



# Apache Camel

in the belly of the



# docker

# Brought to you by...



A screenshot of a Twitter profile for Henryk Konsek. The profile picture shows a man with curly hair and a beard. The bio identifies him as an engineer at Red Hat (JBoss) and an open source hacker. The location is listed as Poland, and a link to his about.me profile is provided. The tweet count is 2,374. A partial tweet is visible on the right side of the screenshot.

**Henryk Konsek**  
@hekonsek

Engineer at Red Hat (JBoss). Open source hacker (JBoss Fabric8, Apache Camel, JBoss Hawt.io). Evangelist of the open source software that rocks.

Poland

[about.me/hekonsek](https://about.me/hekonsek)

TWEETS  
2,374

Tweet

Her  
Sta  
bel  
ap

Her

# What is...



Server for running and managing Linux containers.

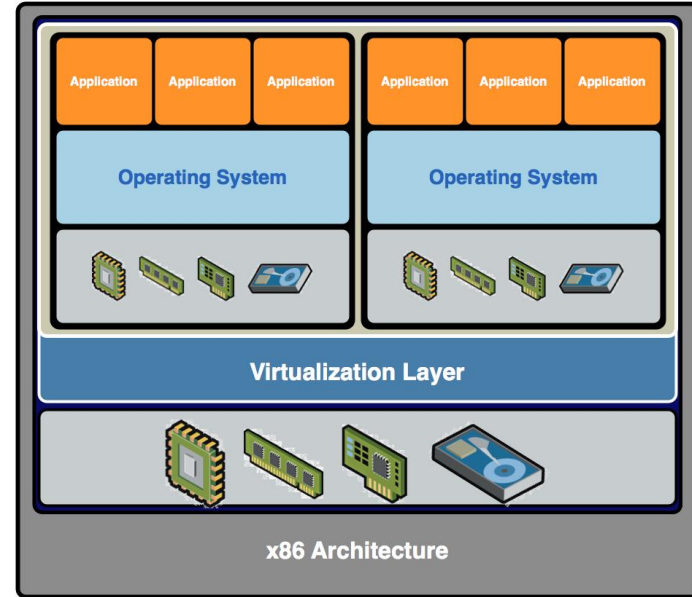
# What are Linux containers?



Para-virtualized Linux instances.

# Why not regular virtualization?

- slooooooow
- gigantic images
- aggressive resource allocation
- bad API



# Key concepts

- image (immutable, no state)
- container (has state)

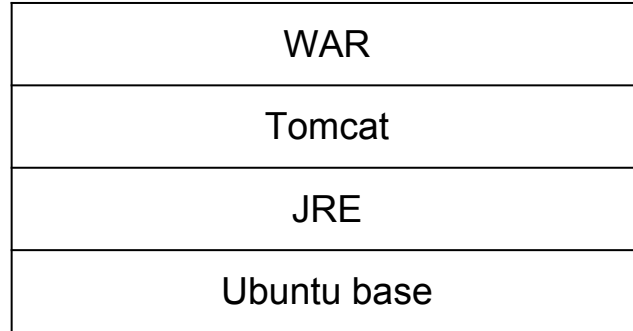
***Container*** is the running ***image***.

# Docker awesomeness #1

Commands:	
attach	Attach to a running container
build	Build an image from a Dockerfile
commit	Create a new image from a container's changes
cp	Copy files/folders from a container's filesystem to the host path
diff	Inspect changes on a container's filesystem
events	Get real time events from the server
export	Stream the contents of a container as a tar archive
history	Show the history of an image
images	List images
import	Create a new filesystem image from the contents of a tarball
info	Display system-wide information
inspect	Return low-level information on a container
kill	Kill a running container
load	Load an image from a tar archive
login	Register or log in to a Docker registry server
logout	Log out from a Docker registry server
logs	Fetch the logs of a container
port	Lookup the public-facing port that is NAT-ed to PRIVATE_PORT
pause	Pause all processes within a container
ps	List containers
pull	Pull an image or a repository from a Docker registry server
push	Push an image or a repository to a Docker registry server
restart	Restart a running container
rm	Remove one or more containers
rmi	Remove one or more images
run	Run a command in a new container
save	Save an image to a tar archive
search	Search for an image on the Docker Hub
start	Start a stopped container
stop	Stop a running container
tag	Tag an image into a repository
top	Lookup the running processes of a container
unpause	Unpause a paused container
version	Show the Docker version information
wait	Block until a container stops, then print its exit code

Commands.

# Docker awesomeness #2



Layers.

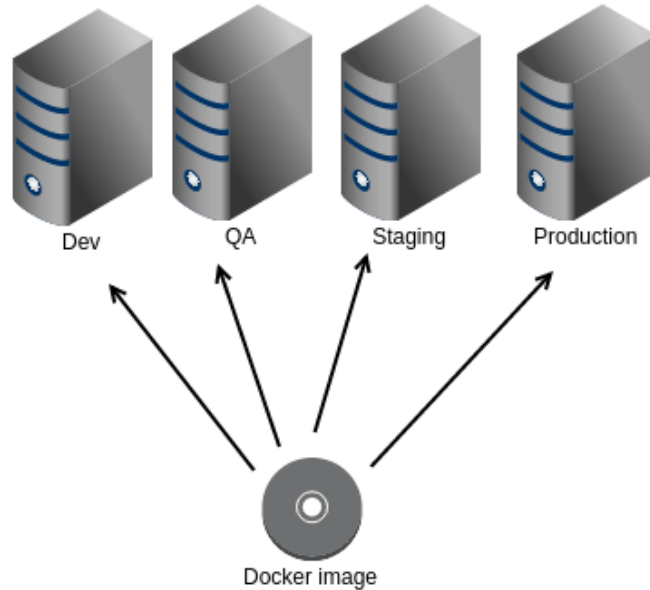


# Docker awesomeness #3



Registries.

# Docker awesomeness #4



Build once. Deploy everywhere!

# Dockerfiles

```
FROM ubuntu
EXPOSE 8080
RUN apt-get install java
RUN mkdir /jars
ADD target/app.jar /jars/
CMD ["java", "-jar", "/jars/app.jar"]
```

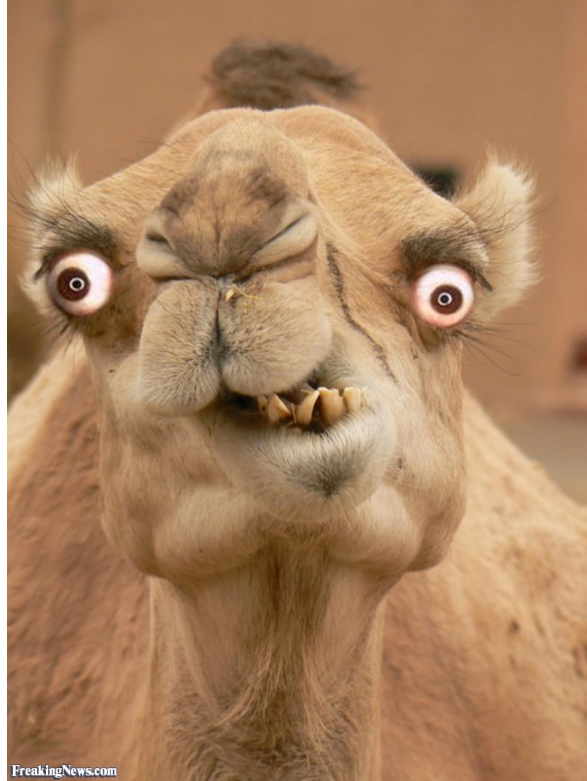
“Recipes” for the new images.

# Create new image in the local repo

```
$ docker build -t com.me/app:1.0
```

```
$ docker run -t com.me/app:1.0
```

# Apache Camel



FreakingNews.com

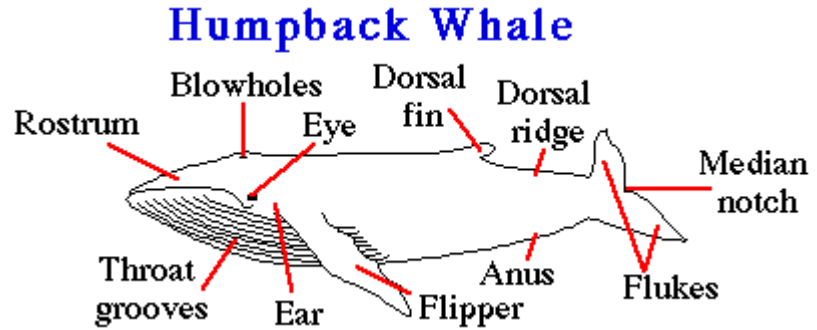
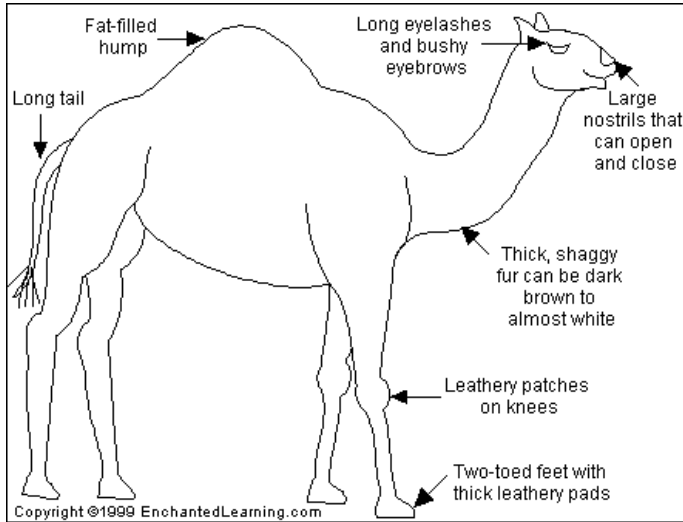
Framework that routes messages like crazy.

# How to deploy Camel?

- Karaf (JBoss Fuse, ServiceMix, Talend ESB)
- Tomcat
- WildFly (JBoss EAP)
- standalone/embedded
- Akka plugin
- Grails plugin
- **Spring Boot**

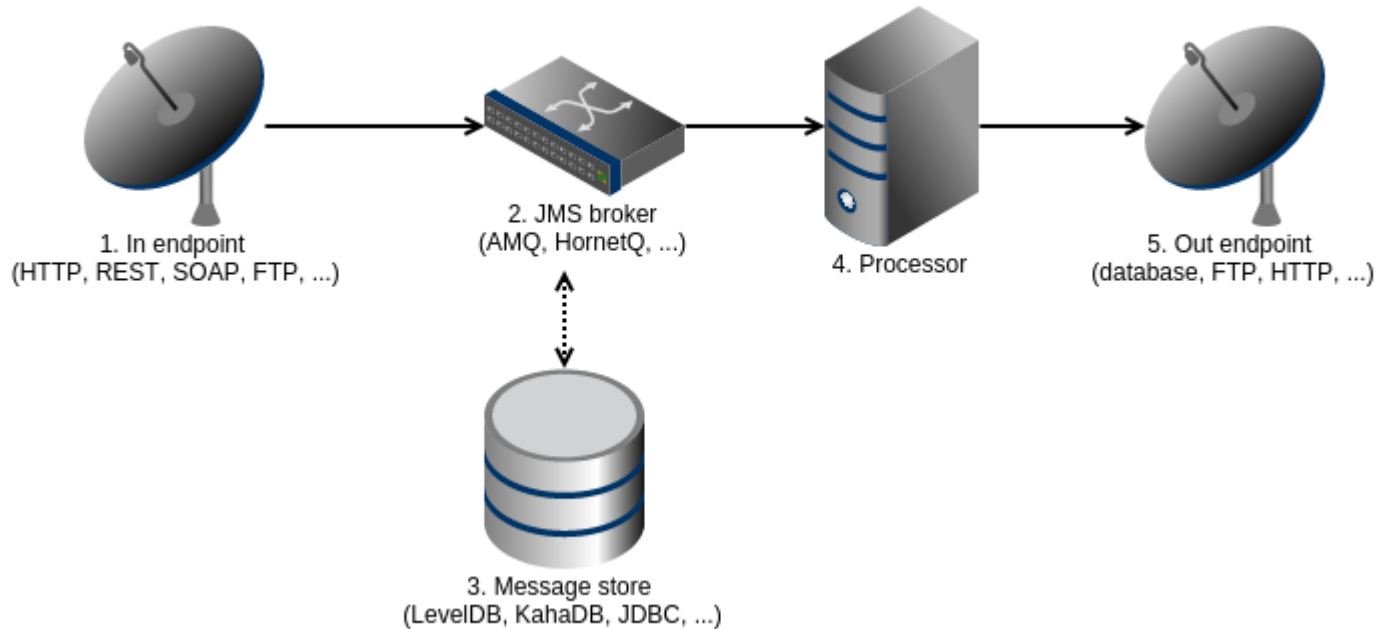


# Camel and Docker



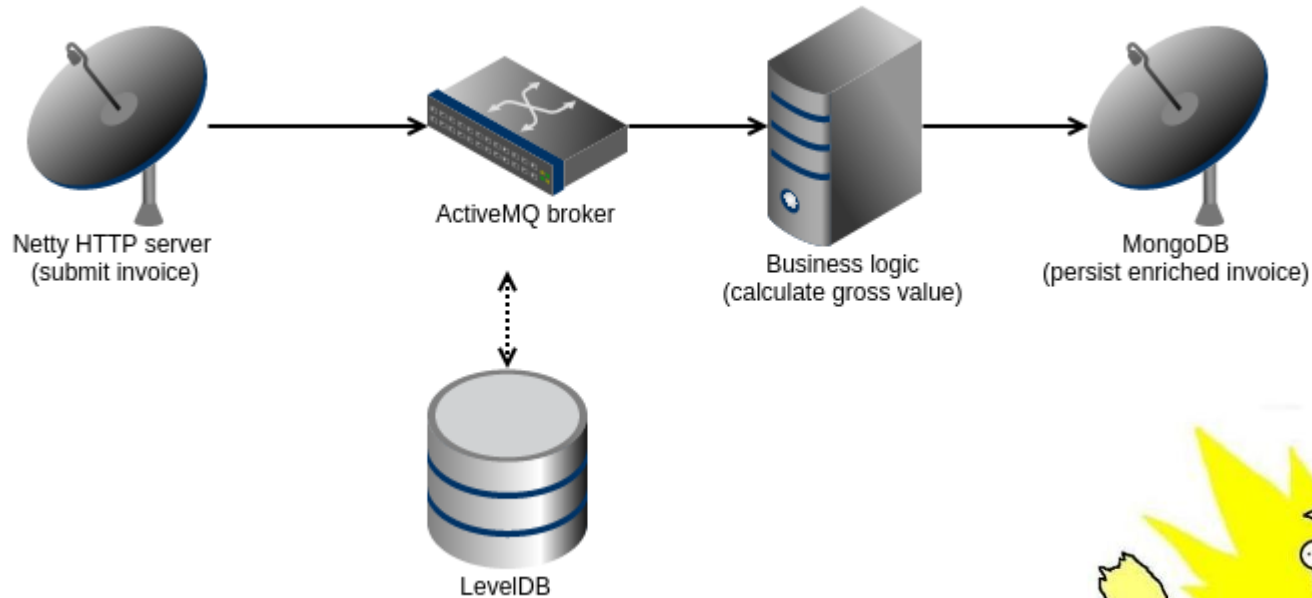
How to split it? What should I dockerize?

# Messaging architecture in a nutshell





# Concrete messaging architecture

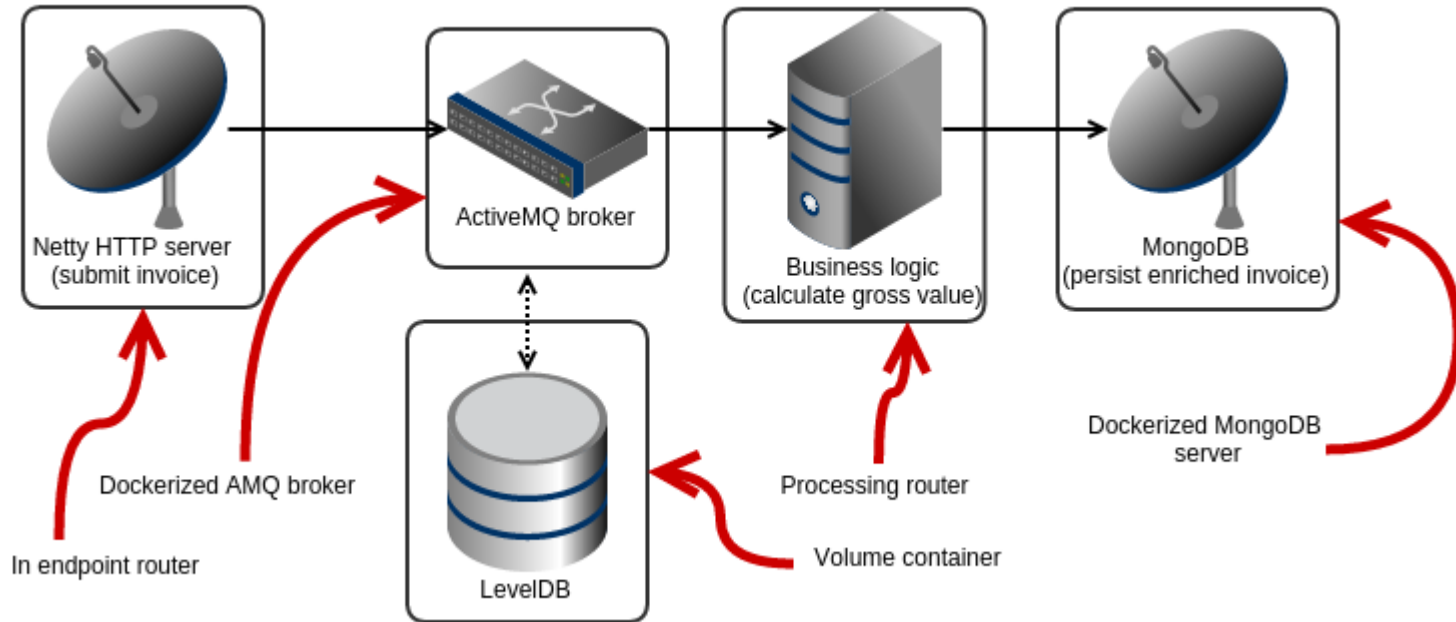


**I DON'T DOCKERIZE  
OFTEN**



**BUT WHEN I DO, I  
DOCKERIZE EVERYTHING**

# Dockerized example



# In-endpoint route

```
from("netty-http:http://0.0.0.0:18080") .  
    setBody(randomUUID()) .  
    inOnly("jms:invoices");
```

...

```
new ActiveMQConnectionFactory("tcp://amqbroker:6162")
```

# Processing route

```
from("jms:invoices").
    setBody().
    groovy("new Invoice(request.body,currentTimeMillis())").
    to("mongodb:mongo?...operation=insert");

...

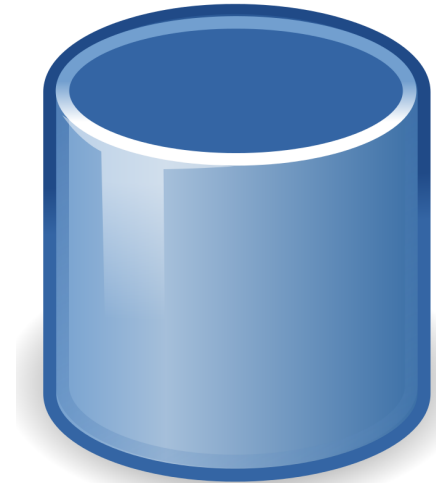
new ActiveMQConnectionFactory("tcp://amqbroker:6162");

...

new MongoClient("mongodb");
```

# How can I get database images?

```
docker run -d -p 27017:27017  
--name mongodb dockerfile/mongodb
```



Provided by database  
community/vendor.

# How can I put fresh jar into image?



The screenshot shows the GitHub repository page for 'rhuss / docker-maven-plugin'. The repository description is 'Maven plugin for managing Docker images and containers'. It has 184 commits, 3 branches, 15 releases, and 4 contributors. The current branch is 'master'. The repository contains a commit titled 'Fixed compile plugin' by 'rhuss' from an hour ago, with the latest commit hash '3c40502a13'. There are two folders listed: 'samples' (Fixed compile plugin, an hour ago) and 'src' (Merge branch 'bind-ip' of github.com:jgangemi/docker-maven-plugin int..., 3 days ago). The right sidebar shows navigation options: Code, Issues, Pull Requests, Wiki, Pulse, and Graphs.

rhuss / **docker-maven-plugin** Watch 5 Star 38

Maven plugin for managing Docker images and containers

184 commits   3 branches   15 releases   4 contributors

branch: master **docker-maven-plugin / +**

Fixed compile plugin

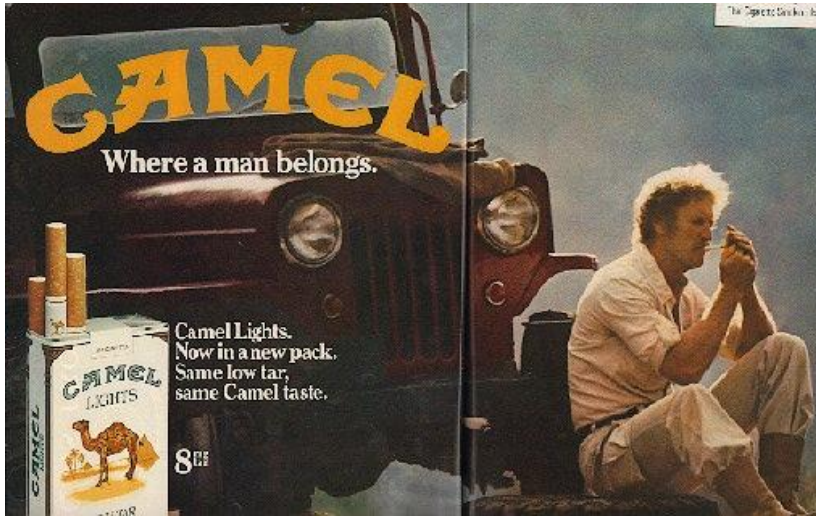
rhuss authored an hour ago   latest commit 3c40502a13

samples	Fixed compile plugin	an hour ago
src	Merge branch 'bind-ip' of github.com:jgangemi/docker-maven-plugin int...	3 days ago

Code  
Issues  
Pull Requests  
Wiki  
Pulse  
Graphs

Docker Maven plugin by Roland 'Jolokia' Huß

# How to bootstrap Camel?



- no Karaf bundle activators
- no server (Tomcat, etc.)
- how can we start CamelContext?



# Custom class with the main method



# Spring Boot for Camel



<http://projects.spring.io/spring-boot>

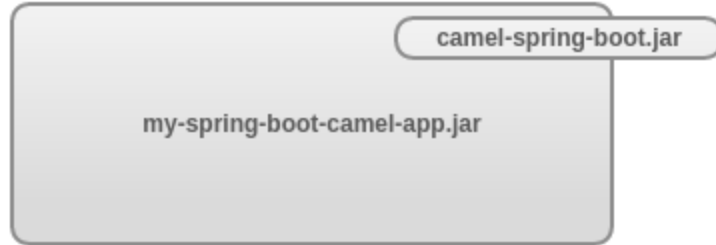
# Camel + Spring Boot: step #1



my-spring-boot-camel-app.jar

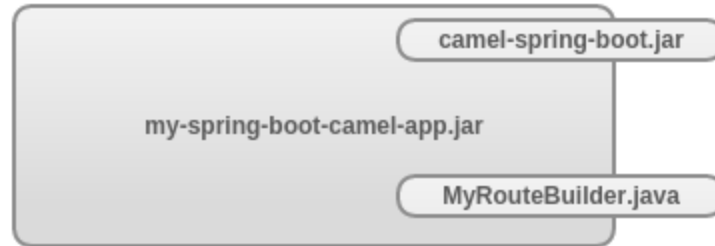
Take a Spring Boot fat jar.

# Camel + Spring Boot: step #2



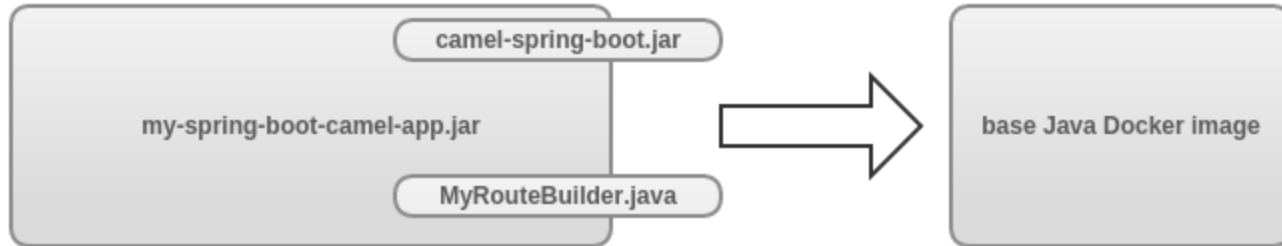
Add camel-spring-boot jar to your classpath.

# Camel + Spring Boot: step #3



Add Camel route to your classpath.

# Camel + Spring Boot: step #4



Dockerize your fat jar and run it!

# ENV-centric runtime configuration

```
# override endpoint definition via ENV variable  
docker run -e FROM=jms:queue -it my-springboot-camel-app
```

```
# run with the given Spring profile  
docker run -e spring.profiles.active=production -it my-  
springboot-camel-app
```

# Monitoring



Expose JMX via REST with the Jolokia base image.



# Kubernetes



- orchestration of many Docker containers
- ...and many Docker servers!
- logical container groups (pods)
- auto-scaling
- wiring your Docker stuff together



**Thank you!**