Desplegando servicio Java en Kubernetes, 1/2

1. Objetivo

- ❏ Desplegar un servicio Java en Kubernetes
- ❏ Todo desde cero
- ❏ Todo en local
- ❏ Aprender kubernetes básico

2. Requisitos

- ✓ Linux (ni idea si Windows funciona)
- ✓ Cuenta en Hub Docker
- ✓ Kubectl instalado (tool linea de comando)

3. Instalar Java (con Sdkman)

```
curl -s "https://get.sdkman.io" | bash
source \"${SDKMAN_DIR}/bin/sdkman-init.sh
sdk install java
sdk install micronaut
```

4. Crear HelloWorld

```
mn create-app --build gradle com.incsteps.demo.helloworld
cd helloworld
```

5. Configurar imagen Docker

Editar `build.gradle` y configurar nuestro namespace

build.gradle

```
// añade:
tasks.named("dockerBuild") {
    images = ["incsteps/helloworld"] //(1)
}
<1> cambia incsteps por tu namespace de Docker Hub
```

6. Controller

src/main/java/com/incsteps/demo/HelloController.java

```
package com.incsteps.demo;

import java.util.Map;
import io.micronaut.http.MediaType;
import io.micronaut.http.annotation.Controller;
import io.micronaut.http.annotation.Get;
import io.micronaut.http.annotation.Produces;

@Controller("/")
public class HelloController {
    @Get
    public Map<String,String> index() {
        return Map.of("result","Hello World");
    }
}
```

7. Publicar la imagen

```
./gradlew clean dockerPush
```

Desplegando servicio Java en Kubernetes, 2/2

incsteps.com

1. Crear Cluster (en local)

```
curl -s \
https://raw.githubusercontent.com/k3d-io/k3d/main/install.sh | bash
k3d cluster create -p "8081:80@loadbalancer" my-cluster
```

el puerto 8081 **local** se "linka" con el 80 **interno** del cluster

2. Deployment

Casi siempre desplegamos una aplicación y un servicio para acceder a ella

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "hello"
spec:
  selector:
    matchLabels:
      app: "hello"
  template:
    metadata:
      labels:
        app: "hello"
    spec:
      containers:
        - name: "hello"
          image: incsteps/helloworld #(1)
          ports:
            - name: http
              containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: "hello"
spec:
  selector:
    app: "hello"
  type: NodePort
  ports:
    - protocol: "TCP"
      port: 8080
```

1. Utiliza tu namespace

3. Ingress (para acceder desde la "calle")

ingress.yml

```
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: hello
spec:
  entryPoints:
    - web
  routes:
    - match: PathPrefix(`/hello/`)
      kind: Rule
      services:
        - name: hello
          port: 8080
      middlewares:
        - name: hello
---
apiVersion: traefik.containo.us/v1alpha1
kind: Middleware
metadata:
  name: hello
spec:
  stripPrefix:
    prefixes:
      - /hello/
```

(k3d usa traefik como LoadBalancer, otros proveedores usan otros)

4. Probando nuestro servicio

Abre un navegador en http://localhost:8081/hello/

Authors :

https://www.linkedin.com/in/jagedn
Mentors juniors by telling old "war" stories

Ir al repositorio de pildoras