



3.2 – Contenedores y Orquestadores

Tema 7 – Herramientas de desarrollo en Kubernetes




Universidad
Rey Juan Carlos

Micael Gallego
micael.gallego@urjc.es
@micael_gallego



About Me

Pablo Chico de Guzmán

-  @pchico83
- Docker & Cloud-Native Meetups
- 4 years working @Docker
- Founder & CTO
- Remote Development Environments
- <https://okteto.com>



Herramientas de desarrollo en Kubernetes

- Introducción
- Desarrollo con Build, Push, Redeploy
- Desarrollo con File Sync
- Desarrollo con Network Bridge
- Otras herramientas de desarrollo
- Conclusiones

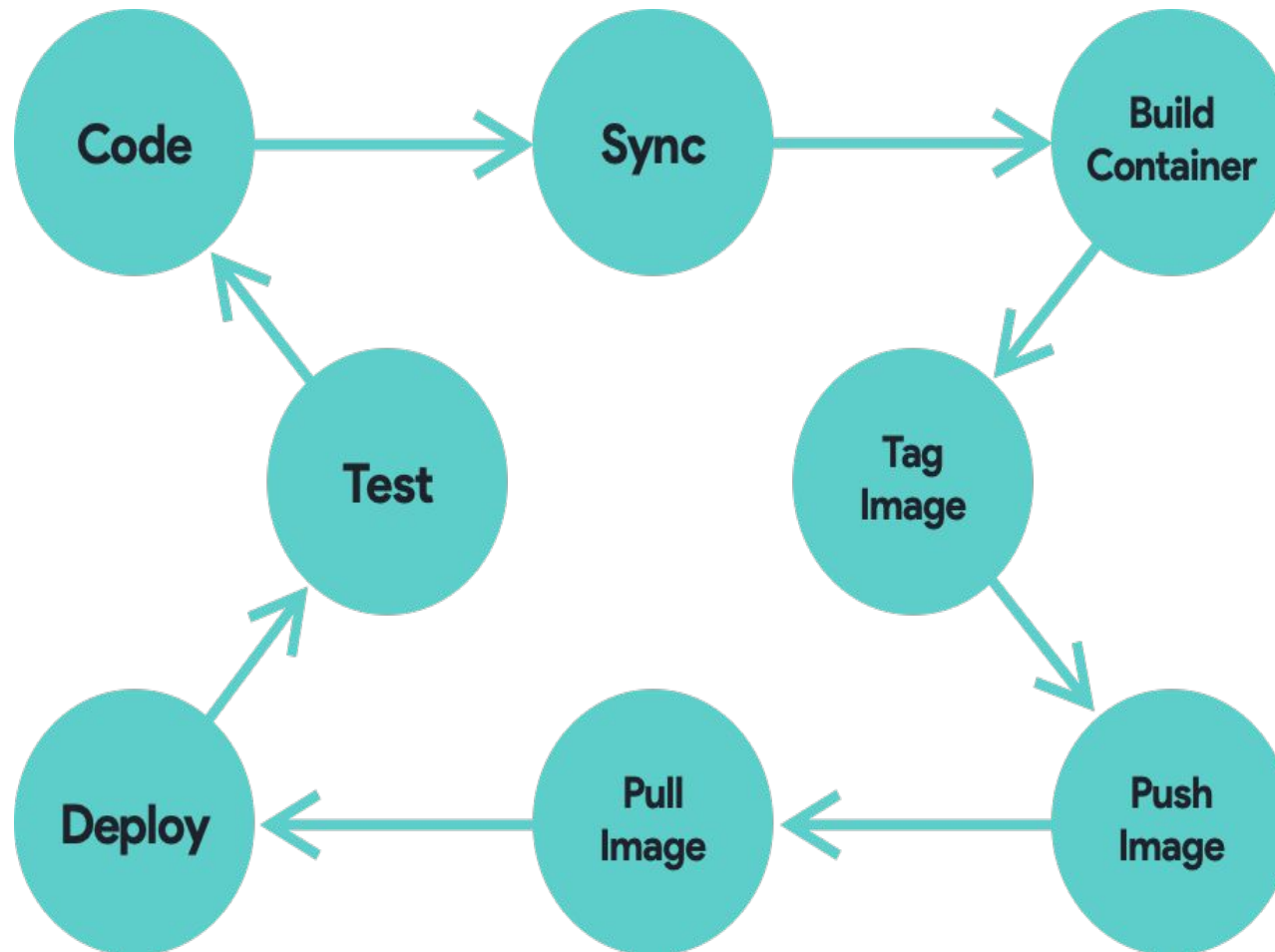
Herramientas de desarrollo en Kubernetes

- Introducción
- Desarrollo en Java
- Desarrollo con Build, Push, Redeploy
- Desarrollo con File Sync
- Desarrollo con Network Bridge
- Otras herramientas de desarrollo
- Conclusiones

- ¿Desplegar en Kubernetes al desarrollar?
 - Aplicaciones nativas de Kubernetes
 - Acceso a la Kubernetes API
 - Acoplada con Config Maps, Secretos, Volúmenes
 - Microservices, Service Discovery de Kubernetes
 - Networking, Service mesh
 - Aplicaciones que consumen muchos recursos

Introducción

- ¿Cómo desplegar en Kubernetes durante el desarrollo?

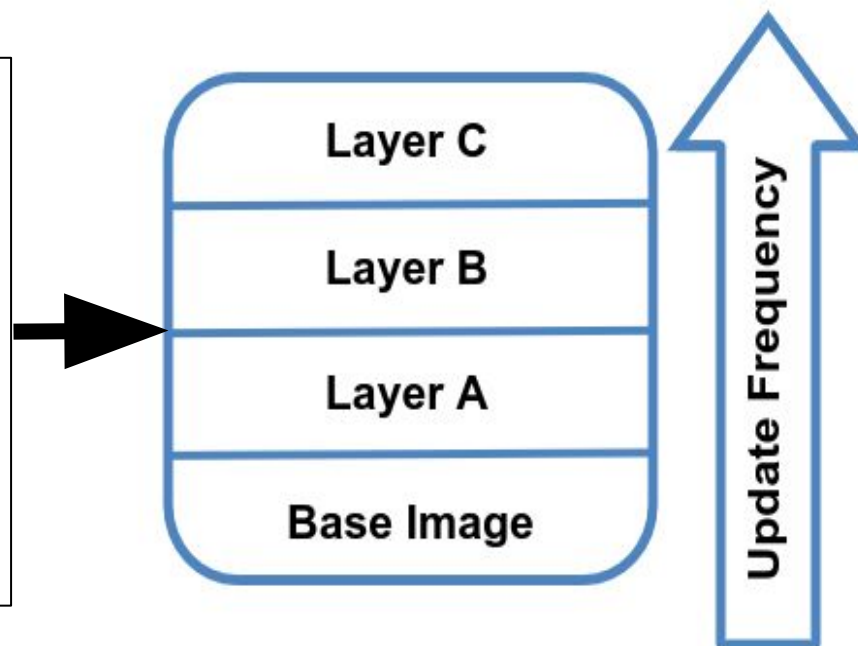


Introducción

- ¿Cómo acelerar el ciclo de Build/Push/Redeploy?

Dockerfile

```
FROM python:3-alpine  
  
ADD requirements.txt requirements.txt  
RUN pip install -r requirements.txt  
  
ADD . /src  
  
CMD ["python", "app.py"]
```



Introducción

- Todos los ejemplos se pueden encontrar en el repositorio

<https://github.com/pchico83/master-cloud-apps>

```
$ git clone https://github.com/pchico83/master-cloud-apps
```

- Concretamente dentro de la carpeta devtools

```
$ cd master-cloud-apps/devtools
```


Herramientas de desarrollo en Kubernetes

- Introducción
- **Desarrollo con Build, Push, Redeploy**
- Desarrollo con File Sync
- Desarrollo con Network Bridge
- Otras herramientas de desarrollo
- Conclusiones

Desarrollo en Java

- **Desarrollo en un IDE sin Kubernetes**
 - Java es un lenguaje que necesita compilación
 - Los IDEs permiten compilación incremental (muy rápida)
 - **Durante la edición:** Para ayudar al developer (autocompletar y errores)
 - **Al guardar el fichero:** Genera los binarios (.class) en disco

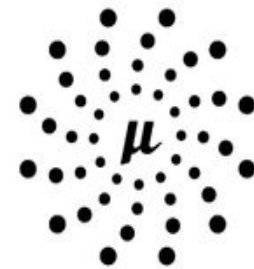


Desarrollo en Java

- **Desarrollo en un IDE sin Kubernetes**
 - Si se usa un IDE la experiencia es “similar” a los lenguajes de script
 - Se guarda el fichero y está compilado de forma inmediata
 - No hay esperas por el proceso de compilación
 - Los frameworks se recargan automáticamente ante cambios (Hot Reload)



QUARKUS



MICRONAUT

- **Compilación y construcción de un contenedor**

Dockerfile

```
FROM maven:3.6-jdk-8 as builder
COPY . /code/
WORKDIR /code
RUN mvn package

FROM openjdk:8-jre
COPY --from=builder /code/target/*.jar /usr/app/
WORKDIR /usr/app
CMD [ "java", "-jar", "demoservice-0.0.1-SNAPSHOT.jar" ]
```

```
$ okteto build -t okteto.dev/demoservice1 .
```

- Despliegue en Kubernetes

```
$ kubectl apply -f k8s
```

```
$ kubectl get deployments,services
```

NAME		READY	UP-TO-DATE	AVAILABLE	AGE
deployment.extensions/demoservice1	0/1	1	0	3m1s	

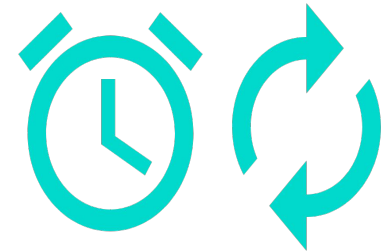
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/demoservice1	ClusterIP	10.104.15.250	<none>	8080/TCP	31s

Desarrollo en Java

- Añadir un cambio en el servicio

- Compilación del código Java
 - Descarga librerías Maven
 - Construye el fat jar desde cero
- Construcción del contenedor con un nuevo fichero de 16Mb
- Publicación del contenedor en un registro (transferencia de red)
- Reinicio del deployment en Kubernetes (si el tag es latest)

demosevice1

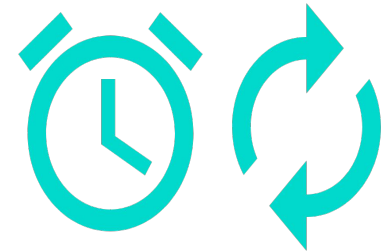


```
$ kubectl rollout restart deployment demosevice1
```

Desarrollo en Java

demoservice2

- Añadir un cambio en el servicio
 - Compilación del código Java
 - Construye el fat jar desde cero
 - Construcción del contenedor con un nuevo fichero de 16Mb
 - Publicación del contenedor en un registro (transferencia de red)
 - Reinicio del deployment en Kubernetes (si el tag es latest)



```
$ kubectl rollout restart deployment demoservice1
```

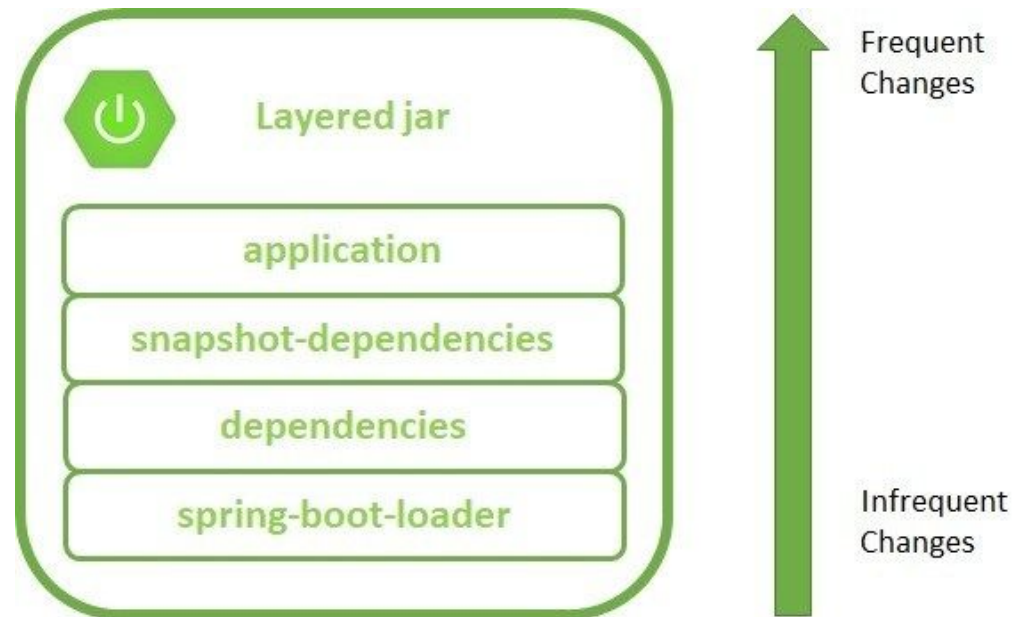
Desarrollo en Java

- ¿Optimizar las capas de la imagen con Java?
 - No crear un fat jar con aplicación y librerías
 - Separar librerías y aplicación en capas diferentes
 - Cuando cambia la aplicación se transfieren los .class de la aplicación
 - Extra bonus! La aplicación arrancará más rápido
- Packaging: an exploded jar with the application's own main is always faster

<https://spring.io/blog/2018/12/12/how-fast-is-spring>

Desarrollo en Java

- A partir de Spring Boot 2.3.0
 - JAR con **capas**
 - Soporte de **Buildpacks**



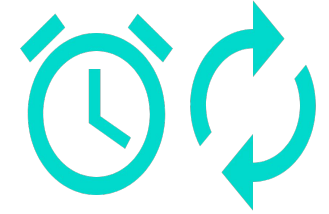
- Dockerfile con capas

Dockerfile

```
FROM adoptopenjdk:11-jre-hotspot as builder
WORKDIR application
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} application.jar
RUN java -Djarmode=layertools -jar application.jar extract
```

```
FROM adoptopenjdk:11-jre-hotspot
WORKDIR application
COPY --from=builder application/dependencies/ ./
COPY --from=builder application/snapshot-dependencies/ ./
COPY --from=builder application/resources/ ./
COPY --from=builder application/application/ ./
ENTRYPOINT ["java", "org.springframework.boot.loader.JarLauncher"]
```

Desarrollo en Java



- Haciendo uso de Buildpacks
 - Compilar, crear imagen y subir a DockerHub

```
$ mvn spring-boot:build-image \  
-Dspring-boot.build-image.imageName=codeurjc/demoservice-k8s2:latest
```

```
$ docker push codeurjc/demoservice-k8s2:latest
```

```
$ kubectl apply -f k8s
```

- Realizar un cambio en el servicio y actualizar la imagen

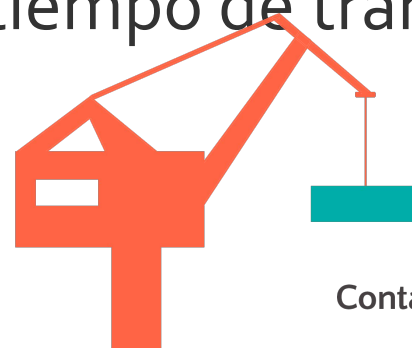
```
$ mvn spring-boot:build-image \  
-Dspring-boot.build-image.imageName=codeurjc/demoservice-k8s2:latest
```

```
$ docker push codeurjc/demoservice-k8s2:latest
```

```
$ kubectl rollout restart deployment demoservice
```

Desarrollo en Java

- Haciendo uso de jib
 - jib es un **plugin de Maven y Gradle** que empaqueta aplicaciones Java directamente como contenedores Docker (**sin generar el .jar**)
 - Las capas **optimizadas** para cachear librerías
 - Al no generar el .jar **envía sólo los .class** de la aplicación (muy poco tamaño > poco tiempo de transferencia)
 - La aplicación **arranca más rápido** (*exploded jar*)



Jib

Containerize your Java application.

<https://github.com/GoogleContainerTools/jib>

Desarrollo en Java

- **jib no necesita el docker engine** para generar las imágenes Docker, todo lo hace con Java
- **Aumenta la seguridad** en el entorno de CI porque no necesita permisos de administración (necesarios para Docker) para crear una imagen
- La imagen está **optimizada para ejecutar aplicaciones Java** (*distroless*)

<https://github.com/GoogleContainerTools/distroless>

<https://github.com/GoogleContainerTools/jib>

- Haciendo uso de jib

pom.xml

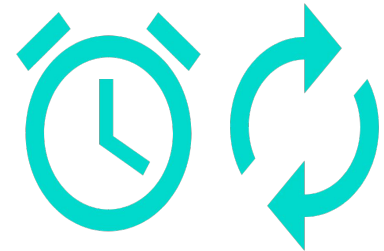
```
<project>
  ...
  <build>
    <plugins>
      <plugin>
        <groupId>com.google.cloud.tools</groupId>
        <artifactId>jib-maven-plugin</artifactId>
        <version>3.0.0</version>
      </plugin>
    </plugins>
  </build>
</project>
```

```
$ ./mvnw compile jib:build -Dimage=codeurjc/demoservice-k8s3:latest
```

```
$ kubectl apply -f k8s
```

Desarrollo en Java

demomervice3



- Automatizar pasos en un comando
 - Compilación del código Java
 - Construye el fat jar desde cero
 - Construcción del contenedor con un nuevo fichero de 16Mb
 - Publicación del contenedor en un registro (transferencia de red)
 - Reinicio del deployment en Kubernetes (si el tag es latest)

```
$ okteto deploy --build
```

Herramientas de desarrollo en Kubernetes

- Introducción
- Desarrollo en Java
- **Desarrollo con File Sync**
- Desarrollo con Network Bridge
- Otras herramientas de desarrollo
- Conclusiones

- ¿Podemos optimizar más aún el proceso?
- Si tuviéramos un compilador Java en el pod podríamos enviar los cambios en los **ficheros fuente** que han sido actualizados
- Con **hot reload** el cambio en el fuente lanzaría la compilación y la actualización del servicio en el pod

File Sync

- Envía código de tu host al pod del cluster
- **Sustituye el pod** de la aplicación por otro pod con Maven (compilador)
- **Ejecuta la aplicación** de nuevo en ese contenedor (mvn spring-boot:run)
- **Sincroniza los ficheros** del host a ese pod de forma automática



<https://okteto.com/>

File Sync

- Instalar **Okteto CLI**
- Desplegamos en Kubernetes

```
$ kubectl apply -f k8s
```

- Creamos el fichero **“okteto.yml”** si no existe

```
$ okteto init
```

- Si el fichero **“okteto.yml”** iniciaremos la sincronización usando:

```
$ okteto up
```

File Sync

demoservice4

okteto.yml

```
name: demoservice
image: maven:3.8.1-jdk-11
command:
  - mvn
  - spring-boot:run
volumes:
  - /root/.m2
forward:
  - 8080:8080
```

Sustituye el pod del deployment
demoservice-dply...

...por una imagen con maven

Ejecuta la aplicación

Guarda la caché de maven en
volúmen

Expone la app en localhost

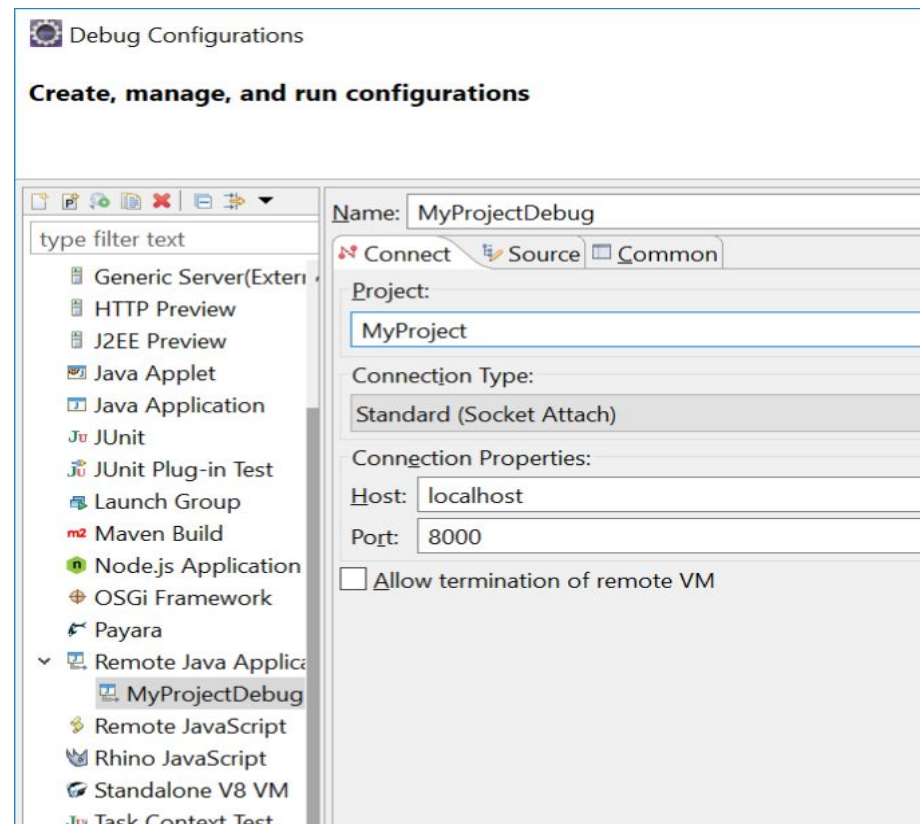
- Fichero “.stignore”
 - Permite ignorar **ficheros** o **carpetas** en la **sincronización**
 - Con el comando “okteto init” se crea un fichero “.stignore” por defecto

<https://www.okteto.com/docs/reference/file-synchronization/>

Port Forwards

demoservice5

- Depuración
 - Los IDEs se puede conectar a una JVM para hacer depuración remota



Port Forwards

demomervice5

- Depuración y LiveReload
 - VSCode también se puede conectar a una JVM para hacer depuración remota

.vscode/launch.json

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "java",
      "name": "demomervice5",
      "request": "attach",
      "hostName": "localhost",
      "port": 5005
    }
  ]
}
```

Okteto File Synchronization

- Podemos configurar la aplicación para que exporte el puerto **5005** para depuración remota desde **IntelliJ** o **VSCode**
- Extra: las dev-tools expone el puerto **35729** para el plugin Remote LiveReload del navegador

pom.xml

```
...  
<plugin>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-maven-plugin</artifactId>  
  <configuration>  
    <jvmArguments>  
      -agentlib:jdwp=transport=dt_socket,server=y,su  
spend=n,address=5005  
    </jvmArguments>  
  </configuration>  
</plugin>  
...
```

okteto.yml

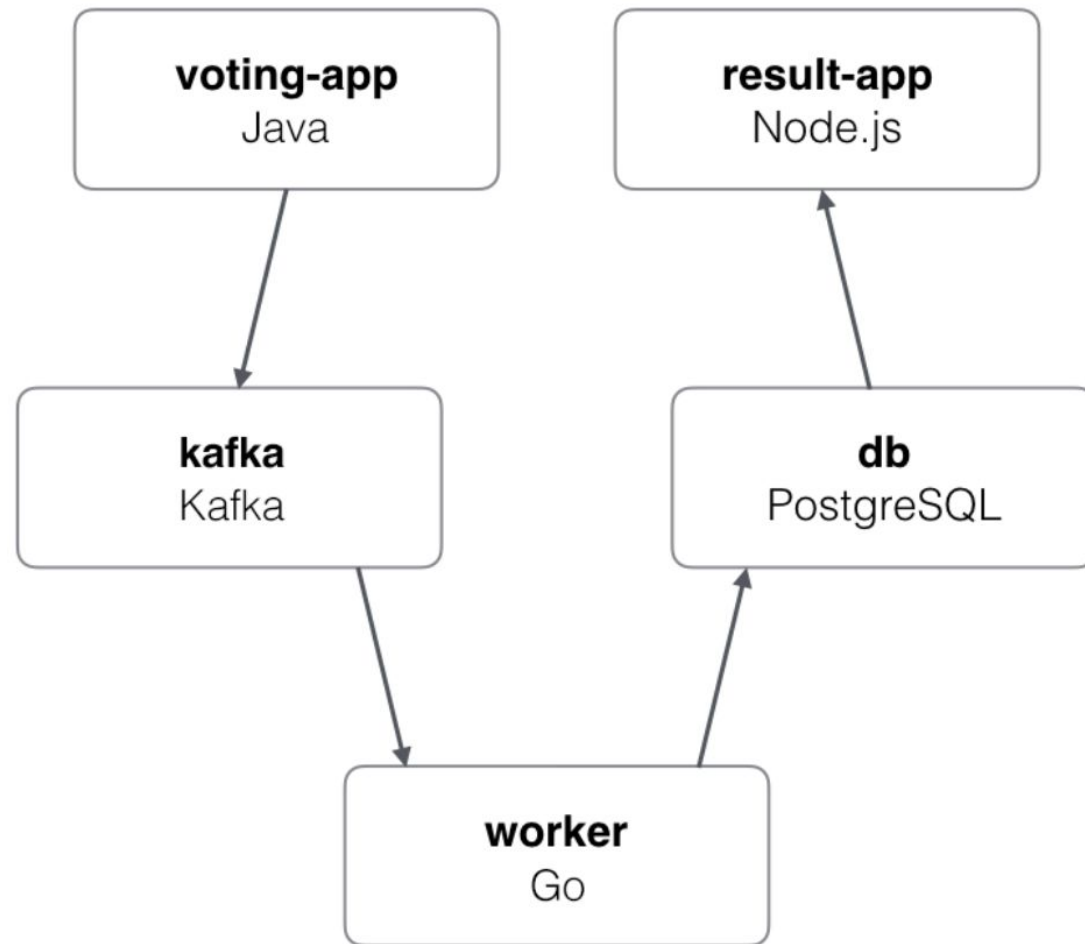
```
name: demomervice  
image: maven:3.8.1-jdk-11  
command:  
  - mvn  
  - spring-boot:run  
volumes:  
  - /root/.m2  
forward:  
  - 8080:8080  
  - 5005:5005
```


File Sync



Voting App

demosservice6



Herramientas de desarrollo en Kubernetes

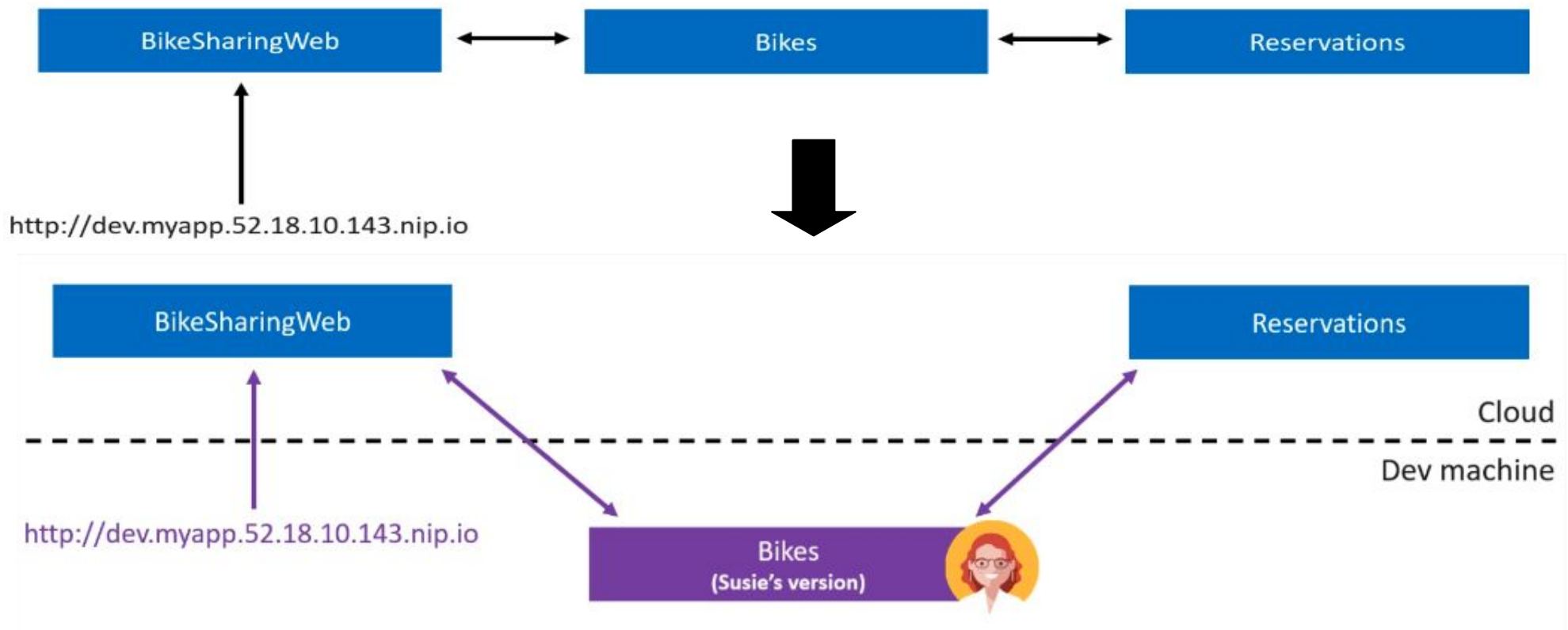
- Introducción
- Desarrollo en Java
- Desarrollo con File Sync
- **Desarrollo con Network Bridge**
- Otras herramientas de desarrollo
- Conclusiones

Network Bridge to Kubernetes

- Utiliza tu host para desarrollar
 - **Sustituye el pod** por otro pod que permite la redirección del tráfico entre kubernetes y el host
 - **Conexión directa** entre el host y el cluster
 - **Ejecuta y depura** el código en local directamente

Network Bridge to Kubernetes

- ¿Cómo funciona?
 - De forma predeterminada se redirige todo el tráfico de un servicio a nuestro host



Herramientas de desarrollo en Kubernetes

- Introducción
- Desarrollo en Java
- Desarrollo con File Sync
- Desarrollo con Network Bridge
- **Otras herramientas de desarrollo**
- Conclusiones

Build, Push, Redeploy

- Skaffold:
<https://skaffold.dev/>
- Garden:
<https://garden.io/>
- Tilt:
<https://tilt.dev/>

File Sync

- Okteto:
<https://www.okteto.com/>
- DevSpace:
<https://devspace.sh/>
- DevPod:
<https://jenkins-x.io/docs/reference/devpods/>

Network Bridge

- Telepresence:
<https://www.telepresence.io/>
- kubefwd:
<https://github.com/txn2/kubefwd>
- Velocity:
<https://velocity.tech/>

Herramientas de desarrollo en Kubernetes

- Introducción
- Desarrollo en Java
- Desarrollo con File Sync
- Desarrollo con Network Bridge
- Otras herramientas de desarrollo
- **Conclusiones**

Conclusiones

- Podemos desarrollar nativamente en Kubernetes sin necesidad de hacer Build/Push/Redeploy
- Existen muchas herramientas para el desarrollo nativo con diferentes estrategias
 - Desarrollar directamente en el cluster
 - Desarrollar en local pero conectando el host al cluster
- ¿Qué herramienta elegimos?
La más adecuada para tu flujo de trabajo