Gilles Philippart

Software Developer,
Société Générale

@gphilippart

Lee Namba

Solution Architect,
Docker

@leenamba

Agenda

Configuration Management with Containers

1. What goes in an image?
2. Managing configurations
3. How to put it in version control

Docker Reference Architecture:
Development Pipeline Best Practices Using Docker EE

www.docker.com/RA_DevPipeline

Docker is so easy!

…ah youth

First Dockerfile

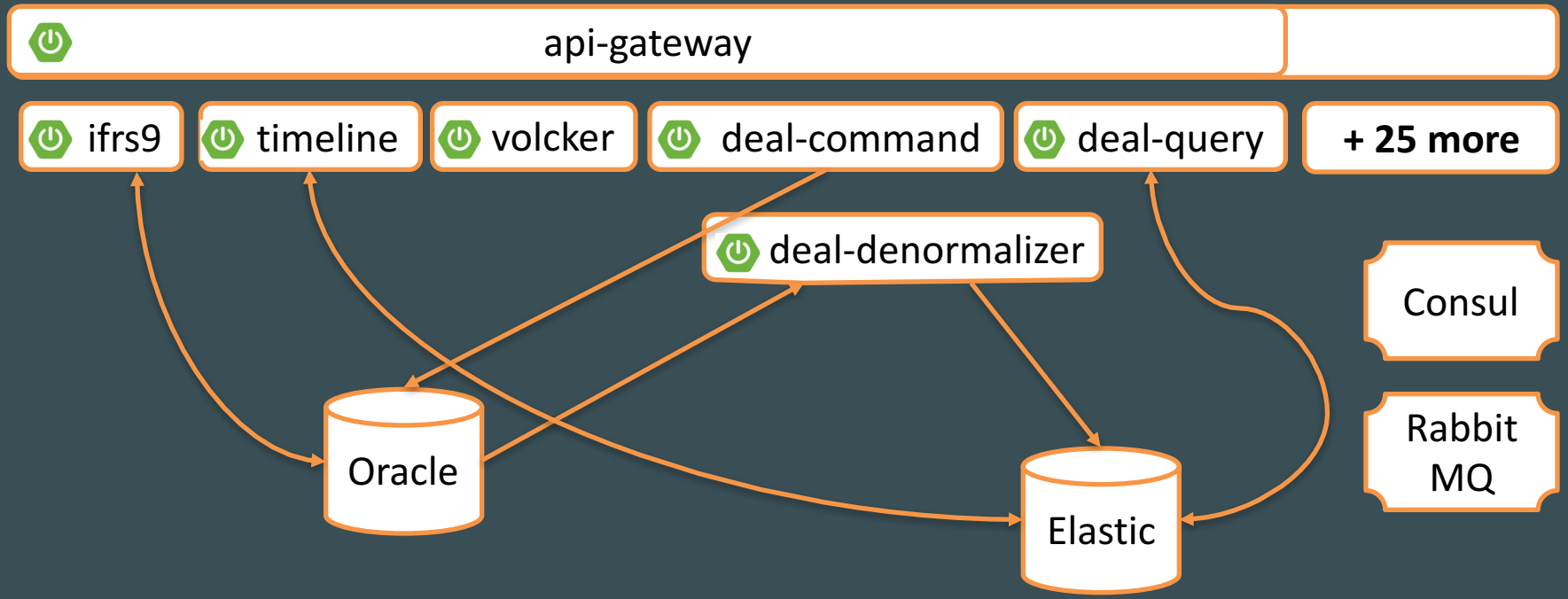
```
FROM dtr-poc.fr.world.socgen/gts_ssb/java8

WORKDIR /OGN

COPY *referential-*.jar ./

ENTRYPOINT ["./start-referential-query-gateway-rest.sh"]

EXPOSE 8083
```

Financing Platform

SOCIETE GENERALE

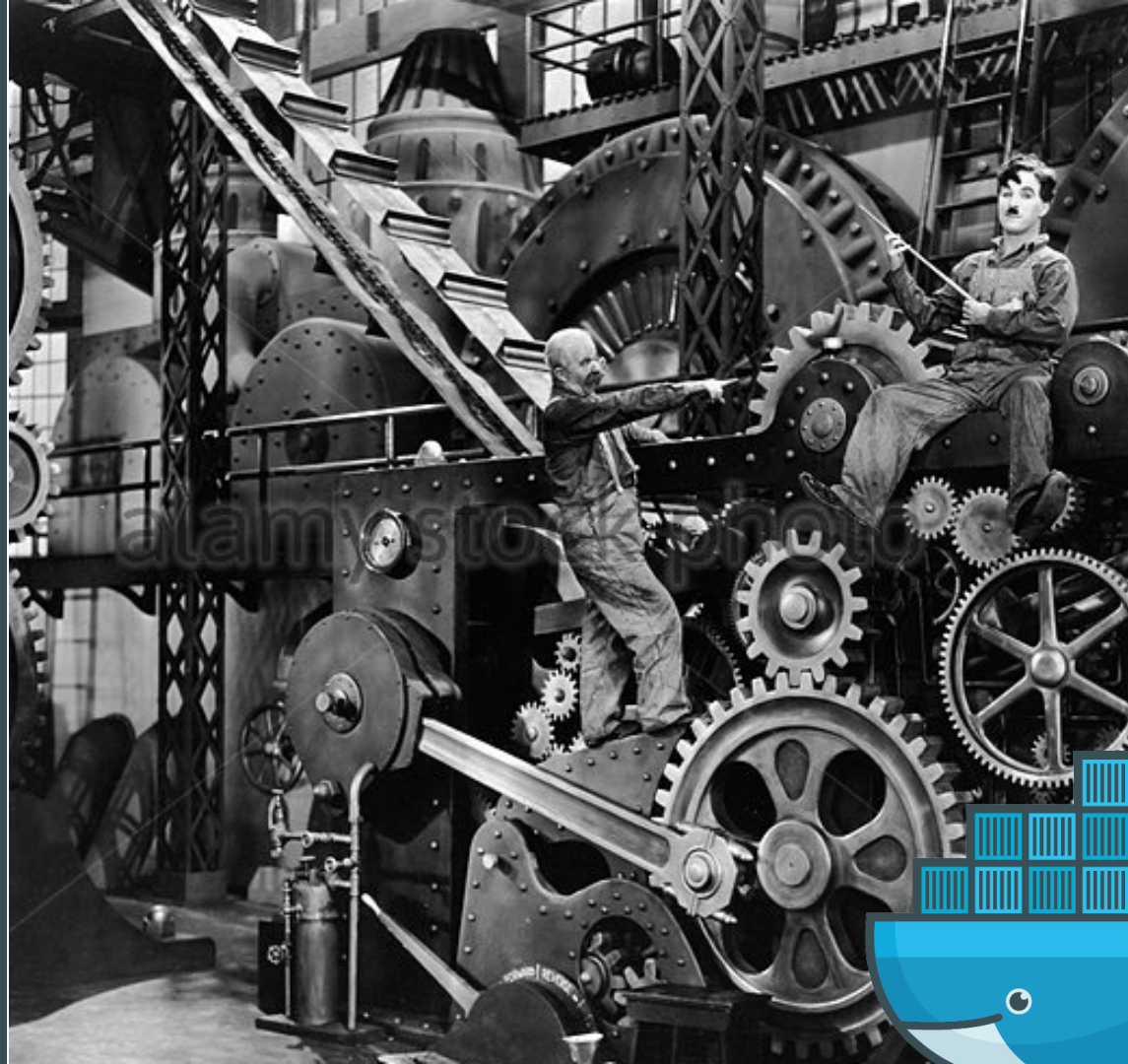
- Global Finance: Solutions for capital raising, structured financing, hedging
- +100 sectors
- Complex business processes

Micro-service architecture



Goals

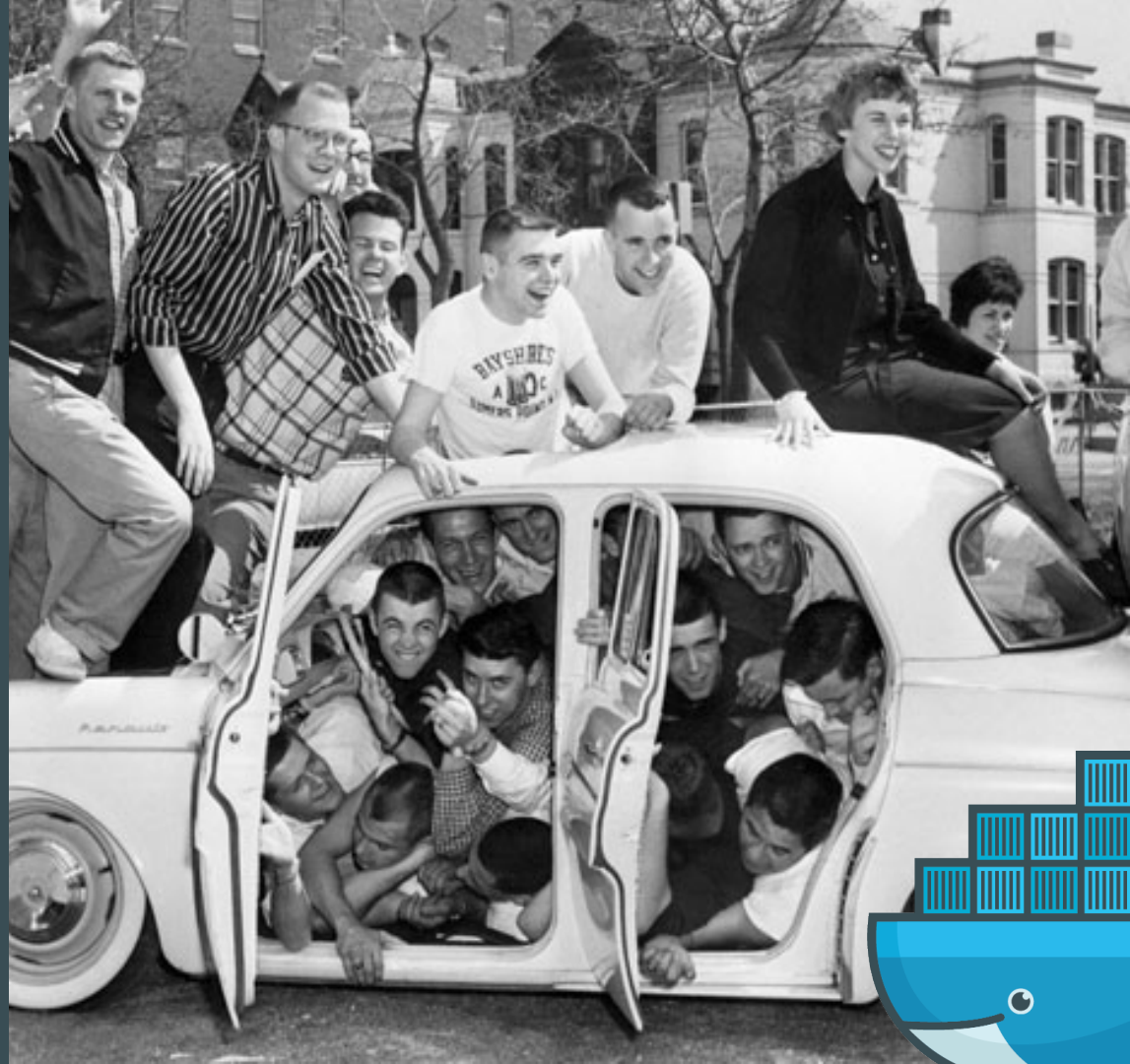
- Lift and shift
  - No change in code
  - No change in configuration
- Canary Deployment, Parallel run
- Challenges
  - 5 environments



Images

What goes in?

What stays out?



It depends…

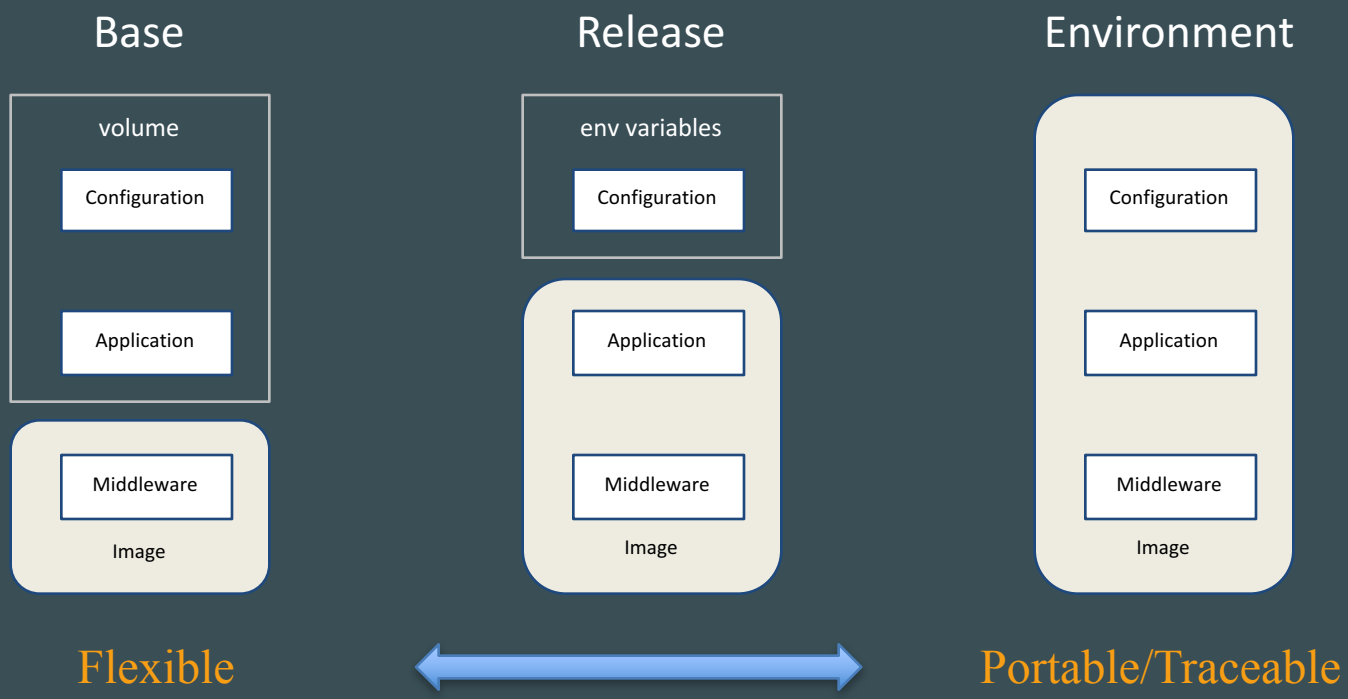
Build App

- Tool which runs once
- Single instance
- Multiple components per image

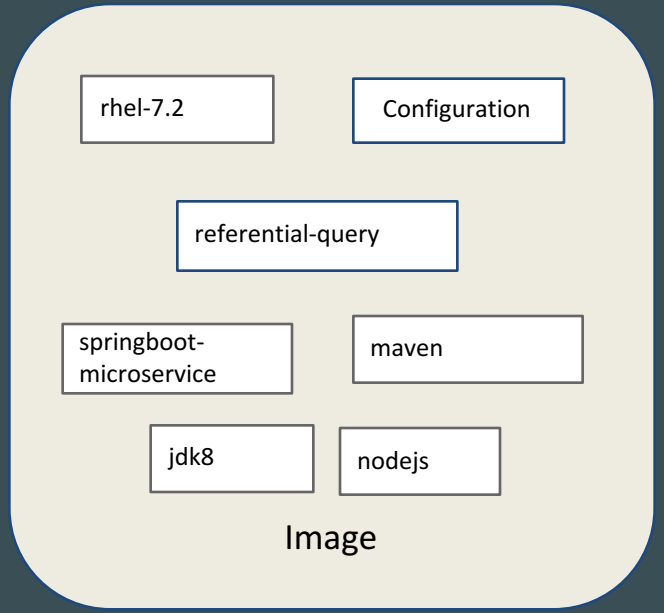
Run App

- Long-lived service
- Scaled
- One component per image

Run Image Types

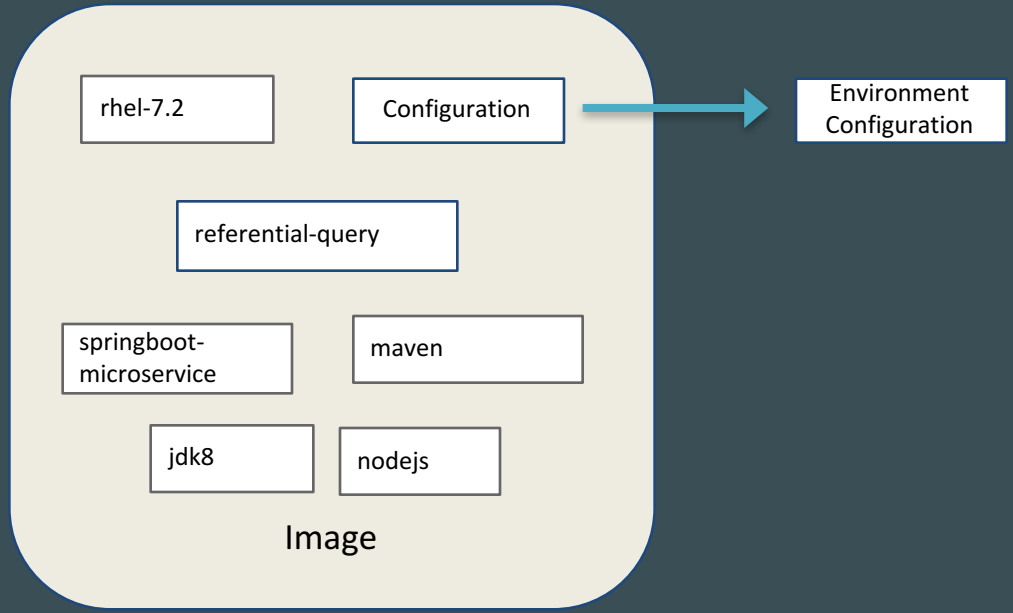


First Try: Image per Environment



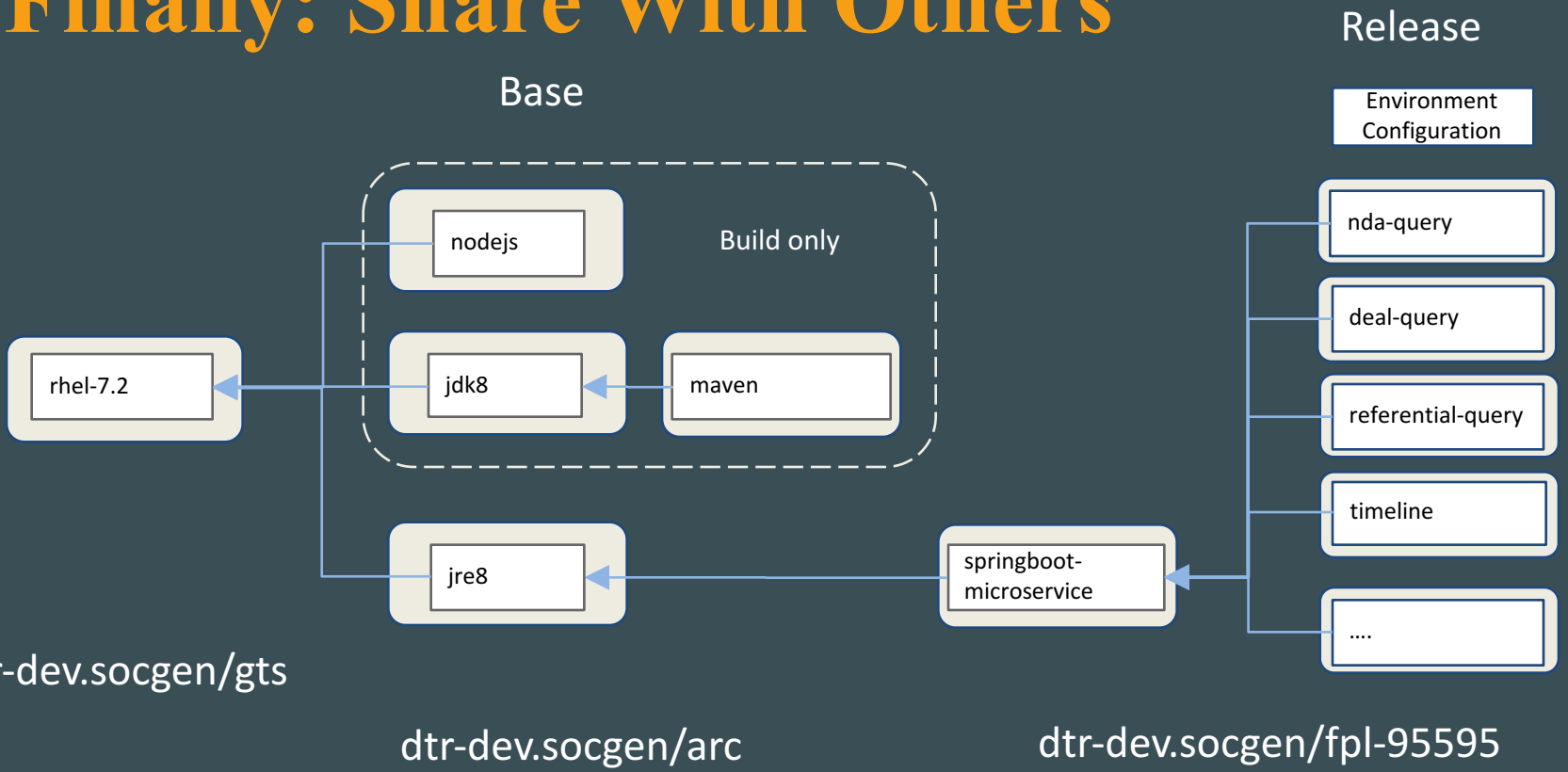
dtr-dev.socgen/fpl-95595

Second Try: Image per Release



dtr-dev.socgen/fpl-55595

Finally: Share With Others



Configuration

Where should configuration go?

Let’s set a variable…hmm?

Dockerfile
```
ENV FOO=bar
```

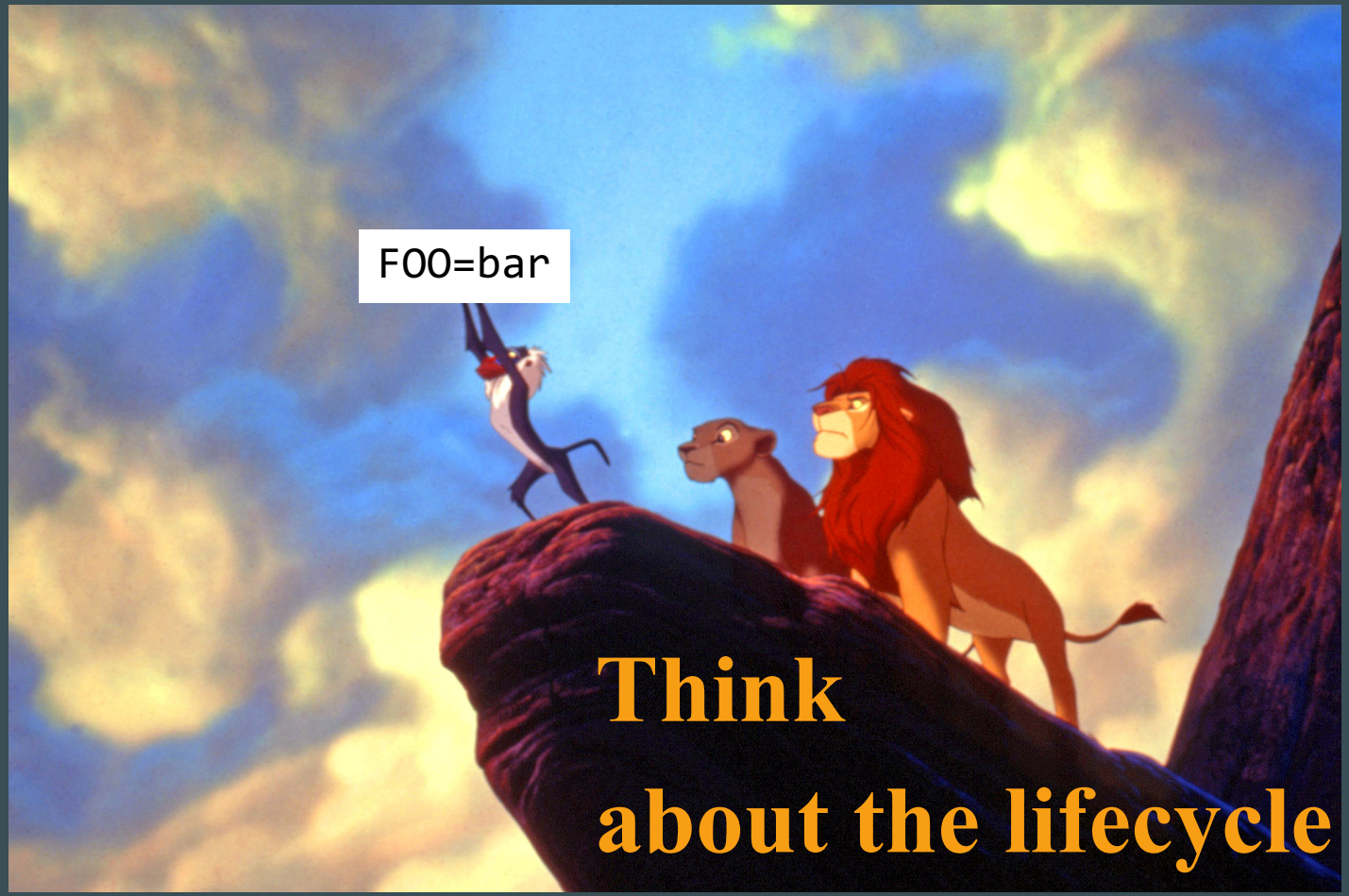
```
$ docker service create -e FOO=bar
```

docker-compose.yml
```
environment:
  FOO: bar
```

docker-compose.yml
```
env_file:
  - ./foo.env
```

entrypoint.sh
```
#!/bin/bash
FOO=$(curl https://con.acme.com/v1/kv/foo | jq -r '.[0].Value’)
```

```
$ echo ’bar’ | docker secret create FOO -
```


FOO=bar
Think
about the lifecycle

Configuration Lifecycle Buckets

When	What	Where
Yearly Build time	Enterprise policies and tools	Enterprise Base Image Dockerfiles
Monthly Build time	Application policies and tools	Application Base Image Dockerfiles
Monthly/Weekly Build time	Application Release	Release Image Dockerfiles
Weekly/Daily Deploy time	Static Environment Configuration	Environment variables docker-compose, .env
Deploy time	Dynamic Environment Configuration	Secrets entrypoint.sh, CLI, volumes
Run time	Elastic Environment Configuration	Service discovery, profiling, debugging, volumes

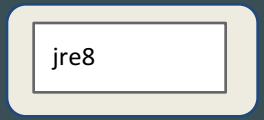
Base OS Image: Yearly

rhel-7.2

dtr-dev.socgen/gts

```
FROM registry.access.redhat.com/rhel7

MAINTAINER john.doe@sgcib.com
LABEL Description="SGCIB ready to go UGM images RHEL7"
LABEL Vendor="RESG/GTS/MKT/SSB"
LABEL versions="20161220"

COPY yum.conf/*.repo /etc/yum.repos.d/
COPY cacert/* /etc/pki/ca-trust/source/anchors
RUN update-ca-trust enable && update-ca-trust extract
```

Base Java Image: Semester



dtr-dev.socgen/arc

```
FROM dtr-dev.socgen/gts/rhel7.2:1.0

RUN yum install –y tar unzip wget git hostname && yum
clean all

ENV JAVA_HOME /opt/java
RUN wget https://itbox.socgen/…/jdk-1.8.0_92-linux-
x64.zip && \
    unzip –qq /jdk-1.8.0_92-linux-x64.zip –d /opt/ && \
    ...

RUN wget https://itbox.socgen/…/jce-policy-8.zip && \
     unzip –qq jce-policy-8.zip –d ...
```

Base Service Image: Every 2 months

springboot-microservice

dtr-dev.socgen/fpl-95595

```
FROM dtr-dev.socgen/arc/jre8

ENV USER_HOME /home/fpluser
WORKDIR $USER_HOME
RUN groupadd ... && useradd ...

ONBUILD COPY target/*.jar ./
ONBUILD COPY target/classes/properties ./

EXPOSE 443 80
ENTRYPOINT /entrypoint.sh
```

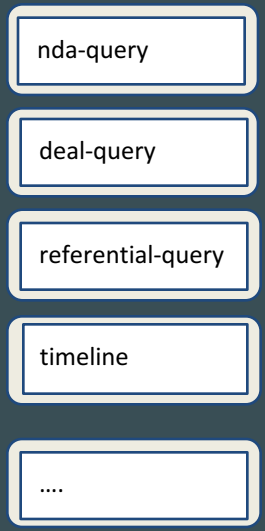
Service Image: Every release

- nda-query
- deal-query
- referential-query
- timeline
- ....

```
FROM dtr-dev.socgen/springboot-microservice
```

dtr-dev.socgen/fpl-95595

Service docker-stack: deploy time




```
version “3.1”
services:
  referential-query:
  image: “dtr-prod.socgen/fpl-95595/referential-
query:1.0
  deploy:
    replicas:4
  environment:
    - ENVNAME=$ENVNAME
    - ogn_metrics_cluster_node=ognuatap005
    - CREDENTIALS_FILE=/run/secrets/credentials.env
  secrets:
    - credentials.env
```

dtr-dev.socgen/fpl-95595

Service entrypoint.sh: deploy time

Configuration monkey patching

```
#!/bin/sh

source $CREDENTIALS_FILE

./convert_cmdb_yaml_to_properties.sh
./create_keystore_via_itaas_cn.sh

envsubt < /props/app.properties
./replace_placeholders_in_web_conf.sh

java –jar ${springboot-app}.jar –config-files
/props/app.properties
```

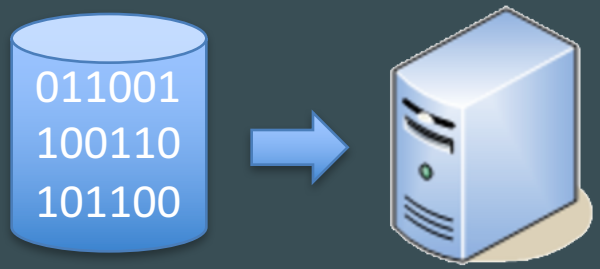
Configuration

Storage

Save files where?

It’s just like coding…

Before: Configuration as **data**



- Complicated Tooling
- Different tool for Ops

After: Configuration as **code**

```
FROM dtr-
dev.socgen/gts/rhel7.2:1.0

RUN yum install –y tar
unzip wget git hostname &&
yum clean all

ENV JAVA_HOME /opt/java
RUN wget
https://itbox.socgen/…/jdk
```

- Lightweight text-based
- Same version control as Devs

…but think about lifecycles

Build Image Files

```
myapp/
 src/
 test/
 conf/
   app.properties
   app.xml
 Dockerfile
 entrypoint.sh
```

- Next to source code

Run Container Files

```
myapp/
  docker-compose.yml
  conf/
    app.env
```

dev

```
myapp/
  docker-compose.yml
  conf/
    app.env
```

prod

- Branches for different environments

First Try: Baked in image

```
> git clone
https://sgithub.fr/fpl-config
```


```
// this is the base image

COPY ./fpl-config ./config

// do stuff
```

Build Context

PROS

- Fully traceable
- Containers don’t need git

CONS

- Must copy all configs to be env agnostic
- Must rebuild all images when configuration changes
- Not ”build-binary only once”

Current: Not baked in image

```
// this is the base image
// ...

ENTRYPOINT ["/bin/bash", "-c",
"
git clone
https://sgithub.fr/fpl-config
//do stuff "]
```

Build Context

PROS

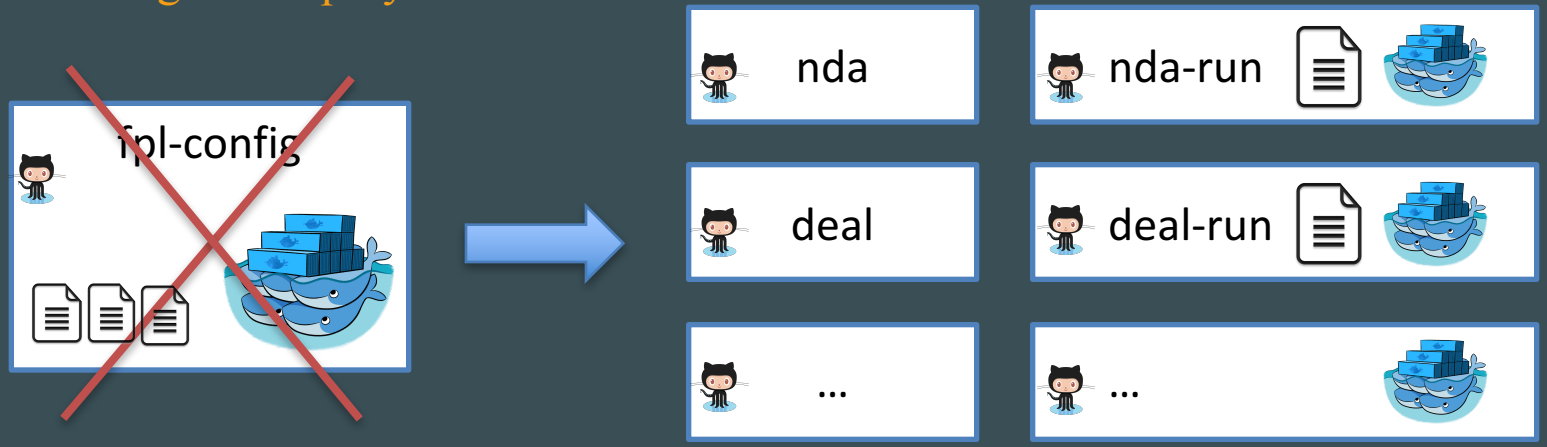
- Images are environment agnostic
- Don't need to rebuild images when configuration change

CONS

- Container needs git
- Config is not traceable unless you log the git commit somewhere

Let’s refactor !

- Dedicated ”run” repo per microservice
  - docker-compose.yml
  - Environment configuration in branches
- Blue-green deployment



Summary

- Use a release image
- Put configuration into buckets
- Separate build from run in version control

Thank You!

Docker Reference Architecture:
Development Pipeline Best Practices Using Docker EE

www.docker.com/RA_DevPipeline

@docker