



Developing  
Java based microservices  
ready for the world of containers



@davsclaus  
davsclaus  
[davsclaus.com](http://davsclaus.com)



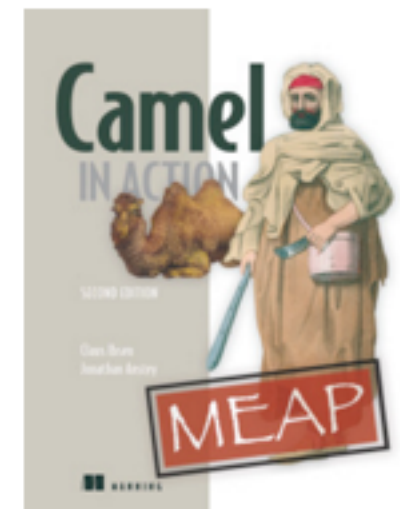
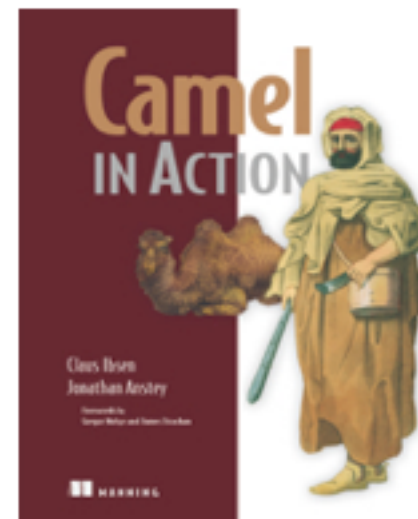
Claus Ibsen

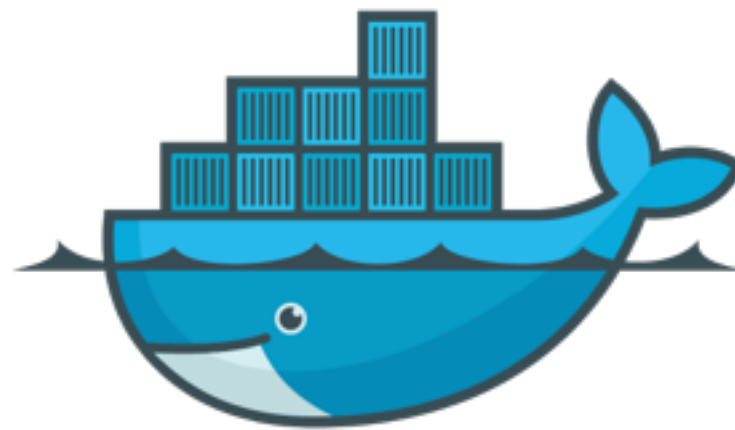
# Claus Ibsen



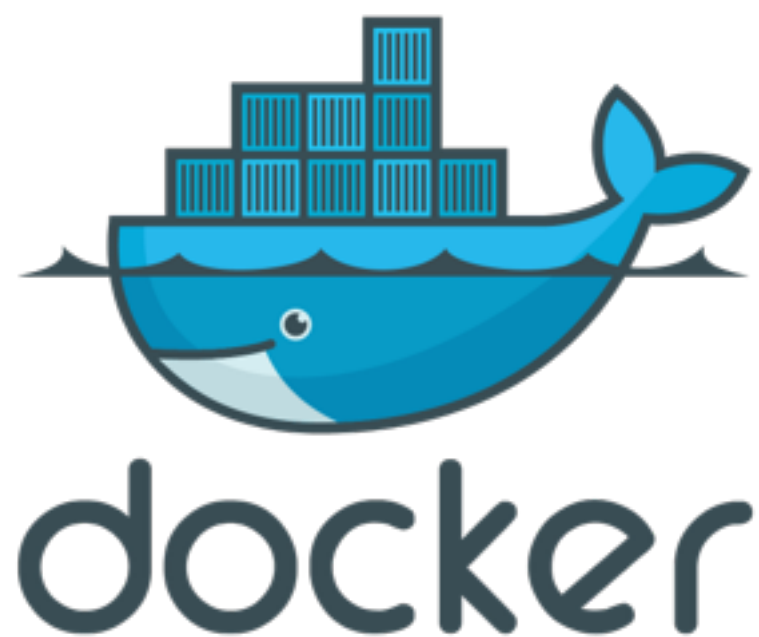
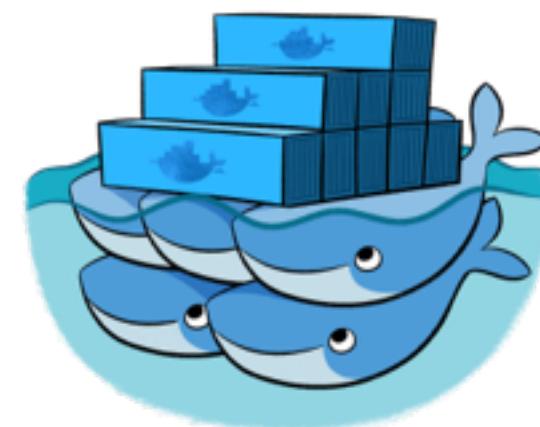
@davsclaus  
davsclaus  
[davsclaus.com](http://davsclaus.com)

- Principal Software Engineer at Red Hat
- Apache Camel  
8 years working with Camel
- Author of Camel in Action books

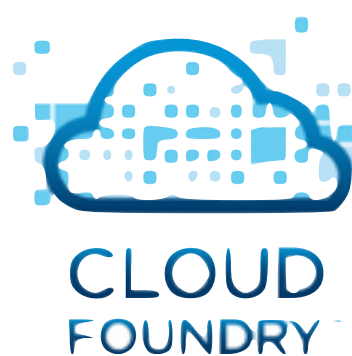




docker

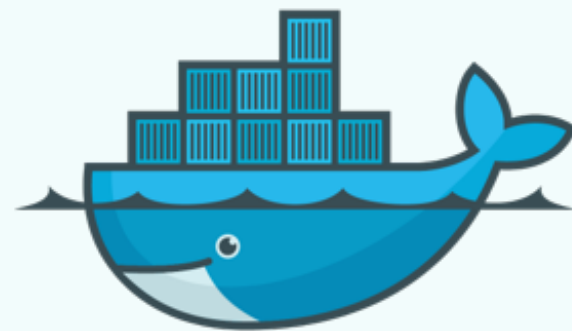


**OPENSIFT**





**OPENSIFT**



**docker**



OPENSIFT



**fabric8**



docker



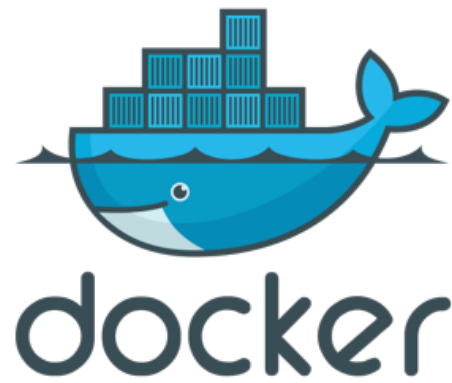
OPENSIFT



**fabric8**



docker



**OS Level Virtualization**



**Docker Orchestration**



**PaaS Platform on top of  
Kubernetes**

**OPENSIFT**



**Services and Tools for  
Kubernetes and OpenShift**



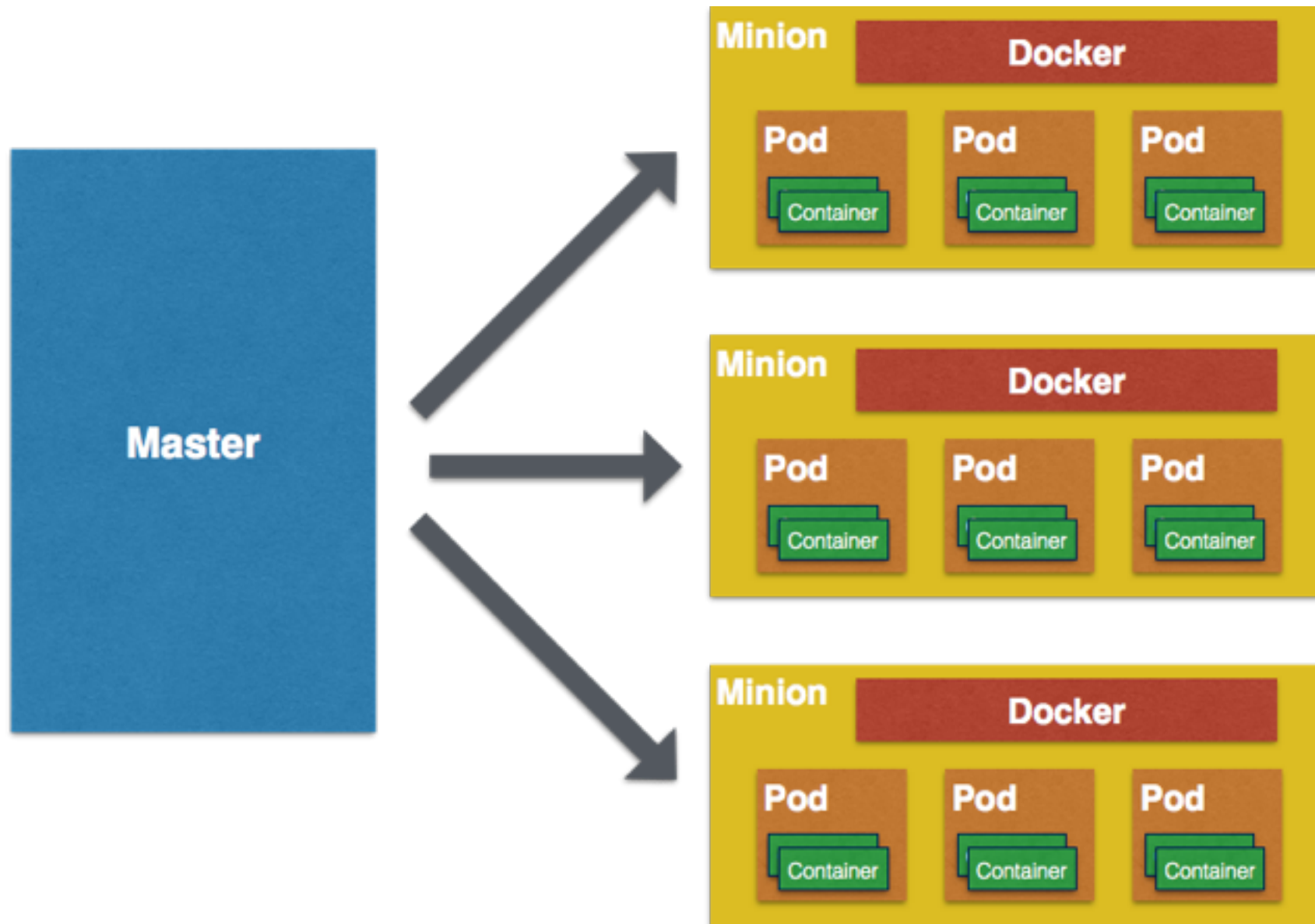
# Kubernetes



# Kubernetes

- ▶ Open Source orchestration platform for Docker containers
  - Rewrite of Google's internal framework "Borg"
- ▶ Declarative specification of a desired state
- ▶ Self-healing
- ▶ Service discovery
- ▶ Scheduling across hosts
- ▶ Replication

# Architecture



# Key Concepts

## ▶ **Nodes**

- Worker machine (physical or VM) running pods

## ▶ **Pods**

- Collection of one or more Docker containers

## ▶ **Replication Controller**

- Creates and takes care of Pods

## ▶ **Services**

- Network Proxy for a collection of Pods

## ▶ **Labels**

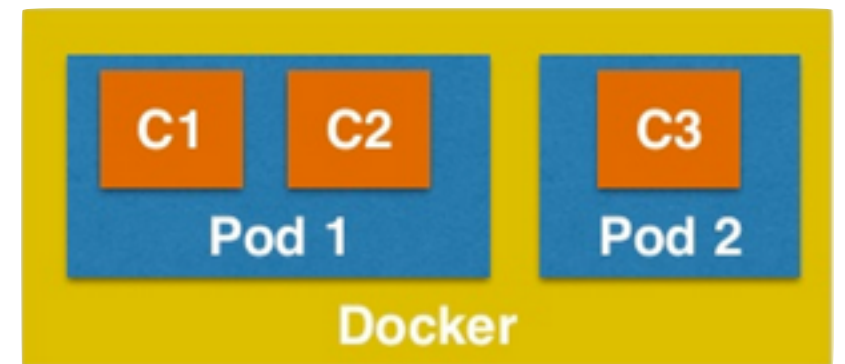
- Grouping and organization of Objects

## ▶ **Namespaces**

- Virtual clusters backend by same physical cluster

# Pod

- ▶ Collection of Docker containers running on the same host.
- ▶ Pods have unique IPs
- ▶ Containers in a Pod ....
  - .... share the same IP address
  - .... can reach each other via local ports
  - .... can share data via volumes
- ▶ Pods can have one or more ***Labels***

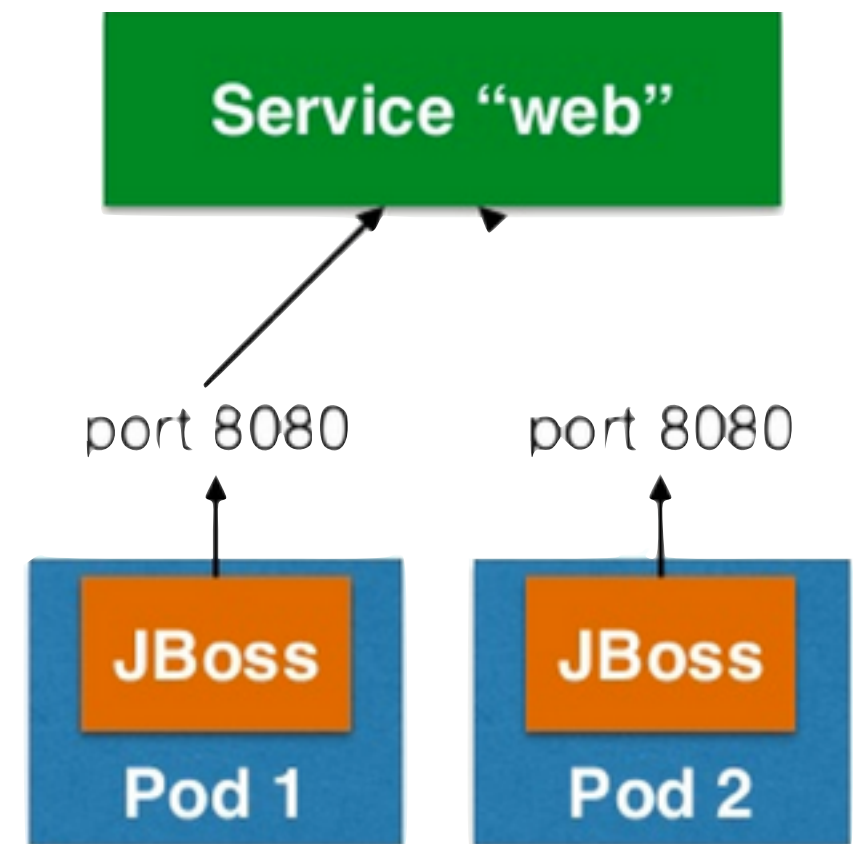


# Replication Controller

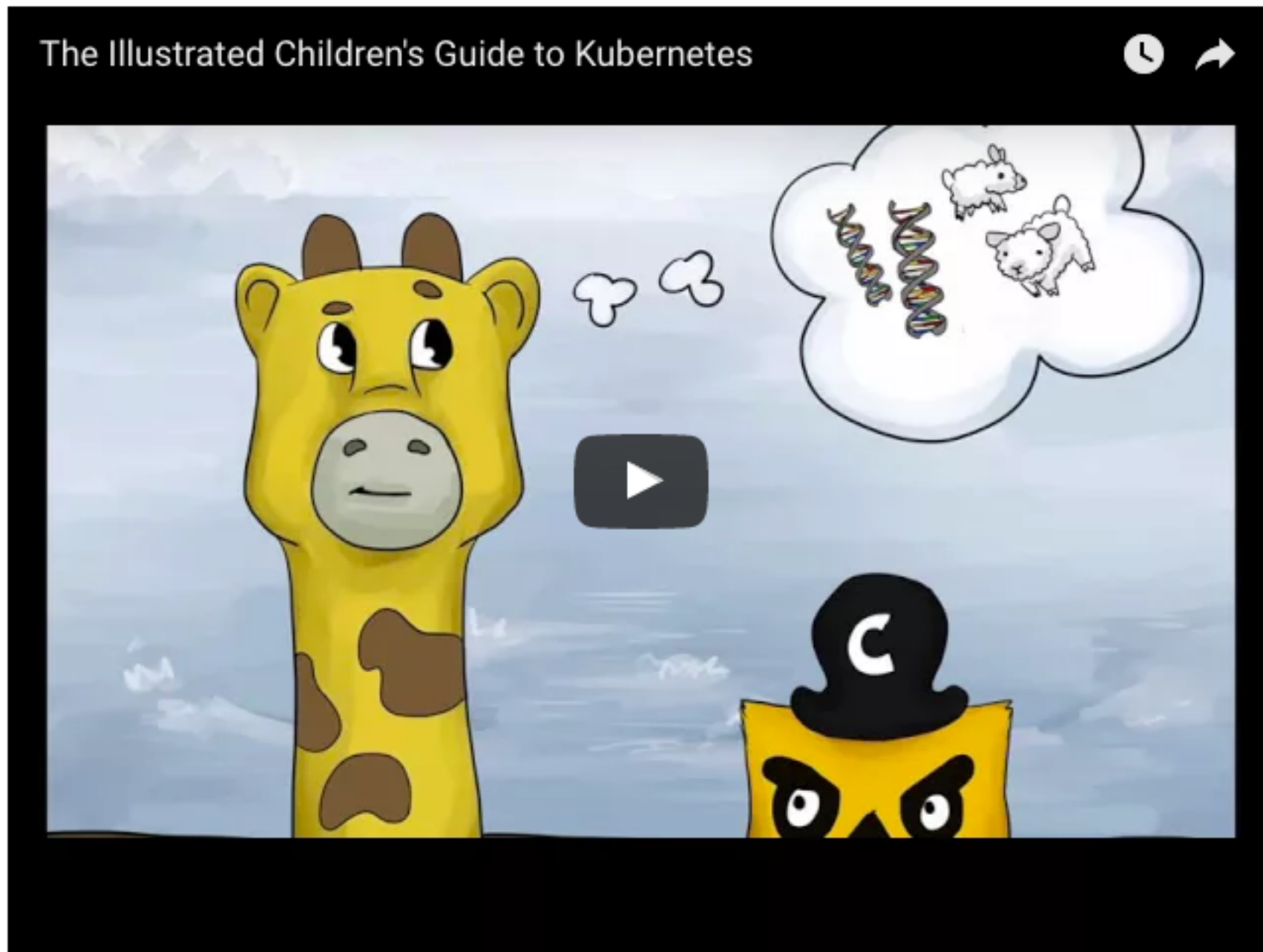
- ▶ Controls Pods selected by **Labels**
- ▶ Ensures that a specified number of Pod replicas is running
- ▶ Holds Pod templates for creating new Pods
- ▶ Autoscaling
- ▶ Rolling Updates

# Service

- ▶ View on a set of Pods with single IP address and port
- ▶ Pods are selected by **Label**
- ▶ Services are referenced by environment variables
- ▶ Service addresses stay stable
  - Pods come and go (with different IPs)



# What is Kubernetes?



<http://blog.kubernetes.io/2016/06/illustrated-childrens-guide-to-kubernetes.html>





# **fabric8**

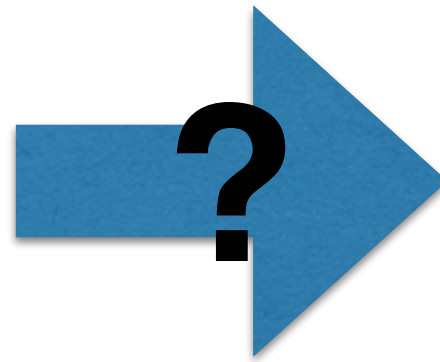
<https://fabric8.io/>

# fabric8

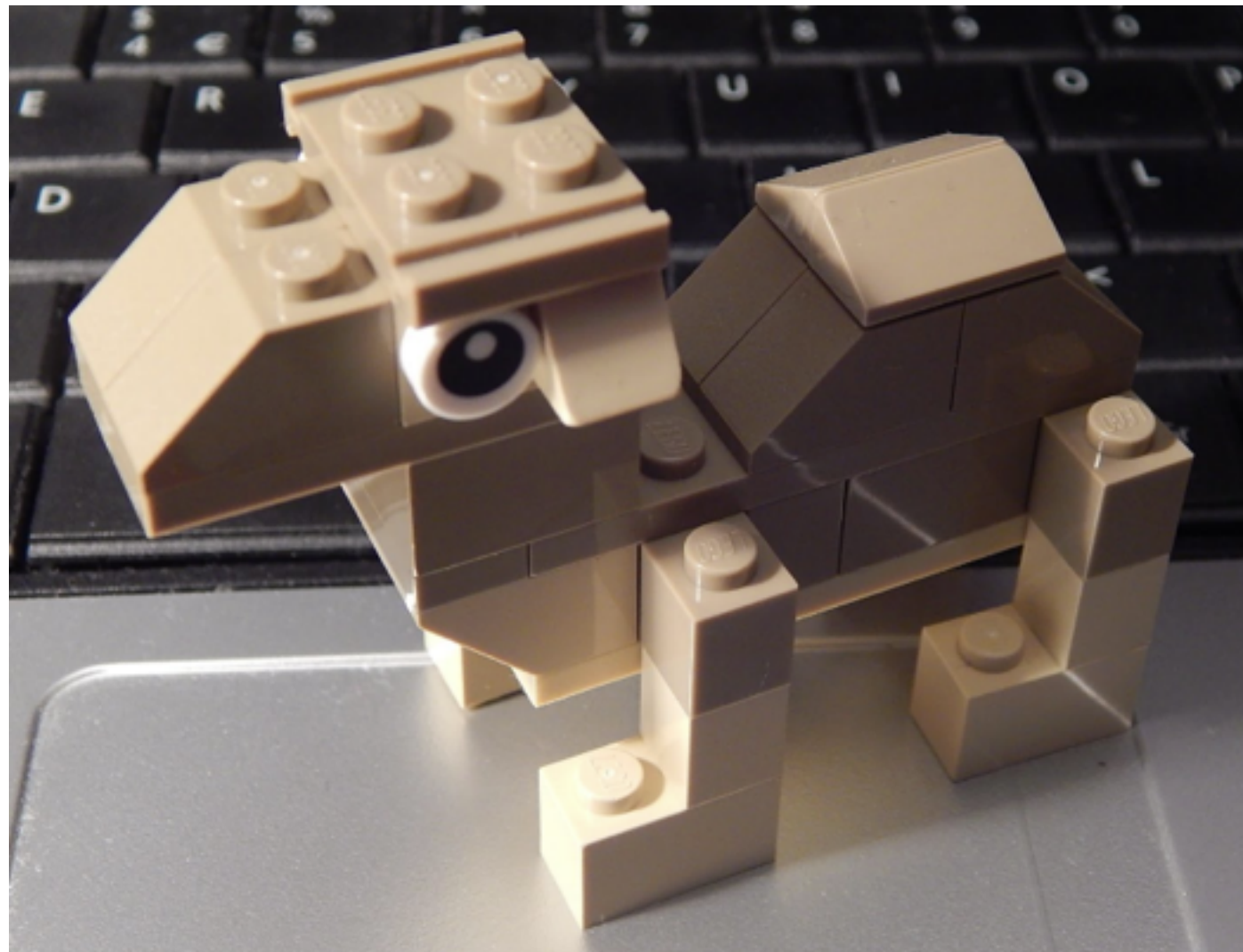
- ▶ Opinionated microservices platform based on Docker, Kubernetes and Jenkins
  - **Management:** console, logging, metrics, dashboards, ...
  - **Continuous Delivery** Pipeline
  - **iPaaS:** Camel route visualization, API registry, Messaging as a Service, ...
  - **Tools:** Kubernetes/OpenShift build integration, Kubernetes component test support, CDI extensions
  - **Quickstarts:** Ready to use examples, also available as Maven archetypes

# Kubernetes Demo Time

# Java Developer



# Build a Camel Demo Time



# Source Code

GitHub, Inc. [US] <https://github.com/davsclaus/fabric8-hello>

## fabric8-hello

Two microservices using Spring Boot and WildFly Swarm with Camel running in kubernetes using

There are two Maven projects:

- client - Spring Boot application with Camel that triggers every 2nd second to call the hello service response.
- helloswarm - WildFly Swarm application hostin a hello service which returns a reply message

The diagram below illustrates this:

```
graph LR; Client[Client] -- HTTP --> Service[Hello Service]; Service --> Client;
```

The diagram illustrates the interaction between two microservices. On the left, a green box represents the 'Client', which is a Spring Boot application. On the right, a blue box represents the 'Hello Service', which is a WildFly Swarm application. A horizontal arrow points from the Client to the Hello Service, labeled 'HTTP'. A second horizontal arrow points from the Hello Service back to the Client, indicating a response.

<https://github.com/davsclaus/fabric8-hello>

# Hello Service

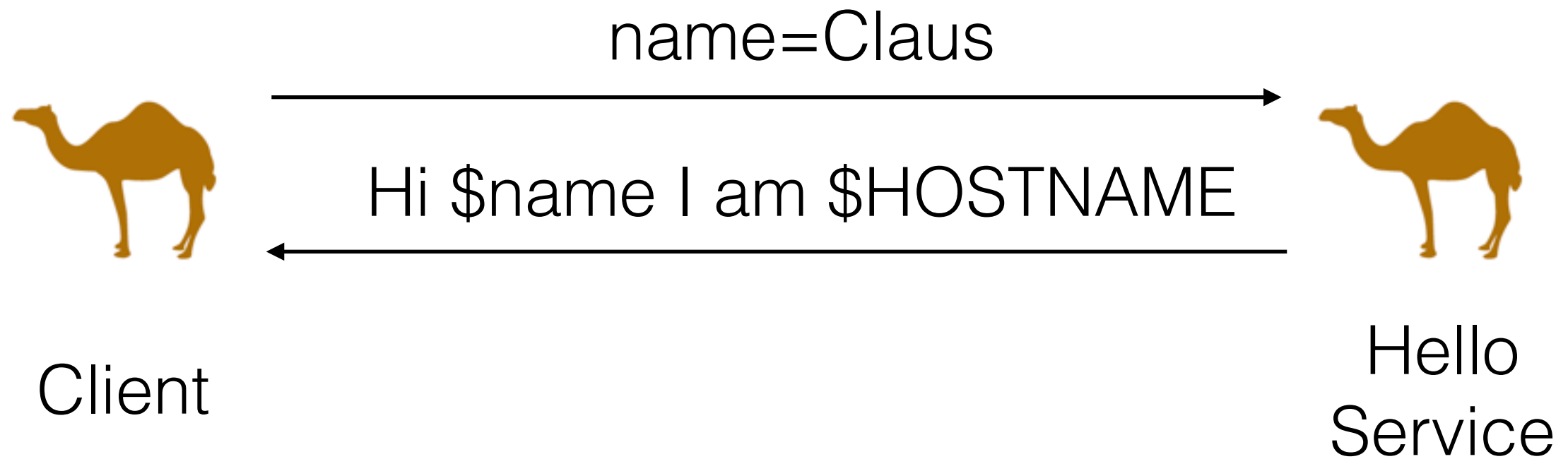


Client



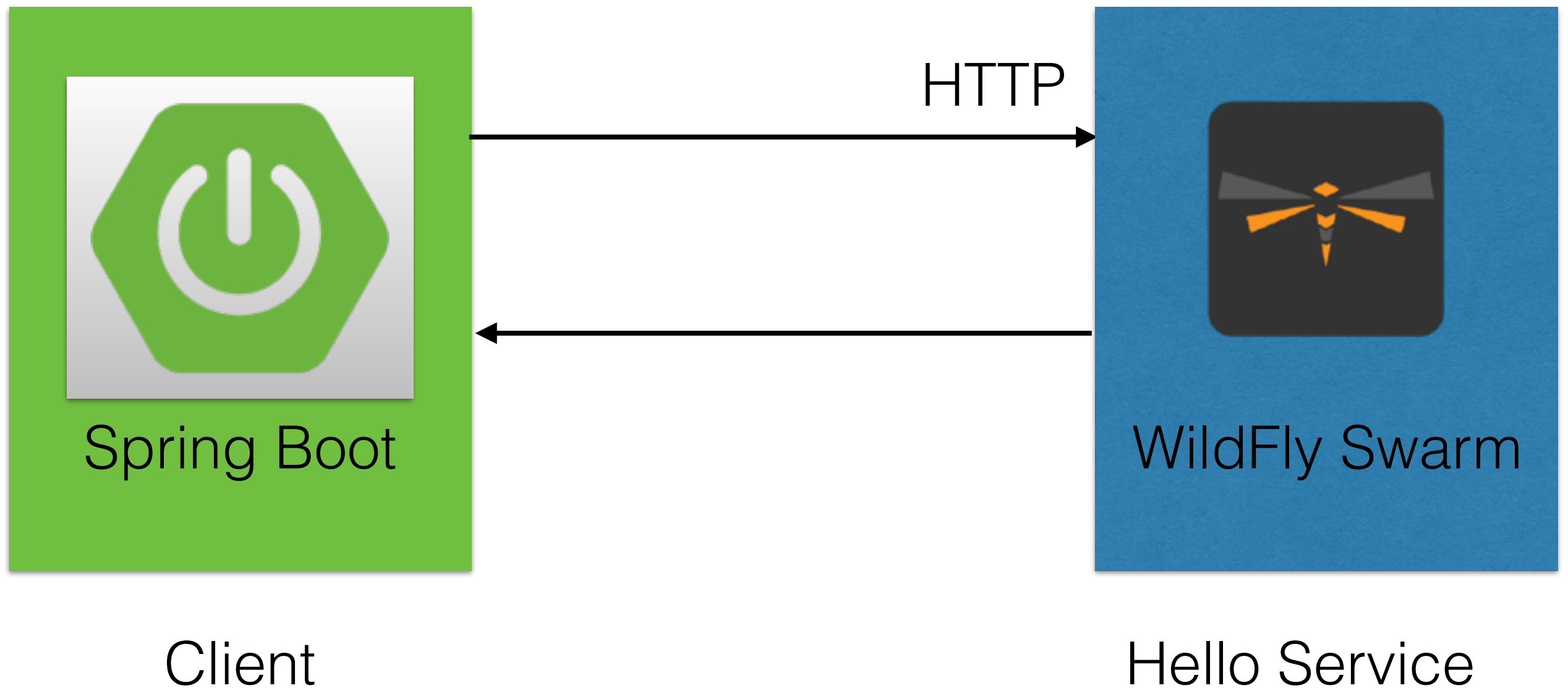
Hello  
Service

# Hello Service

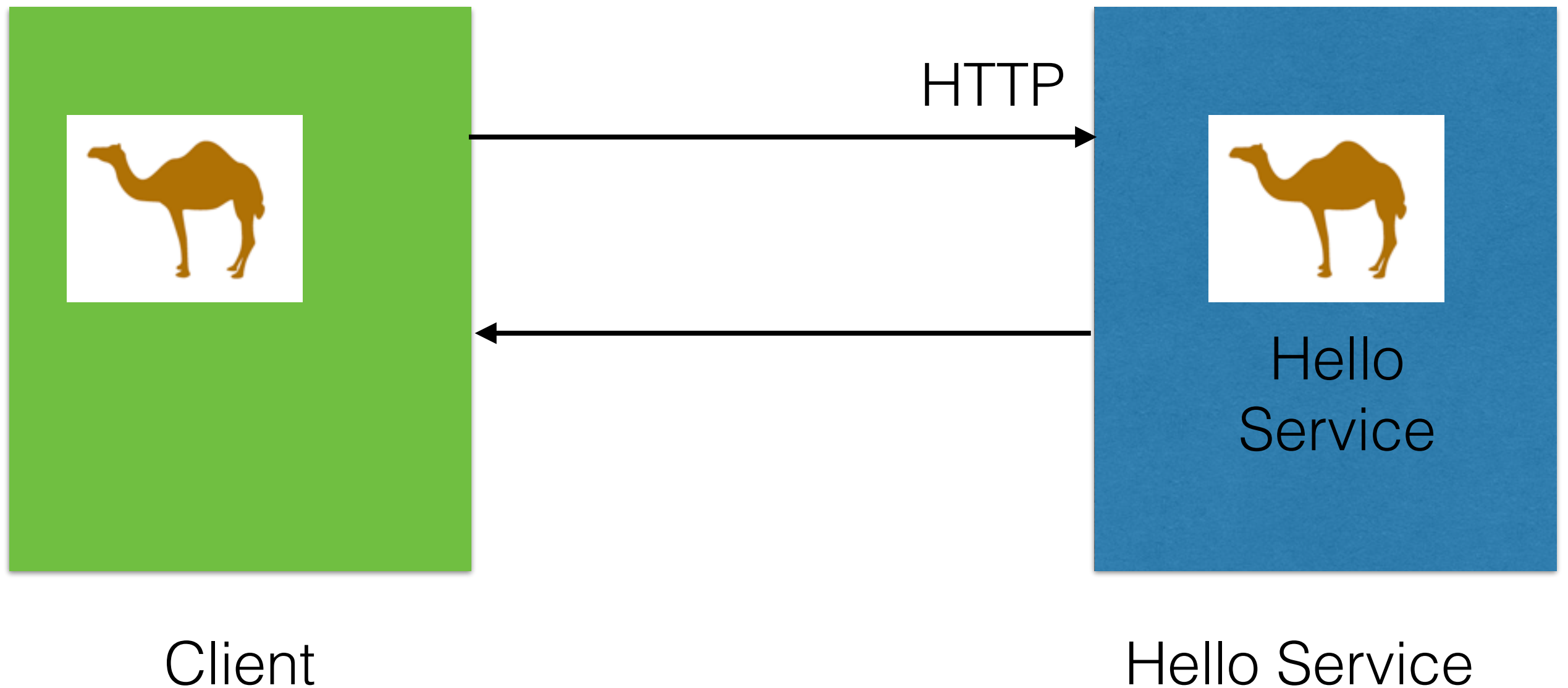




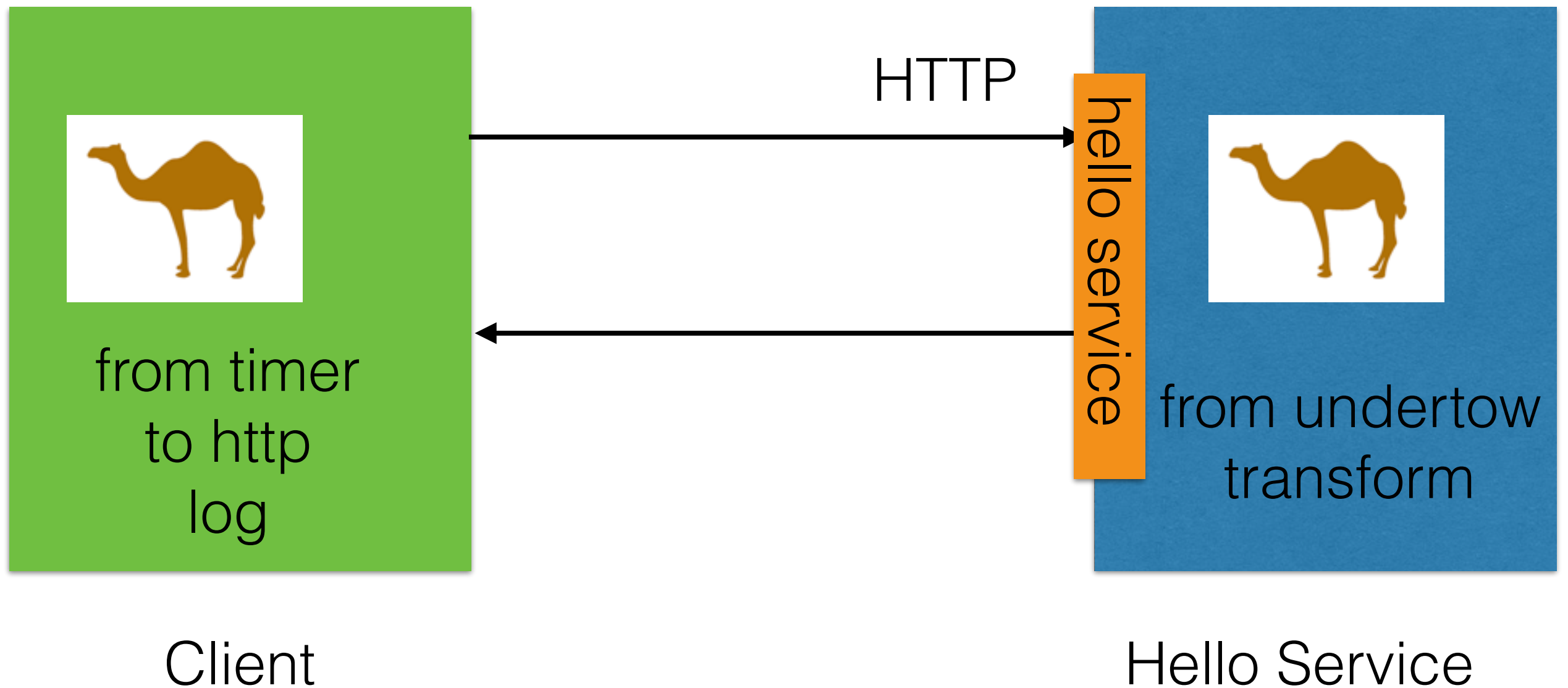
# Implementation



# Implementation



# Implementation



# Spring Boot Starter

start.spring.io

☆

SPRING INITIALIZR bootstrap your application now

Generate a 

Maven Project

 with Spring Boot 

1.4.0

## Project Metadata

Artifact coordinates

Group

com.foo

Artifact

client

## Dependencies

Add Spring Boot Starters and dependencies to your a

Search for dependencies

Web, Security, JPA, Actuator, Devtools..

Selected Dependencies

Web X

Actuator X

Apache Camel X

Generate Project ⌘ + ↵




# Client

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("timer:foo?period=2000")
            .to("netty4-http:http://localhost:8080/hello")
            .log("${body}");
    }
}
```

# WildFly Swarm Generator

 WildFly Swarm **GENERATOR** NEW! **BLOG** **DOCUMENTATION**

## WildFly Swarm Project Generator

Rightsize your Java EE microservice in a few clicks

### Instructions

1. Click on the Generate button to download the *helloswarm.zip* file
2. Unzip the file in a directory of your choice
3. Run `mvn wildfly-swarm:run` in the unzipped directory

**Group ID**

**Artifact ID**

**Generate Project**

**Dependencies**

Not sure what you are looking for? [View all available dependencies](#)

**Selected dependencies**  
**Camel CDI** ✕ **Camel Undertow** ✕



# Hello Service



```
@Singleton
public class HelloRoute extends RouteBuilder {

    @Inject
    @Uri("undertow:http://0.0.0.0:8080/hello")
    private Endpoint undertow;

    @Inject
    private HelloBean hello;

    @Override
    public void configure() throws Exception {
        from(undertow).bean(hello);
    }
}
```

# Hello Service

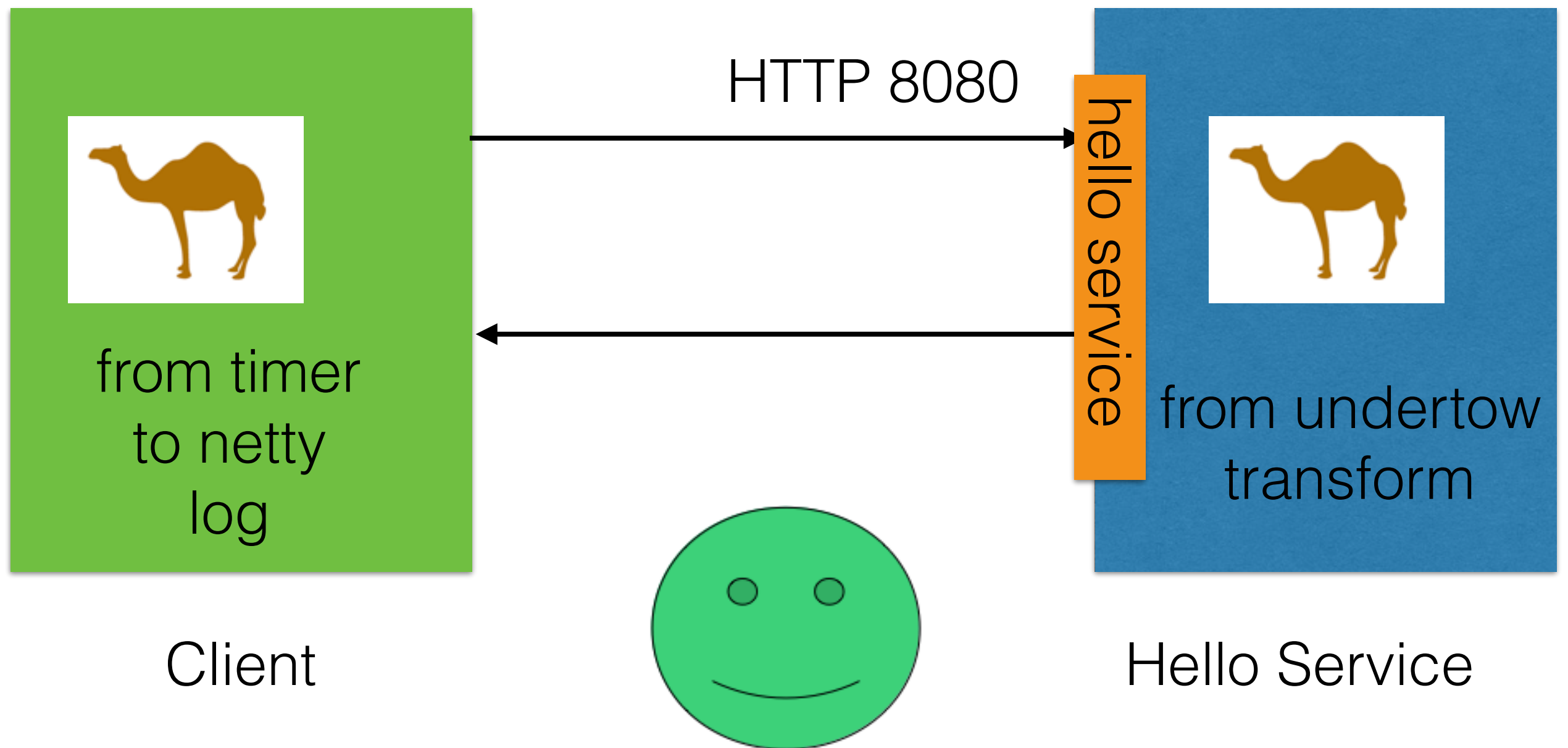


```
@Singleton
public class HelloBean {

    public String sayHello() throws Exception {
        return "Swarm says hello from "
            + InetAddressUtil.getLocalHostName();
    }
}
```



# Ready to run local



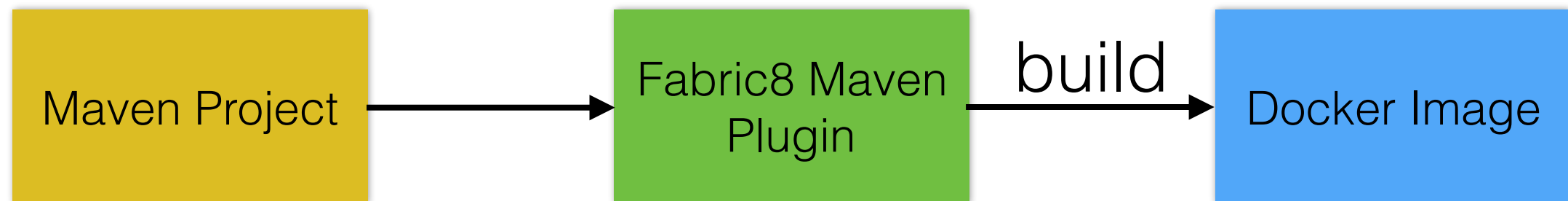
# How to build Docker Image?



Maven Project

Docker Image

# Docker Maven Plugin



<https://maven.fabric8.io>

# Fabric8 Maven Plugin

```
<plugin>  
  <groupId>io.fabric8</groupId>  
  <artifactId>fabric8-maven-plugin</artifactId>  
  <version>3.1.37</version>  
</plugin>
```

<https://maven.fabric8.io>

# Install fabric8-maven-plugin

```
mvn io.fabric8:fabric8-maven-plugin:3.1.37:setup
```

<https://maven.fabric8.io>

# Fabric8 Maven Plugin

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>fabric8-maven-plugin</artifactId>
  <version>3.1.32</version>
  <executions>
    <execution>
      <id>fmp</id>
      <goals>
        <goal>resource</goal>
        <goal>helm</goal>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

# Build Docker Image

```
mvn package fabric8:build
```

```
[INFO] --- fabric8-maven-plugin:3.1.32:build (fmp) @ client ---
```

```
[INFO] F8> Running in Kubernetes mode
```

```
[INFO] F8> Running generator spring-boot
```

```
[INFO] F8> Running generator java-exec
```

```
[INFO] F8> Pulling from fabric8/java-alpine-openjdk8-jdk
```

```
e110a4a17941: Pull complete
```

```
deb4805e2548: Pull complete
```

```
04712c369ba1: Pull complete
```

```
2c82593e8eb2: Downloading [=====> ] 49.54 MB/49.68 MB
```

```
58bb43a36a2e: Download complete
```

```
a2f01f9f1b00: Download complete
```

```
e6f2f9c9e249: Download complete
```

```
179ec0f1d75a: Download complete
```

```
61cbb3b63095: Download complete
```

```
5bd50883c949: Download complete
```

```
□
```



# Local Docker Repository

Hello Service

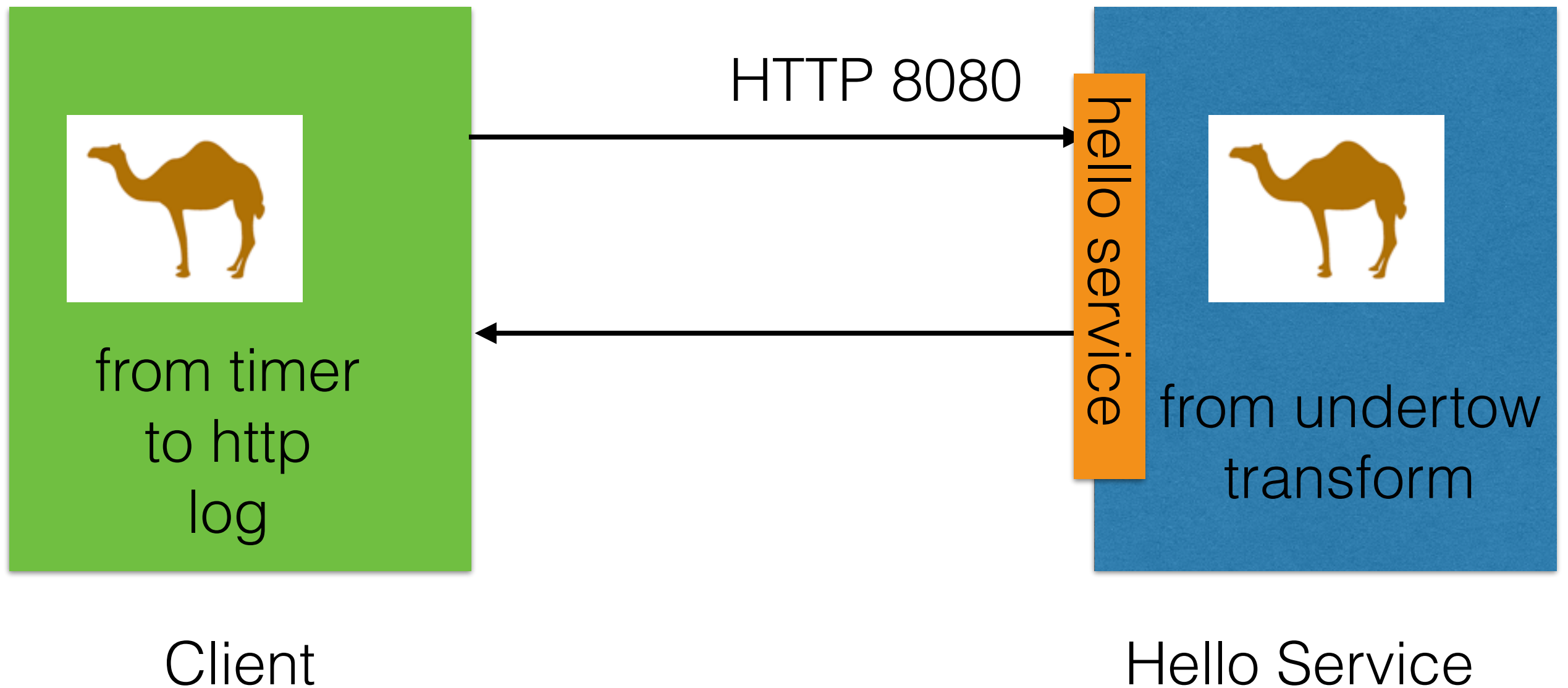
```
davsclaus:/Users/davsclaus/Documents/workspace/client/$ docker images
```

REPOSITORY	TAG	IMAGE ID
foo/helloswarm	latest	f572db11fc2e
foo/client	latest	c16cac32bf39
openshift/origin-deployer	v1.3.0-rc1	7e0bab2a9a21
openshift/origin-node	v1.3.0-rc1	ea67ec68a53a
fabric8/exposed	latest	7e2e98c75db5
fabric8/fabric8	2.2.174	efb7bc509cb0
fabric8/java-alpine-openjdk8-jdk	latest	c9139d27b712

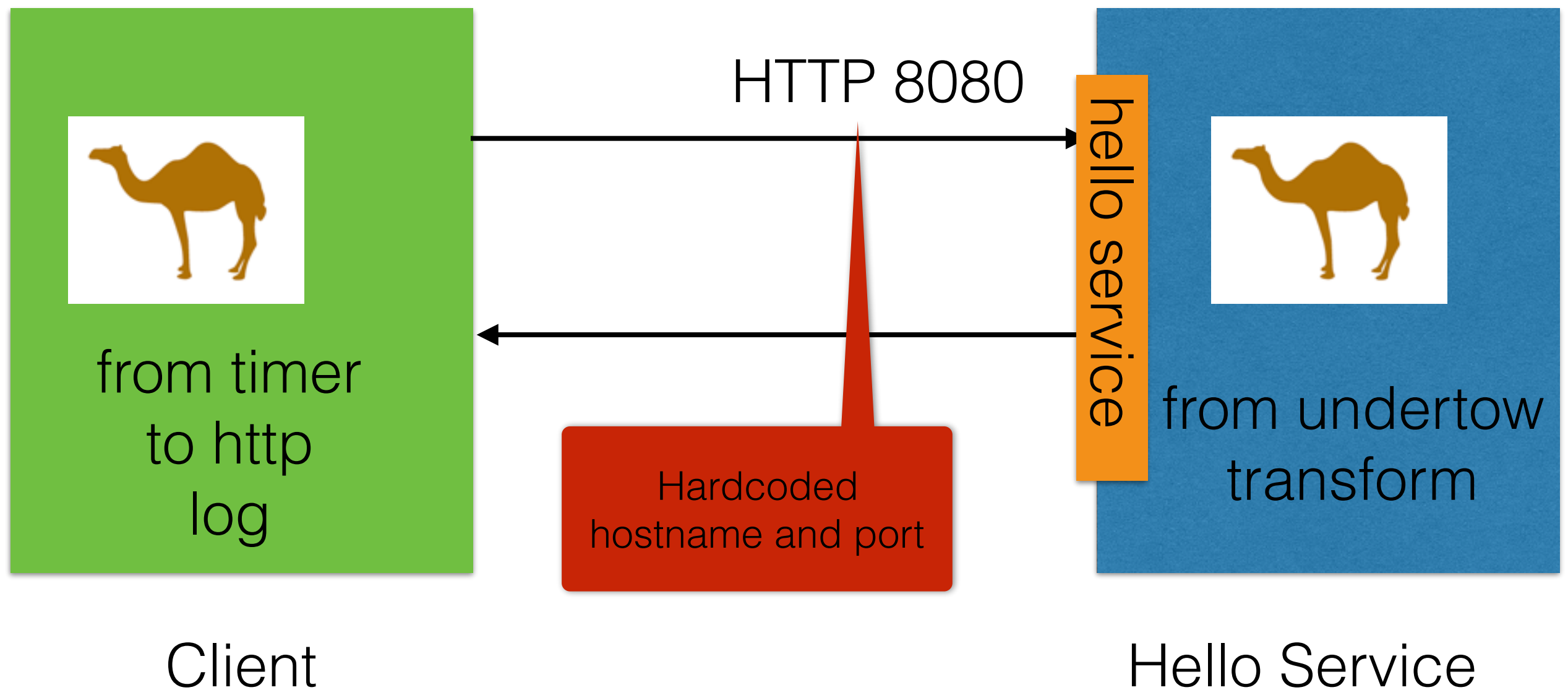
Client



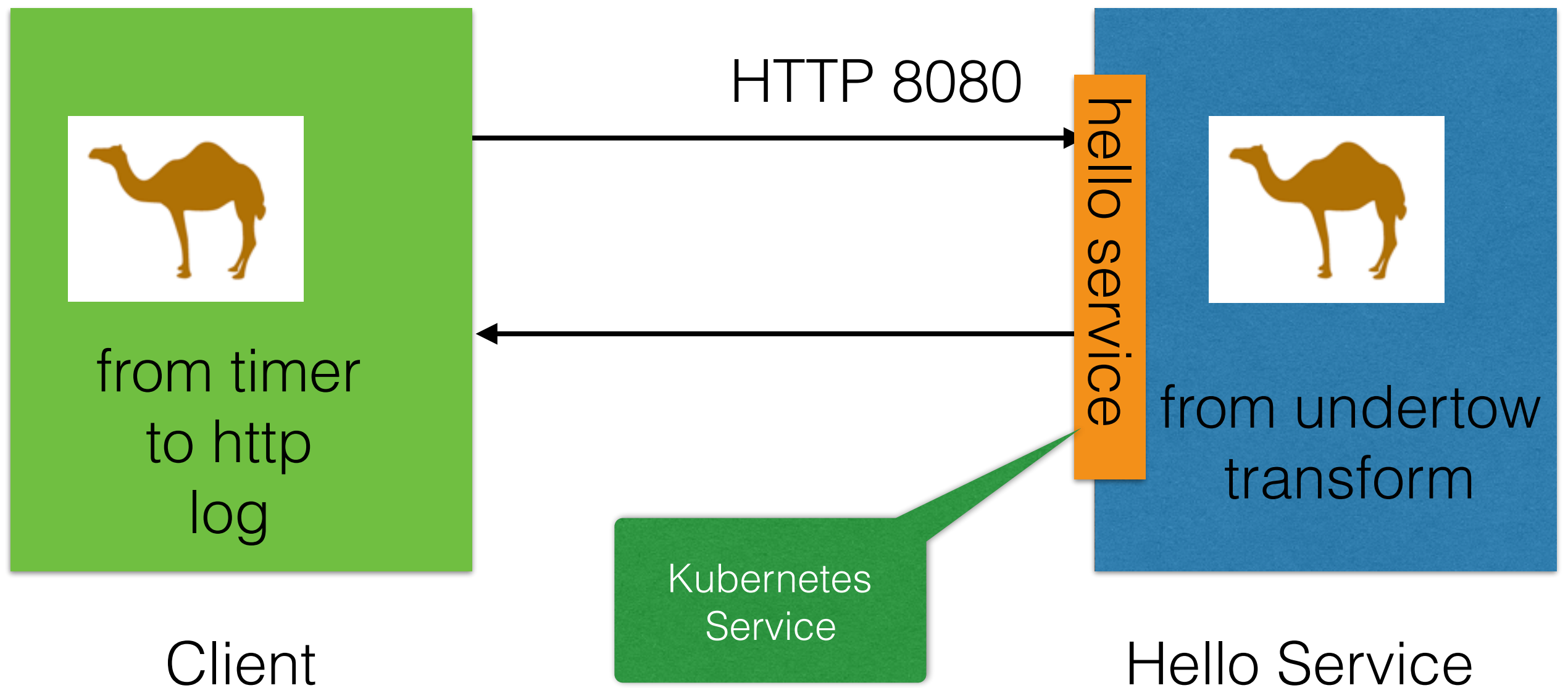
# Our Demo



# Static vs Dynamic Platform



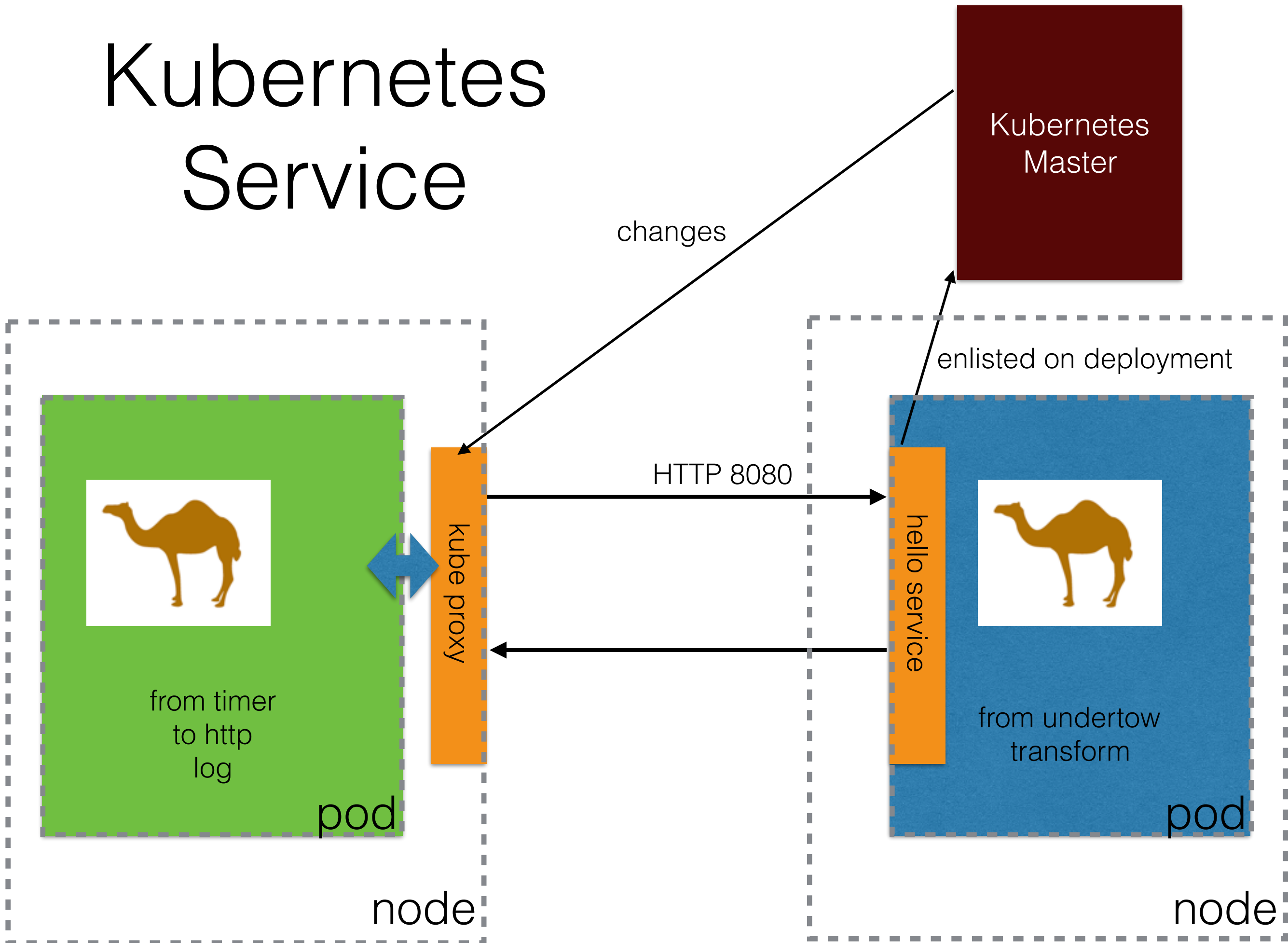
# Dynamic Platform



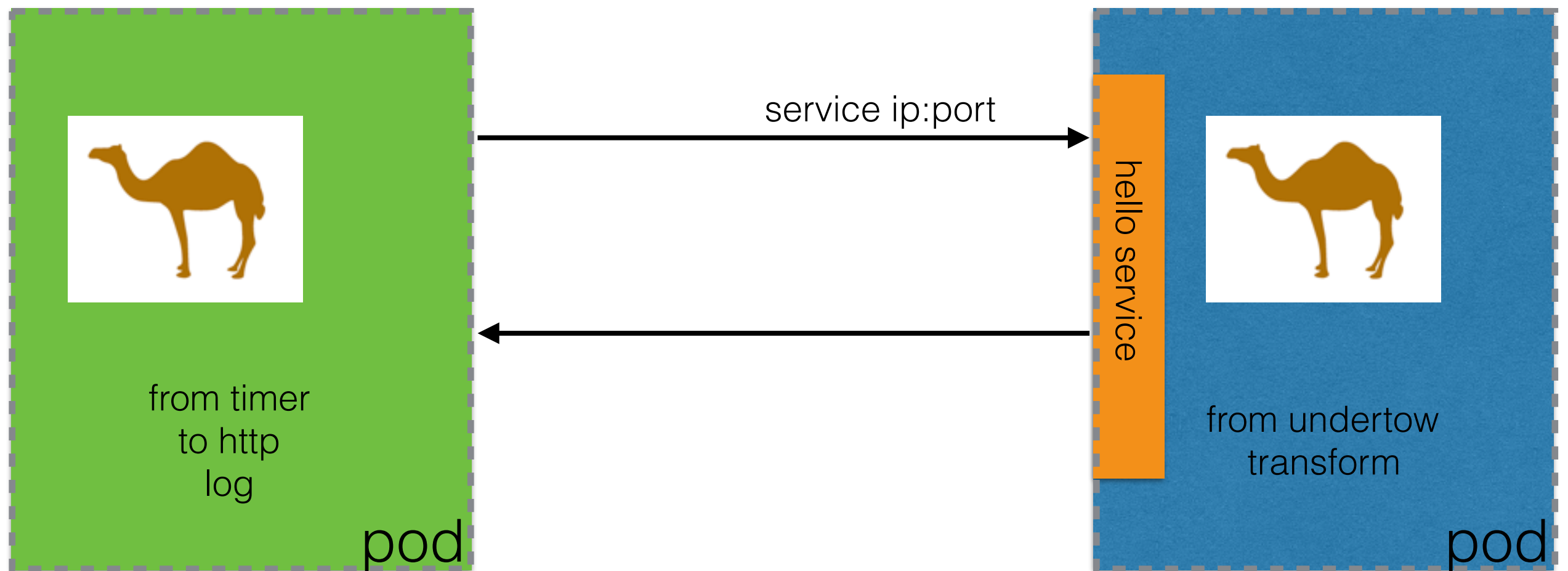
# Kubernetes Service

- Network Connection to one or more Pods
- Unique static IP and port (lifetime of service)

# Kubernetes Service



# Kubernetes Service from user point of view





# Out of the box service

```
| name: "helloworld"  
spec:  
  ports:  
  - port: 8080  
    protocol: "TCP"  
    targetPort: 8080
```

Maven artifactId

WildFly Swarm  
HTTP port

# Custom Configured Service service.yml

```
metadata:
  annotations:
    api.service.kubernetes.io/path: /hello
  name: "hello"
spec:
  ports:
    - port: 8181
      protocol: "TCP"
      targetPort: 8080
  type: LoadBalancer
```

Service Name

Service Port =  
Outside

Container Port =  
Inside



# Using Kubernetes Service



from timer  
to http  
log

Client

We want to use hello service

How do we do that?

# Using Kubernetes Service



from timer  
to http  
log

Client

- Environment Variables
  - Hostname
  - Port

```
declare -x HELLO_SERVICE_HOST="172.30.15.193"  
declare -x HELLO_SERVICE_PORT="8282"
```

Service Discovery using DNS is available  
in newer versions of Kubernetes.

# Service using ENV



from timer  
to http  
log

Client

- Use ENV

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("timer:foo?period=2000")
            .to("netty4-http:http://{{service:hello}}/hello")
            .log("${body}");
    }
}
```

# Service using DNS



from timer  
to http  
log

Client

- Use DNS

```
@Component
public class MyRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("timer:foo?period=2000")
            .to("netty4-http:http://hello:8181/hello")
            .log("${body}");
    }
}
```

# Service using DNS



from timer  
to http  
log

Client

```
@Component
public class MyRoute extends RouteBuilder {

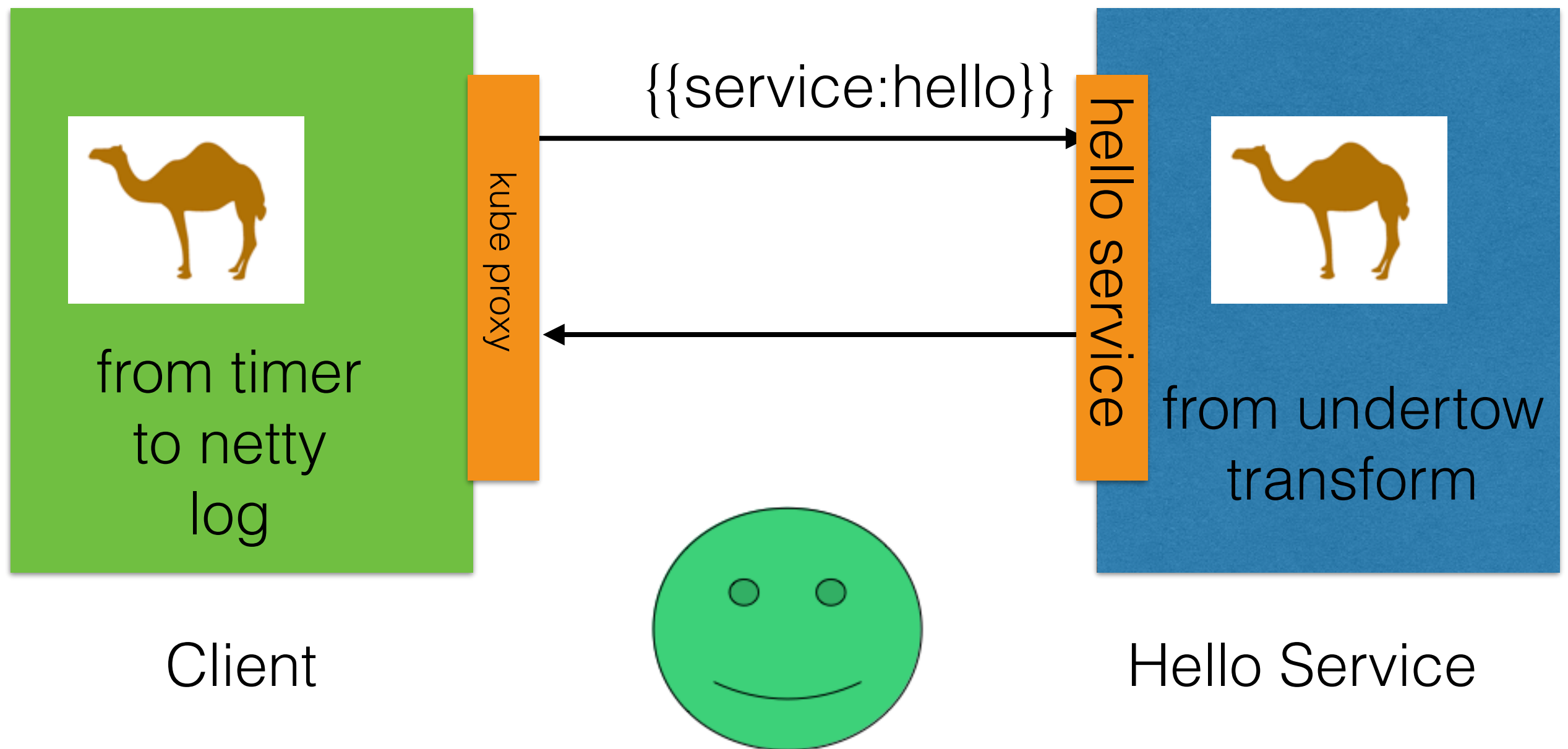
    @Override
    public void configure() throws Exception {
        from("timer:foo?period=2000")
            .to("netty4-http:http://hello:8181/hello")
            .log("${body}");
    }
}
```

Kubernetes  
Service

```
metadata:
  annotations:
    api.service.kubernetes.io/path: /hello
  name: hello
spec:
  ports:
    - port: 8181
      protocol: TCP
      targetPort: 8080
  type: LoadBalancer
```



# Ready to run in Kubernetes

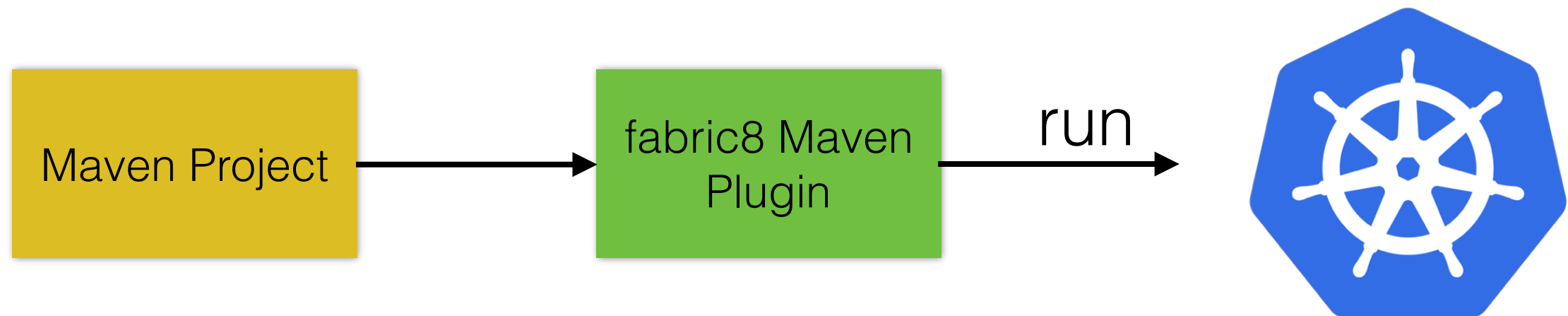


# How to deploy to Kubernetes?

Maven Project



# How to deploy to Kubernetes?





# Deploy - Hello Service

hello service



from undertow  
transform

Hello Service

- mvn fabric8:run

```
davsclaus:/Users/davsclaus/Documents/workspace/helloswarm/$ mvn fabric8:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building Wildfly Swarm Example 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] >>> fabric8-maven-plugin:3.1.32:run (default-cli) > install @ helloswa
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ hellosw
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- fabric8-maven-plugin:3.1.32:resource (fmp) @ helloswarm ---
[INFO] F8> Running in Kubernetes mode
```

# Deploy - Client



from timer  
to http  
log

Client

- `mvn fabric8:run`

```
davsclaus:/Users/davsclaus/Documents/workspace/client/$ mvn fabric8:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building client 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] >>> fabric8-maven-plugin:3.1.32:run (default-cli) > install @ client >>>
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ client ---
```

# Running Kubernetes Locally

- MiniKube
- MiniShift
- Vagrant
- OpenShift CDK

<https://fabric8.io/guide/getStarted/index.html>

# Running MiniShift

- Install MiniShift

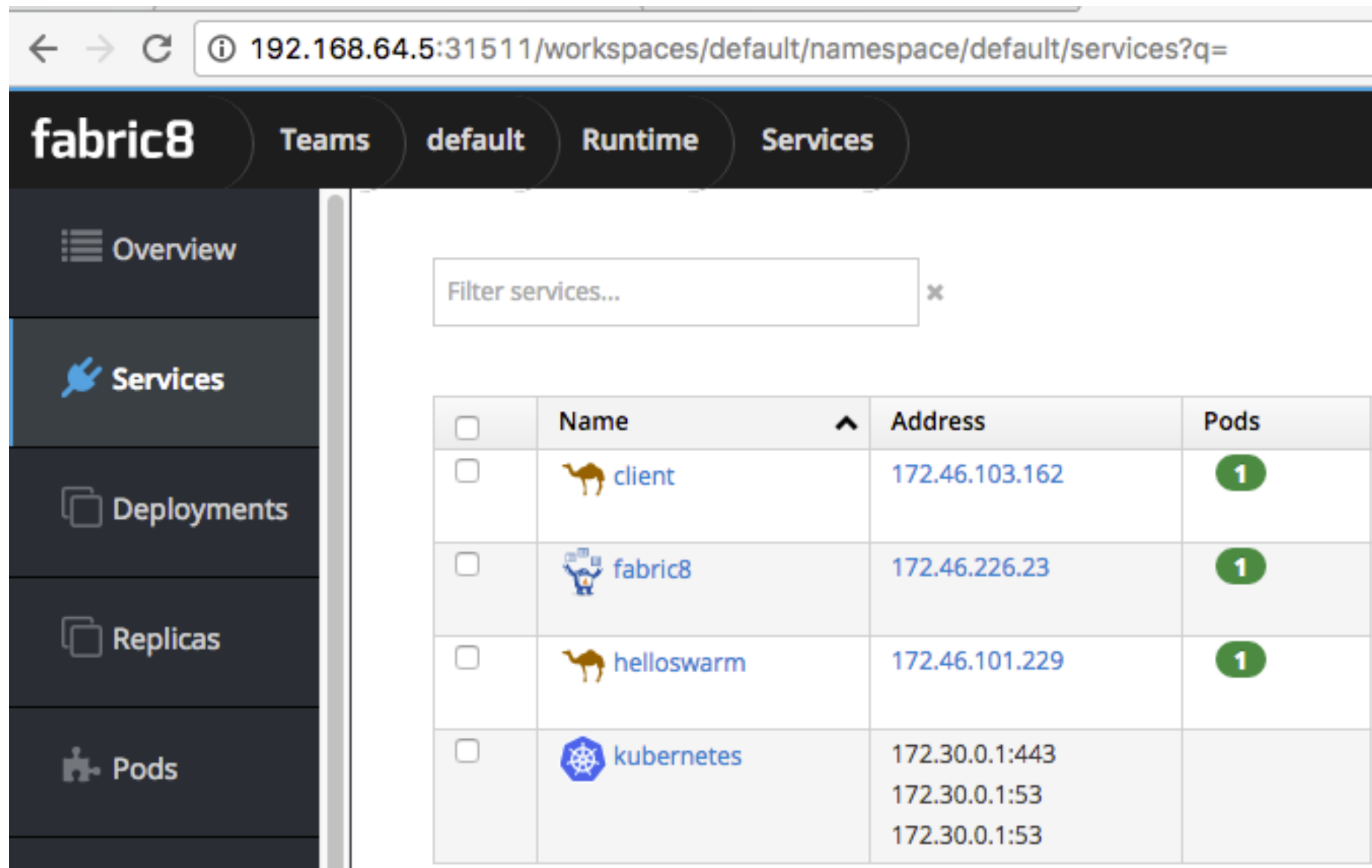
```
minishift start --memory=2000
```

- Install fabric8





```
gofabric8 deploy -y  
--domain=$(minishift ip).xip.io  
--api-server=$(minishift ip) --app=
```

<https://fabric8.io/guide/getStarted/minishift.html>

# fabric8 Web Console



The screenshot shows the fabric8 Web Console interface. The browser address bar displays the URL: 192.168.64.5:31511/workspaces/default/namespace/default/services?q=. The navigation bar includes tabs for Teams, default, Runtime, and Services. The left sidebar contains links for Overview, Services (selected), Deployments, Replicas, and Pods. The main content area features a 'Filter services...' input field and a table of services.

<input type="checkbox"/>	Name	Address	Pods
<input type="checkbox"/>	 client	172.46.103.162	1
<input type="checkbox"/>	 fabric8	172.46.226.23	1
<input type="checkbox"/>	 helloswarm	172.46.101.229	1
<input type="checkbox"/>	 kubernetes	172.30.0.1:443 172.30.0.1:53 172.30.0.1:53	

minishift service fabric8

# OpenShift CLI

You can also use CLI from  
docker &  
kubernetes

- `oc get pods`

```
davsclaus:/Users/davsclaus/Documents/workspace/client/$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
client-1-60c0o	1/1	Running	5	9m
exposecontroller-1-2fb1p	1/1	Running	0	35m
fabric8-cg8c0	1/1	Running	0	36m
helloswarm-1-qcnv7	1/1	Running	0	14m



# OpenShift CLI

- `oc get service`

```
davsclaus:/Users/davsclaus/Documents/workspace/$ oc get service
```

NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)
client	172.30.14.254	172.46.103.162,172.46.103.162	8080/TCP
fabric8	172.30.140.79	172.46.226.23,172.46.226.23	80/TCP
hello	172.30.21.39	172.46.203.233,172.46.203.233	8181/TCP
kubernetes	172.30.0.1	<none>	443/TCP,5

# OpenShift CLI

- `oc logs -f <pod name>`

```
davsclaus:/Users/davsclaus/Documents/workspace/$ oc logs -f client-1-qlfym
I> No access restrictor found, access to any MBean is allowed
Jolokia: Agent started with URL http://172.17.0.3:8778/jolokia/
2016-09-09 13:07:33.604:INFO:ifasjipjsoejs.Server:jetty-8.y.z-SNAPSHOT
2016-09-09 13:07:33.649:INFO:ifasjipjsoejs.AbstractConnector:Started Select
```

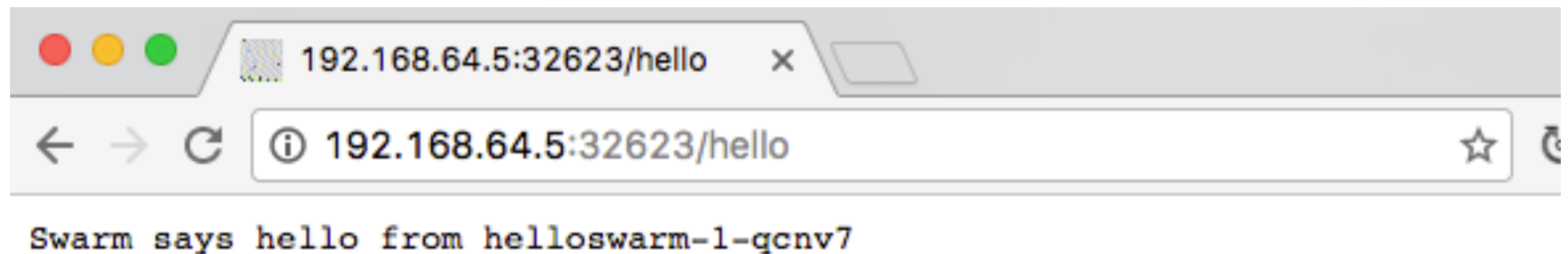
```
  .   _--_
 ^\ /  _-' _--_ _-( )_ _--_ _--_ \ \ \ \
( ( ) \__| ' _| ' _| ' _| ' _| \ \ \ \
 \ \  _ _ ) | | _ ) | | | | | | | | ( _ | | ) ) )
  '  | _ _ | . _ | | | | | | | | \ \ , | / / / /
=====|_|=====|_|_/=/ / / /
:: Spring Boot ::      (v1.4.0.RELEASE)
```

```
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-qcnv7
```






# Access Service from your laptop




- minishift service hello



# Scaling

- Change deployment replicas

<input type="checkbox"/>	Name ^	Pods	Scale
<input type="checkbox"/>	 client	1	1
<input type="checkbox"/>	 exposecontroller	1	1
<input type="checkbox"/>	 helloswarm	1	1

<input type="checkbox"/>	Name ^	Pods	Scale
<input type="checkbox"/>	 client	1	1
<input type="checkbox"/>	 exposecontroller	1	1
<input type="checkbox"/>	 helloswarm	2	2



# Scaling

Load balancing  
is random

- Service load balancing

```
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-665c1
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-qcnv7
Swarm says hello from helloswarm-1-665c1
```

# Angry Pods



1st person shooter - Kill your pods

# Links



@davsclaus  
davsclaus  
[davsclaus.com](http://davsclaus.com)

- fabric8
  - <http://fabric8.io>
- Demo source code
  - <https://github.com/davsclaus/fabric8-hello>
- Try kubernetes / fabric8
  - <https://fabric8.io/guide/getStarted/minishift.html>
- Videos, blogs and more
  - <https://fabric8.io/community/index.html>