servizio che rende visibile il nostro deployment all'esterno del container possiamo procedere in diversi modi:

- farlo direttamente da terminale se ci si sta appoggiando ad un REGISTRY DOCKER

```
kubectl expose deployment example-books-deployment --type=NodePort --port=3000
```

- creare i file di configurazione YAML per il deploy ed applicarlo su kubernetes

```
kubectl apply -f example-books-service.yaml
```

Il file del service di tipo *NodePort* contiene tre porte che indicano rispettivamente:

- ***port***: La porta di kubernetes
- ***targetPort***: La porta interna al container (es.:3000)
- ***nodePort***: La porta esposta sull'host (es.:30001)

### Verificare il funzionamento

Per verificare il funzionamento possiamo accedere alla pagina `http://localhost:30001/api-docs/` del browser e vedere le API esposte.

## Scaling

Una volta esposto il nostro container su *Kubernetes* tramite il servizio, è possibile decidere di scalare il container. Anche questa operazione può essera effettuata

- direttamente da kubectl

```
kubectl scale deployment example-books-microservice -n course-kubernetes --replicas=3
```

- attraverso modifiche al file yaml

```
...
spec:
replicas: 1
...
```

Inoltre si possono impostare delle regole per far gestire in automatico lo *scaling* a kubernets. Per farlo però bisogna utilizzare una versione di *KUBERNETES* che supporta l'autoscaling e bisogna attivare le metriche su KUBERNETES. Per esempio si può impostare lo scaling a seconda dell'utilizzo della *cpu* o il *numero di richieste* o la memoria utilizzata.

Una volta effettuato lo *scale* del deployment, se noi controlliamo ci accorgiamo che adesso ci sono 3 POD running:

kubectl scale deployment example-books-microservice -n course-kubernetes --replicas=3 deployment.apps/example-books-microservice scaled.

```
...:\kubectl get pods -n course-kubernetes
NAME                                        READY   STATUS    RESTARTS   AGE
example-books-microservice-5d6db6cdd8-h29z9   1/1     Running   0          67m
example-books-microservice-5d6db6cdd8-vdgzv   1/1     Running   0          6m33s
example-books-microservice-5d6db6cdd8-vzm4t   1/1     Running   0          6m33s
```