Continuous Delivery with Docker Containers and Java
The Good, the Bad, and the Ugly

Daniel Bryant

@danielbryantuk

BigPictureTech

Containers: Expectations versus reality

“DevOps”

@danielbryantuk

- Tech Consultant, Product Architect at Datawire,…
  - Ex-academic, software developer, DBA, ops, CTO, conference tourist
  - Java Champion, Continuous Delivery (CI/CD) advocate
  - Leading change through technology and teams

oreil.ly/2RgU3Pe

bit.ly/2jWDSF7

Continuous Delivery & Docker

Velocity (with stability) is key to business success

"Continuous delivery is achieved when stability and speed can satisfy business demand.

Discontinuous delivery occurs when stability and speed are insufficient."

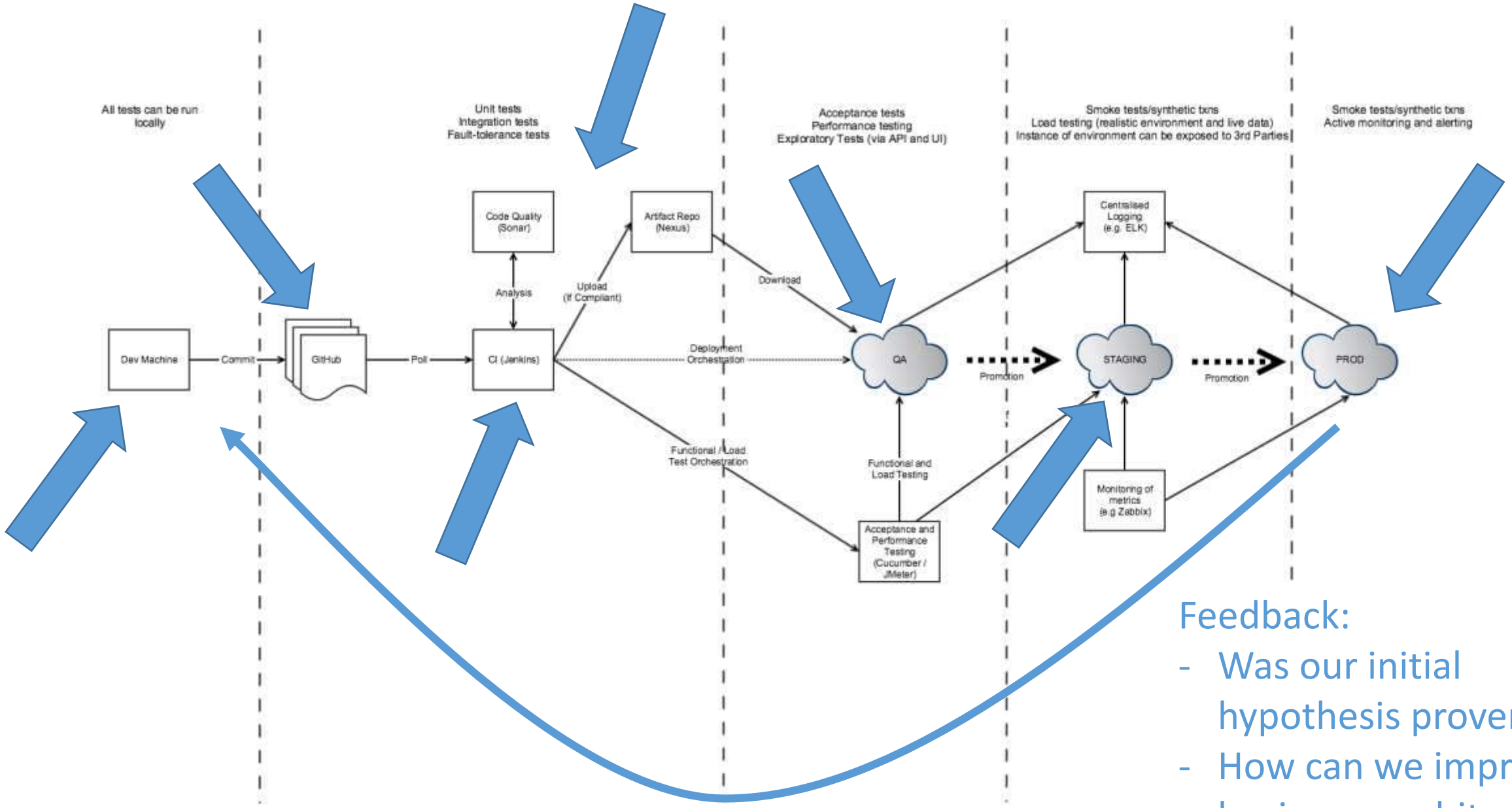
- Steve Smith (@SteveSmithCD)

Velocity (with stability) is key to business success

"Continuous delivery is achieved when **stability** and **speed** can satisfy business demand.

Discontinuous delivery occurs when stability and speed are insufficient."

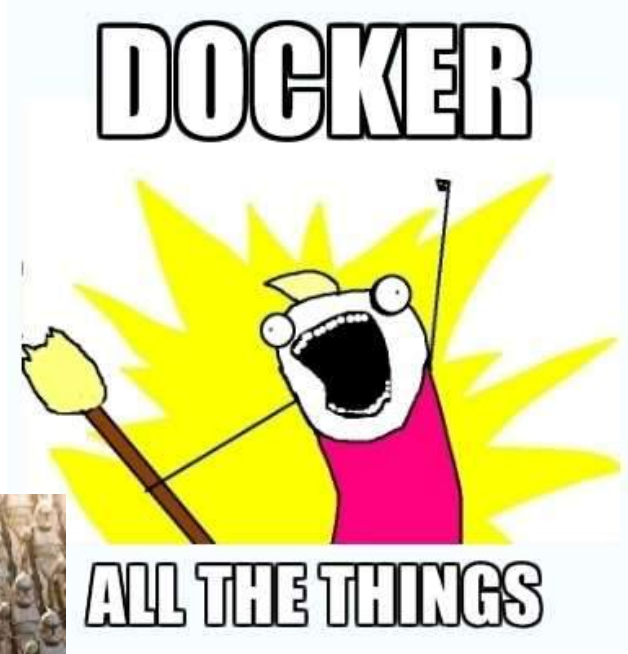
- Steve Smith (@SteveSmithCD)



Feedback:

- Was our initial hypothesis proven?
- How can we improve business, architecture and ops?

The good (with Docker and Java)



- Dev environment setup can Dockerized

- Docker enables repeatable builds

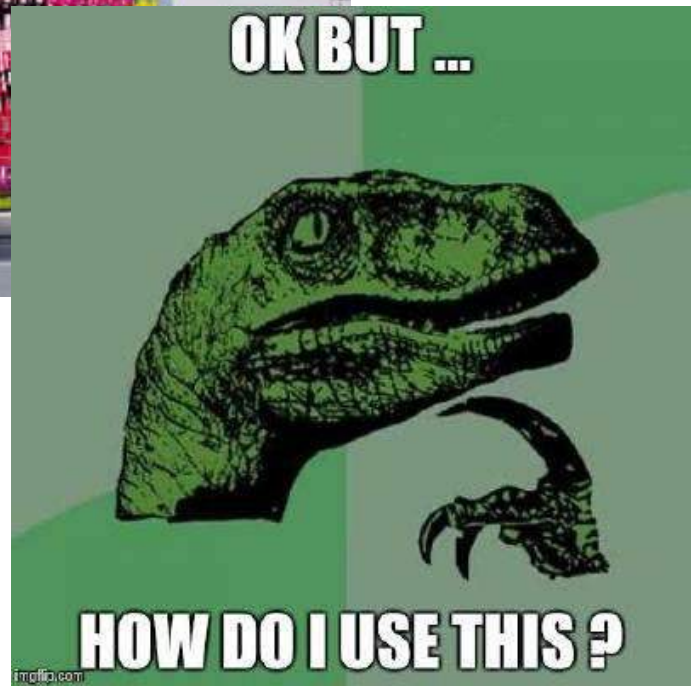
- Legacy tech (old frameworks etc) can be hermetically sealed

The bad (lessons learned for speed/stability)

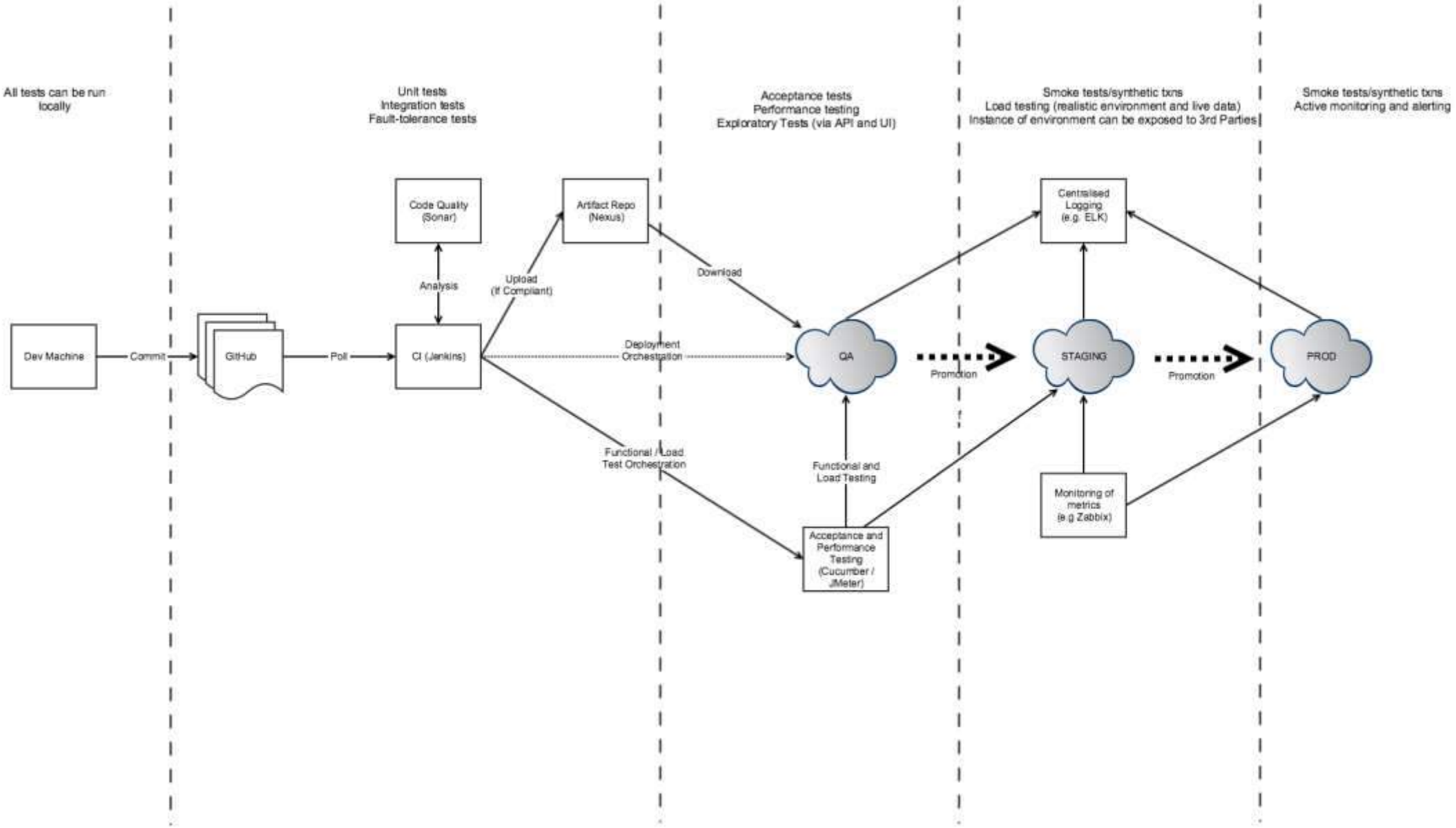
"Why is the container image 1GB?
It's a helloworld Java app!!!"

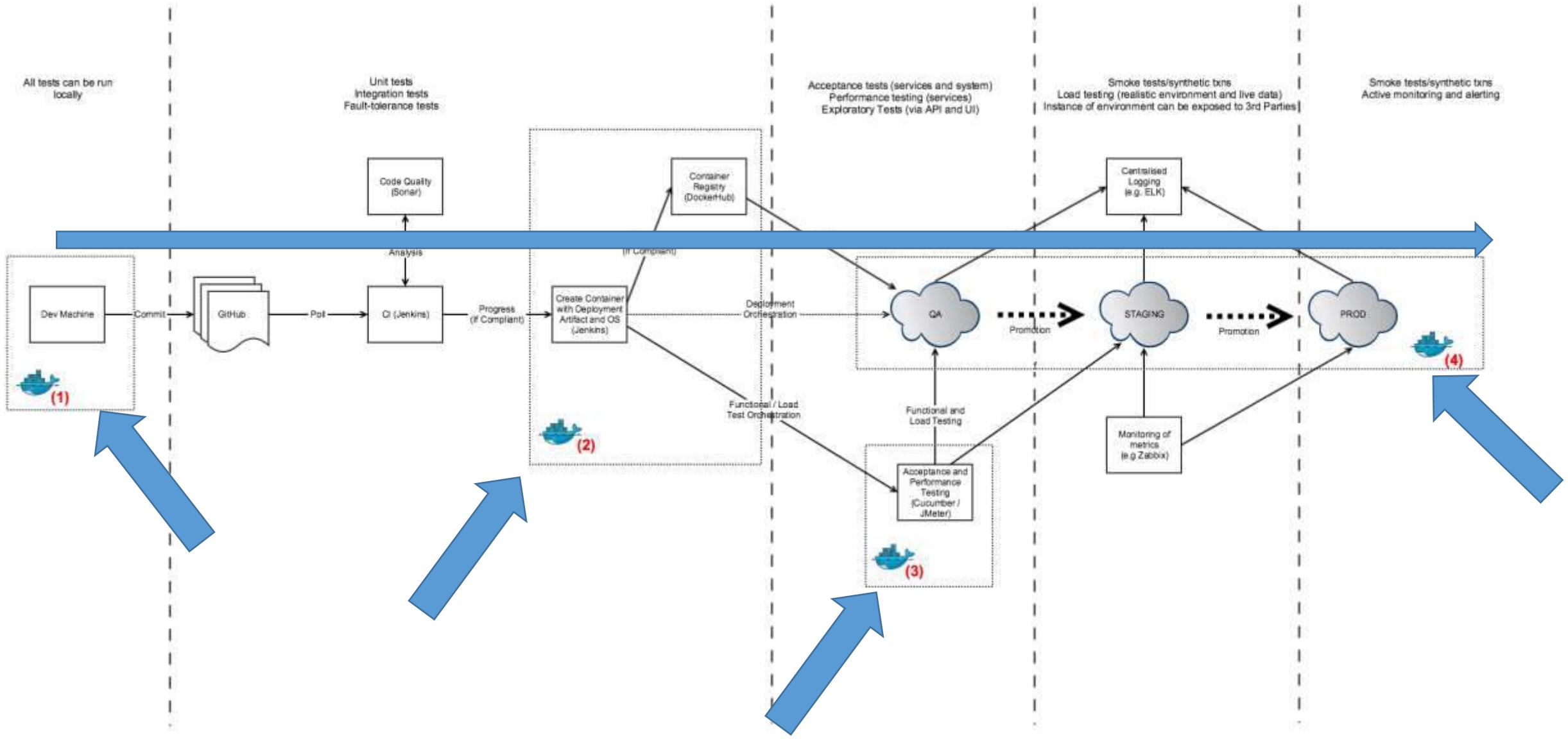
"Dinosaurs must have completed their dev/test/deploy loop faster than me"

"This Java app runs slow (or freezes) in Docker"

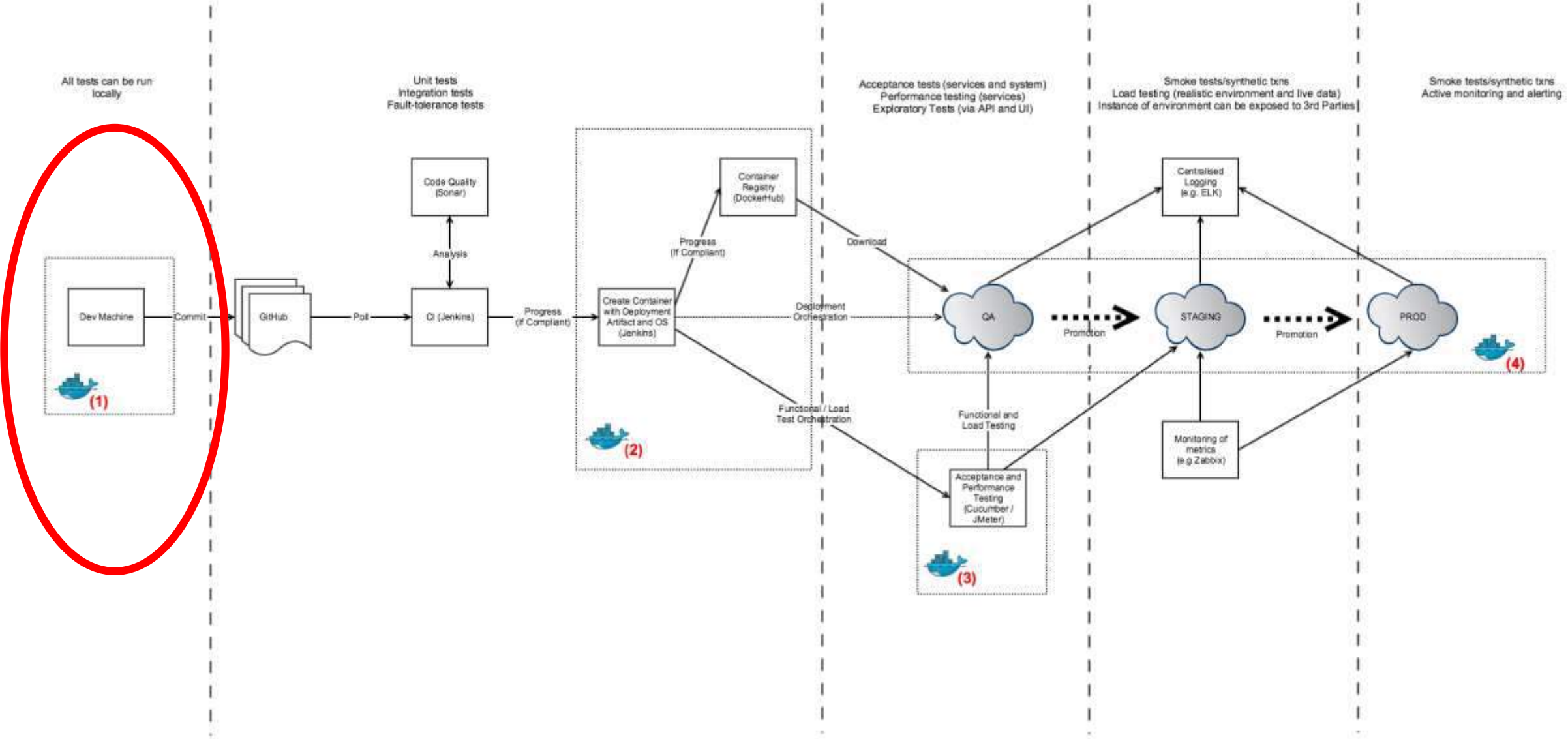


Impact of container tech on CD


All tests can be run locally
Unit tests
Integration tests
Fault-tolerance tests
Acceptance tests
Performance testing
Exploratory Tests (via API and UI)
Smoke tests/synthetic txns
Load testing (realistic environment and live data)
Instance of environment can be exposed to 3rd Parties
Smoke tests/synthetic txns
Active monitoring and alerting
Dev Machine
Commit
GitHub
Poll
CI (Jenkins)
Code Quality (Sonar)
Analysis
Upload (if Compliant)
Artifact Repo (Nexus)
Download
Deployment Orchestration
Functional / Load Test Orchestration
QA
Promotion
STAGING
Promotion
PROD
Functional and Load Testing
Acceptance and Performance Testing (Cucumber / JMeter)
Centralised Logging (e.g. ELK)
Monitoring of metrics (e.g Zabbix)


All tests can be run locally
Unit tests
Integration tests
Fault-tolerance tests
Acceptance tests (services and system)
Performance testing (services)
Exploratory Tests (via API and UI)
Smoke tests/synthetic txns
Load testing (realistic environment and live data)
Instance of environment can be exposed to 3rd Parties
Smoke tests/synthetic txns
Active monitoring and alerting
Code Quality (Sonar)
Container Registry (DockerHub)
Centralised Logging (e.g. ELK)
Analysis
(If Compliant)
Dev Machine
Commit
GitHub
Poll
CI (Jenkins)
Progress (if Compliant)
Create Container with Deployment Artifact and OS (Jenkins)
Deployment Orchestration
QA
Promotion
STAGING
Promotion
PROD
(1)
(2)
(3)
(4)
Functional / Load Test Orchestration
Functional and Load Testing
Monitoring of metrics (e.g Zabbix)
Acceptance and Performance Testing (Cucumber / JMeter)

Lessons learned


All tests can be run locally
Unit tests
Integration tests
Fault-tolerance tests
Acceptance tests (services and system)
Performance testing (services)
Exploratory Tests (via API and UI)
Smoke tests/synthetic txns
Load testing (realistic environment and live data)
Instance of environment can be exposed to 3rd Parties
Smoke tests/synthetic txns
Active monitoring and alerting
Dev Machine
Commit
GitHub
Poll
Code Quality (Sonar)
Analysis
CI (Jenkins)
Progress (if Compliant)
Create Container with Deployment Artifact and OS (Jenkins)
Progress (if Compliant)
Container Registry (DockerHub)
Download
Deployment Orchestration
Functional / Load Test Orchestration
Functional and Load Testing
Acceptance and Performance Testing (Cucumber / JMeter)
QA
Promotion
STAGING
Promotion
PROD
Centralised Logging (e.g. ELK)
Monitoring of metrics (e.g Zabbix)
(1)
(2)
(3)
(4)

Make your dev environment like production

- Must build/test containers locally
  - Perform (at least) happy path tests
- Use identical base images from production
  - With same configuration

Lesson learned: Dockerfile content is **super** important

```
1 FROM openjdk:8
2 ADD target/shopfront-0.0.1-SNAPSHOT.jar app.jar
3 EXPOSE 8010
4 ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

- OS choice (alpine or Distroless?)
- Configuration
- Build artifacts
- Exposing ports

- Oracle vs OpenJDK vs …?
- JDK vs JRE (vs jlinked binary?)
- Hotspot vs OpenJ9 vs SubstrateVM vs…?
- AOT vs JIT && CDS and ACDS

Please talk to the sysadmin people:
Their operational knowledge is invaluable

Start from good foundations: base image

```
1 FROM openjdk:8
2 ADD target/shopfront-0.0.1-SNAPSHOT.jar app.jar
3 EXPOSE 8010
4 ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

```
(master *) stockmanager $ docker image ls
REPOSITORY          TAG                          IMAGE ID        CREATED          SIZE
[redacted]
openjdk             8                            c14ba9d23b3a    2 weeks ago      624MB
openjdk             11.0.1-jdk-oraclelinux7      393d2ab988d9    3 weeks ago      463MB
maven               3.6.0-jdk-8-alpine           b3f92f93bc47    4 weeks ago      119MB
openjdk             8u181-jre-alpine3.8          2e01f547f003    5 weeks ago      83MB
openjdk             8u181-jdk-alpine3.8          97bc1352afde    5 weeks ago      103MB
```

AdoptOpenJDK

Prebuilt OpenJDK Binaries

Java™ is the world's leading programming language and platform. The code for Java is open source and available at OpenJDK™. AdoptOpenJDK provides prebuilt OpenJDK binaries from a fully open source set of build scripts and infrastructure.
Get Docker Images on Docker Hub. Nightlies can be found in the Archive.

Download for macOS x64

1. Choose a Version

- OpenJDK 8 (LTS)
- OpenJDK 11 (LTS)

2. Choose a JVM Help Me Choose

- HotSpot
- OpenJ9

Latest release
jdk8u192-b12 - 72 MB

Other platforms

Archive

https://adoptopenjdk.net/

English

Start from good foundations: base image

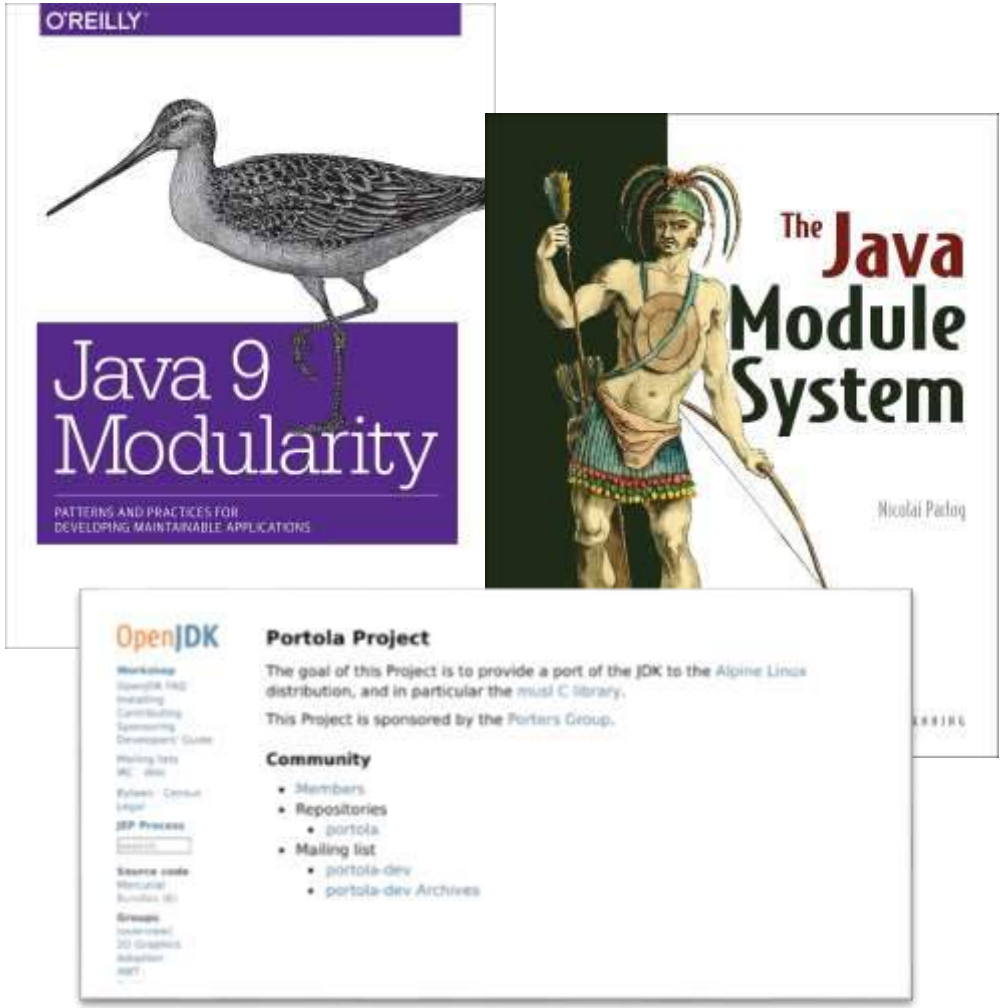
```
FROM openjdk:8
```

```
FROM openjdk:8u181-jdk-alpine3.8
```

```
FROM openjdk:8u181-jre-alpine3.8
ADD target/shopfront-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8010
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

```
(master *) stockmanager $ docker image ls
REPOSITORY          TAG                      IMAGE ID        CREATED          SIZE
openjdk             8                        c14ba9d23b3a    2 weeks ago      624MB
openjdk             11.0.1-jdk-oraclelinux7  393d2ab988d9    3 weeks ago      463MB
maven               3.6.0-jdk-8-alpine       b3f92f93bc47    4 weeks ago      119MB
openjdk             8u181-jre-alpine3.8      2e01f547f003    5 weeks ago      83MB
openjdk             8u181-jdk-alpine3.8      97bc1352afde    5 weeks ago      103MB
```

Getting smaller with Java 9, Java 11 LTS, Java 12



- Use jlink to create custom JRE
- Binary contains only:
  - Your app modules
  - Dependencies (JARs, modules)
  - JRE modules needed
- Portola Project (JDK 12+)

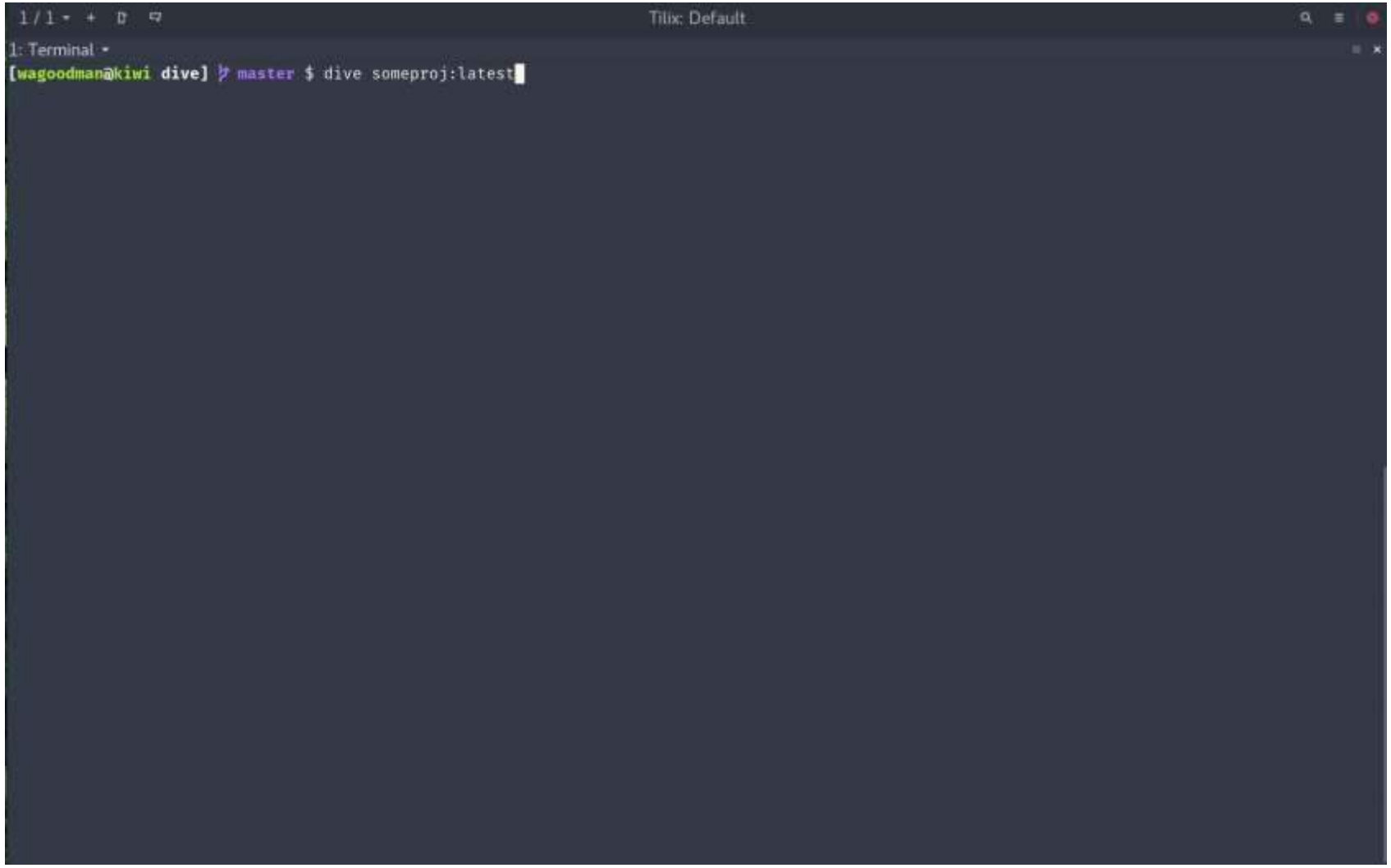
But, why is my image so big?

```
(master *) stockmanager $ mvn dependency:tree
[INFO] Scanning for projects...
[INFO]
[INFO] -------------< uk.co.danielbryant.djshopping:stockmanager >-------------
[INFO] Building stockmanager 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-dependency-plugin:2.10:tree (default-cli) @ stockmanager ---
[INFO] uk.co.danielbryant.djshopping:stockmanager:jar:0.0.1-SNAPSHOT
[INFO] +- org.springframework.boot:spring-boot-starter-web:jar:1.5.7.RELEASE:compile
[INFO] |  +- org.springframework.boot:spring-boot-starter:jar:1.5.7.RELEASE:compile
[INFO] |  |  +- org.springframework.boot:spring-boot:jar:1.5.7.RELEASE:compile
[INFO] |  |  +- org.springframework.boot:spring-boot-autoconfigure:jar:1.5.7.RELEASE:compile
[INFO] |  |  +- org.springframework.boot:spring-boot-starter-logging:jar:1.5.7.RELEASE:compile
[INFO] |  |  |  +- ch.qos.logback:logback-classic:jar:1.1.11:compile
[INFO] |  |  |  |  \- ch.qos.logback:logback-core:jar:1.1.11:compile
[INFO] |  |  |  +- org.slf4j:jul-to-slf4j:jar:1.7.25:compile
[INFO] |  |  |  \- org.slf4j:log4j-over-slf4j:jar:1.7.25:compile
[INFO] |  |  \- org.yaml:snakeyaml:jar:1.17:runtime
[INFO] |  +- org.springframework.boot:spring-boot-starter-tomcat:jar:1.5.7.RELEASE:compile
[INFO] |  |  +- org.apache.tomcat.embed:tomcat-embed-core:jar:8.5.20:compile
[INFO] |  |  +- org.apache.tomcat.embed:tomcat-embed-el:jar:8.5.20:compile
[INFO] |  |  \- org.apache.tomcat.embed:tomcat-embed-websocket:jar:8.5.20:compile
[INFO] |  +- org.hibernate:hibernate-validator:jar:5.3.5.Final:compile
[INFO] |  |  +- javax.validation:validation-api:jar:1.1.0.Final:compile
[INFO] |  |  +- org.jboss.logging:jboss-logging:jar:3.3.1.Final:compile
[INFO] |  |  \- com.fasterxml:classmate:jar:1.3.4:compile
[INFO] |  +- com.fasterxml.jackson.core:jackson-databind:jar:2.8.10:compile
[INFO] |  |  +- com.fasterxml.jackson.core:jackson-annotations:jar:2.8.0:compile
```

```
(master *) stockmanager $ mvn dependency:analyze
```

```
[INFO] --- maven-dependency-plugin:2.10:analyze (default-cli) @ stockmanager ---
[WARNING] Used undeclared dependencies found:
[WARNING]    org.springframework:spring-tx:jar:4.3.11.RELEASE:compile
[WARNING]    org.springframework:spring-web:jar:4.3.11.RELEASE:compile
[WARNING]    org.springframework:spring-beans:jar:4.3.11.RELEASE:compile
[WARNING]    org.springframework.boot:spring-boot-test:jar:1.5.7.RELEASE:test
[WARNING]    org.apache.tomcat.embed:tomcat-embed-core:jar:8.5.20:compile
[WARNING]    org.springframework.boot:spring-boot-autoconfigure:jar:1.5.7.RELEASE:compile
[WARNING]    org.hamcrest:hamcrest-library:jar:1.3:test
[WARNING]    org.springframework:spring-test:jar:4.3.11.RELEASE:test
[WARNING]    junit:junit:jar:4.12:compile
[WARNING]    info.cukes:cucumber-core:jar:1.2.5:compile
[WARNING]    org.slf4j:slf4j-api:jar:1.7.25:compile
[WARNING]    org.springframework.data:spring-data-commons:jar:1.13.7.RELEASE:compile
[WARNING]    org.hibernate.javax.persistence:hibernate-jpa-2.1-api:jar:1.0.0.Final:compile
[WARNING]    org.springframework.boot:spring-boot:jar:1.5.7.RELEASE:compile
[WARNING]    org.springframework:spring-context:jar:4.3.11.RELEASE:compile
[WARNING] Unused declared dependencies found:
[WARNING]    org.springframework.boot:spring-boot-starter-actuator:jar:1.5.7.RELEASE:compile
[WARNING]    org.springframework.boot:spring-boot-starter-test:jar:1.5.7.RELEASE:test
[WARNING]    org.springframework.boot:spring-boot-starter-data-jpa:jar:1.5.7.RELEASE:compile
```

Take a “dive” into a container



https://github.com/wagoodman/dive

stockmanager — dive openjdk:11.0.1-jdk-oraclelinux7 — 143×39

...ly-docker-java-shopping/stockmanager — dive openjdk:11.0.1-jdk-oraclelinux7 | ...ryant-uk/tmp/new/oreilly-docker-java-shopping/shopfront — vim Dockerfile | ...-bryant-uk/tmp/new/oreilly-docker-java-shopping/stockmanager — -bash

```
[• Layers]
Cmp Image ID                    Size  Command
    sha256:bcaa84a0d08577fc3f  117 MB  FROM sha256:bcaa84a0d08577fc3f
    sha256:0f19a3bf0af3caa206   18 MB  set -eux;  yum install -y   gzip
    sha256:a6766cf937d8d60df0  329 MB  set -eux;   curl -fL -o /openjdk

[Layer Details]

Digest: sha256:0f19a3bf0af3caa2065b4c13a71cfb0c044825750c717ce47291c798
ae658a86
Tar ID: ef391ae14085888645bce38ad2725964c9e1f072e0ed1a04cedd153a2e6a4d3
1
Command:
/bin/sh -c set -eux;  yum install -y   gzip   tar     freetype fontconf
ig  ;  rm -rf /var/cache/yum

[Image Details]

Total Image size: 464 MB
Potential wasted space: 14 MB
Image efficiency score: 98 %

Count   Total Space  Path
    2         11 MB  /var/lib/rpm/Packages
    2        1.1 MB  /var/lib/rpm/Basenames
    2        340 kB  /var/lib/yum/history/history-2018-11-06.sqlite
    2        291 kB  /var/lib/rpm/Dirnames
    2        135 kB  /var/lib/rpm/Providename
    2         98 kB  /var/lib/rpm/Requirename
    3         34 kB  /etc/pki/nssdb/key4.db
    2         33 kB  /var/lib/rpm/Sha1header
    3         28 kB  /etc/pki/nssdb/cert9.db
    2         26 kB  /var/lib/yum/history/history-2018-11-06.sqlite-jou
rnal
    2         16 kB  /var/lib/rpm/Obsoletename
    2         16 kB  /var/lib/rpm/Sigmd5
    2         16 kB  /var/lib/rpm/Name
    2         16 kB  /var/lib/rpm/Installtid
```

```
[Current Layer Contents]
Permission     UID:GID       Size  Filetree
-rwxrwxrwx         0:0        0 B  ├── bin → usr/bin
dr-xr-xr-x         0:0        0 B  ├── boot
drwxr-xr-x         0:0        0 B  ├── dev
-rw-------         0:0        0 B  │   ├── console
-rw-rw-rw-         0:0        0 B  │   ├── full
-rw-------         0:0        0 B  │   ├── initctl
-rw-rw-rw-         0:0        0 B  │   ├── null
-rw-rw-rw-         0:0        0 B  │   ├── ptmx
-rw-rw-rw-         0:0        0 B  │   ├── random
-rw-rw-rw-         0:0        0 B  │   ├── tty
-rw-rw-rw-         0:0        0 B  │   ├── tty0
-rw-rw-rw-         0:0        0 B  │   ├── urandom
-rw-rw-rw-         0:0        0 B  │   └── zero
drwxr-xr-x         0:0      1.8 MB ├── etc
-rw-r--r--         0:0      5.1 kB │   ├── DIR_COLORS
-rw-r--r--         0:0      5.7 kB │   ├── DIR_COLORS.256color
-rw-r--r--         0:0      4.7 kB │   ├── DIR_COLORS.lightbgcolor
-rw-r--r--         0:0        94 B │   ├── GREP_COLORS
drwxr-xr-x         0:0        0 B  │   ├── X11
drwxr-xr-x         0:0        0 B  │   │   ├── applnk
drwxr-xr-x         0:0        0 B  │   │   └── fontpath.d
-rw-r--r--         0:0      1.5 kB │   ├── aliases
drwxr-xr-x         0:0        0 B  │   ├── alternatives
-rwxrwxrwx         0:0        0 B  │   │   └── libnssckbi.so.x86_64 → /u
drwxr-xr-x         0:0       11 kB │   ├── bash_completion.d
-rw-r--r--         0:0       11 kB │   │   └── yum-utils.bash
-rw-r--r--         0:0      2.9 kB │   ├── bashrc
drwxr-xr-x         0:0        0 B  │   ├── chkconfig.d
-rw-r--r--         0:0      1.6 kB │   ├── csh.cshrc
-rw-r--r--         0:0      866 B  │   ├── csh.login
drwxr-xr-x         0:0      1.9 kB │   ├── default
-rw-r--r--         0:0      1.8 kB │   │   ├── nss
-rw-r--r--         0:0      119 B  │   │   └── useradd
-rw-r--r--         0:0        0 B  │   ├── environment
-rw-r--r--         0:0        0 B  │   ├── exports
-rw-r--r--         0:0       70 B  │   ├── filesystems
```

stockmanager — dive openjdk:11.0.1-jdk-oraclelinux7 — 143×39

...ly-docker-java-shopping/stockmanager — dive openjdk:11.0.1-jdk-oraclelinux7 | ...ryant-uk/tmp/new/oreilly-docker-java-shopping/shopfront — vim Dockerfile | ...-bryant-uk/tmp/new/oreilly-docker-java-shopping/stockmanager — -bash

```
[• Layers]
Cmp  Image ID                    Size  Command
     sha256:bcaa84a0d08577fc3f  117 MB  FROM sha256:bcaa84a0d08577fc3f
     sha256:0f19a3bf0af3caa206   18 MB  set -eux;  yum install -y   gzip
     sha256:a6766cf937d8d60df0  329 MB  set -eux;   curl -fL -o /openjdk

[Layer Details]

Digest: sha256:a6766cf937d8d60df031fbbfb4d97e0ca16fe1324f2e2ce5e640d0eb
22342130
Tar ID: 4697d9fb77558f2c820f9b027796ca0924689195496cd1be13c095dd80e8978
9
Command:
/bin/sh -c set -eux;   curl -fL -o /openjdk.tgz "$JAVA_URL";  echo "$JA
VA_SHA256 */openjdk.tgz" | sha256sum -c -;  mkdir -p "$JAVA_HOME";  tar
 --extract --file /openjdk.tgz --directory "$JAVA_HOME" --strip-compone
nts 1;  rm /openjdk.tgz;   ln -sfT "$JAVA_HOME" /usr/java/default;  ln
-sfT "$JAVA_HOME" /usr/java/latest;  for bin in "$JAVA_HOME/bin/"*; do
  base="$(basename "$bin")";   [ ! -e "/usr/bin/$base" ];   alternative
s --install "/usr/bin/$base" "$base" "$bin" 20000;  done;   java -Xshar
e:dump;   java --version;  javac --version

[Image Details]

Total Image size: 464 MB
Potential wasted space: 14 MB
Image efficiency score: 98 %

Count   Total Space  Path
    2         11 MB  /var/lib/rpm/Packages
    2        1.1 MB  /var/lib/rpm/Basenames
    2        340 kB  /var/lib/yum/history/history-2018-11-06.sqlite
    2        291 kB  /var/lib/rpm/Dirnames
    2        135 kB  /var/lib/rpm/Providename
    2         98 kB  /var/lib/rpm/Requirename
    3         34 kB  /etc/pki/nssdb/key4.db
    2         33 kB  /var/lib/rpm/Sha1header
    3         28 kB  /etc/pki/nssdb/cert9.db
```

```
[Current Layer Contents]
Permission     UID:GID       Size  Filetree
-rwxrwxrwx        0:0        0 B  ├── bin → usr/bin
dr-xr-xr-x        0:0        0 B  ├── boot
drwxr-xr-x        0:0        0 B  ├── dev
-rw-------        0:0        0 B  │   ├── console
-rw-rw-rw-        0:0        0 B  │   ├── full
-rw-------        0:0        0 B  │   ├── initctl
-rw-rw-rw-        0:0        0 B  │   ├── null
-rw-rw-rw-        0:0        0 B  │   ├── ptmx
-rw-rw-rw-        0:0        0 B  │   ├── random
-rw-rw-rw-        0:0        0 B  │   ├── tty
-rw-rw-rw-        0:0        0 B  │   ├── tty0
-rw-rw-rw-        0:0        0 B  │   ├── urandom
-rw-rw-rw-        0:0        0 B  │   └── zero
drwxr-xr-x        0:0      1.8 MB ├── etc
-rw-r--r--        0:0      5.1 kB │   ├── DIR_COLORS
-rw-r--r--        0:0      5.7 kB │   ├── DIR_COLORS.256color
-rw-r--r--        0:0      4.7 kB │   ├── DIR_COLORS.lightbgcolor
-rw-r--r--        0:0       94 B  │   ├── GREP_COLORS
drwxr-xr-x        0:0        0 B  │   ├── X11
drwxr-xr-x        0:0        0 B  │   │   ├── applnk
drwxr-xr-x        0:0        0 B  │   │   └── fontpath.d
-rw-r--r--        0:0      1.5 kB │   ├── aliases
drwxr-xr-x        0:0        0 B  │   ├── alternatives
-rwxrwxrwx        0:0        0 B  │   │   ├── jaotc → /usr/java/openjdk
-rwxrwxrwx        0:0        0 B  │   │   ├── jar → /usr/java/openjdk-1
-rwxrwxrwx        0:0        0 B  │   │   ├── jarsigner → /usr/java/ope
-rwxrwxrwx        0:0        0 B  │   │   ├── java → /usr/java/openjdk-
-rwxrwxrwx        0:0        0 B  │   │   ├── javac → /usr/java/openjdk
-rwxrwxrwx        0:0        0 B  │   │   ├── javadoc → /usr/java/openj
-rwxrwxrwx        0:0        0 B  │   │   ├── javap → /usr/java/openjdk
-rwxrwxrwx        0:0        0 B  │   │   ├── jcmd → /usr/java/openjdk-
-rwxrwxrwx        0:0        0 B  │   │   ├── jconsole → /usr/java/open
-rwxrwxrwx        0:0        0 B  │   │   ├── jdb → /usr/java/openjdk-1
-rwxrwxrwx        0:0        0 B  │   │   ├── jdeprscan → /usr/java/ope
-rwxrwxrwx        0:0        0 B  │   │   ├── jdeps → /usr/java/openjdk
-rwxrwxrwx        0:0        0 B  │   │   ├── jhsdb → /usr/java/openjdk
```

Building in containers (multi-stage FTW)

```
FROM maven:3.6.0-jdk-8-alpine AS BUILD
COPY src /usr/src/myapp/src
COPY pom.xml /usr/src/myapp
RUN mvn -f /usr/src/myapp/pom.xml clean package

FROM openjdk:8u181-jre-alpine3.8
COPY --from=BUILD /usr/src/myapp/target/stockmanager-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8030
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

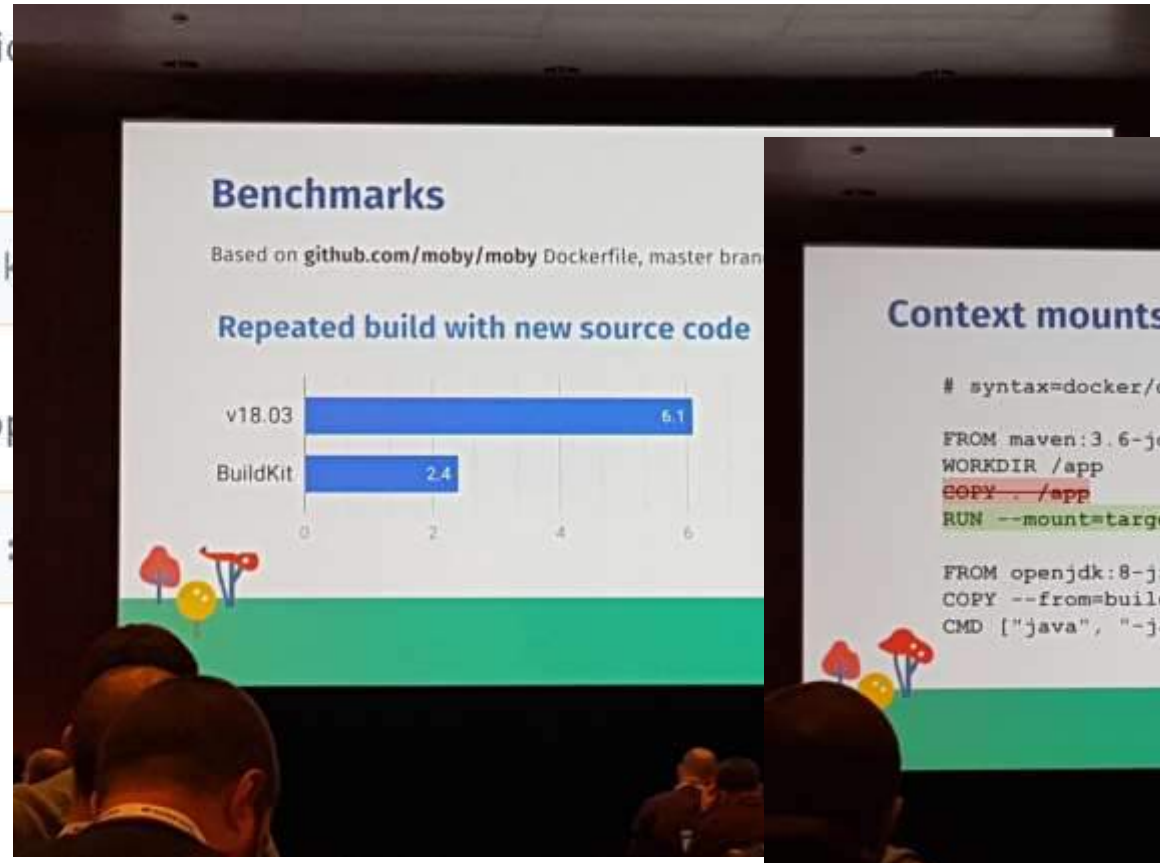
BuildKit

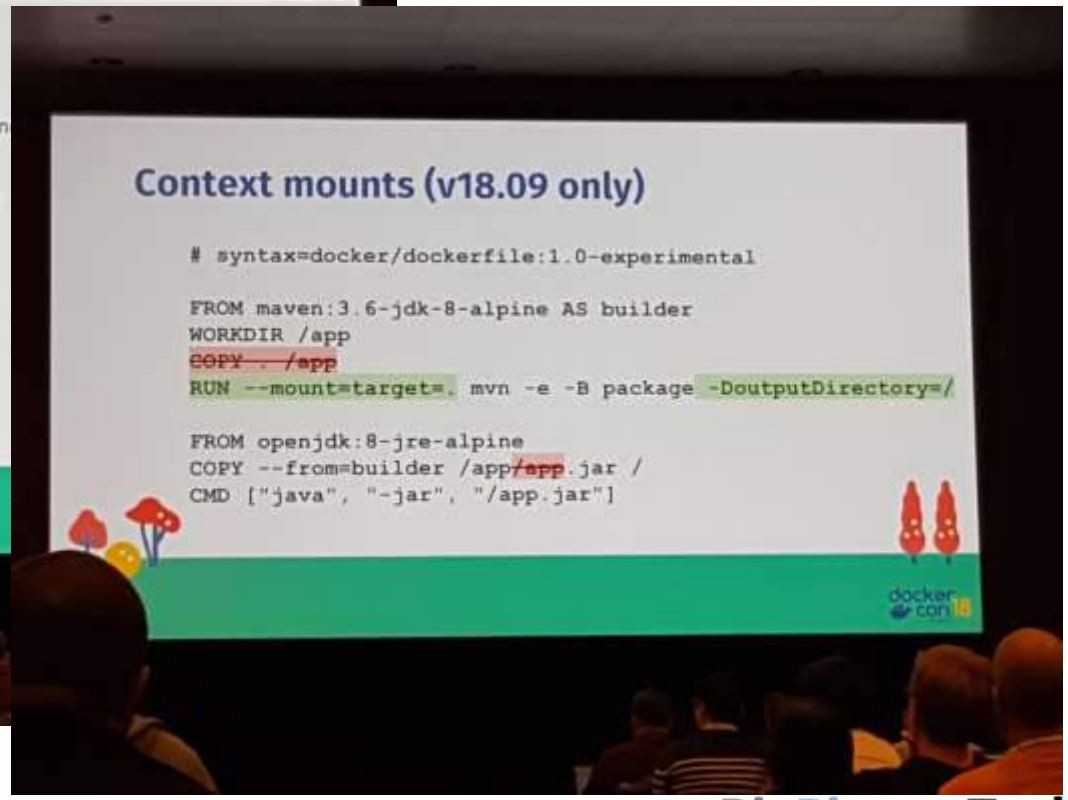
- **BuildKit is now generally available** – Access improved build performance (see slides 22-26) and scalability with the optio environment variable, e.g.

```
$ DOCKER_BUILDKIT=1 dock
```

You can also set the feature op

```
{"features":{"buildkit":
```



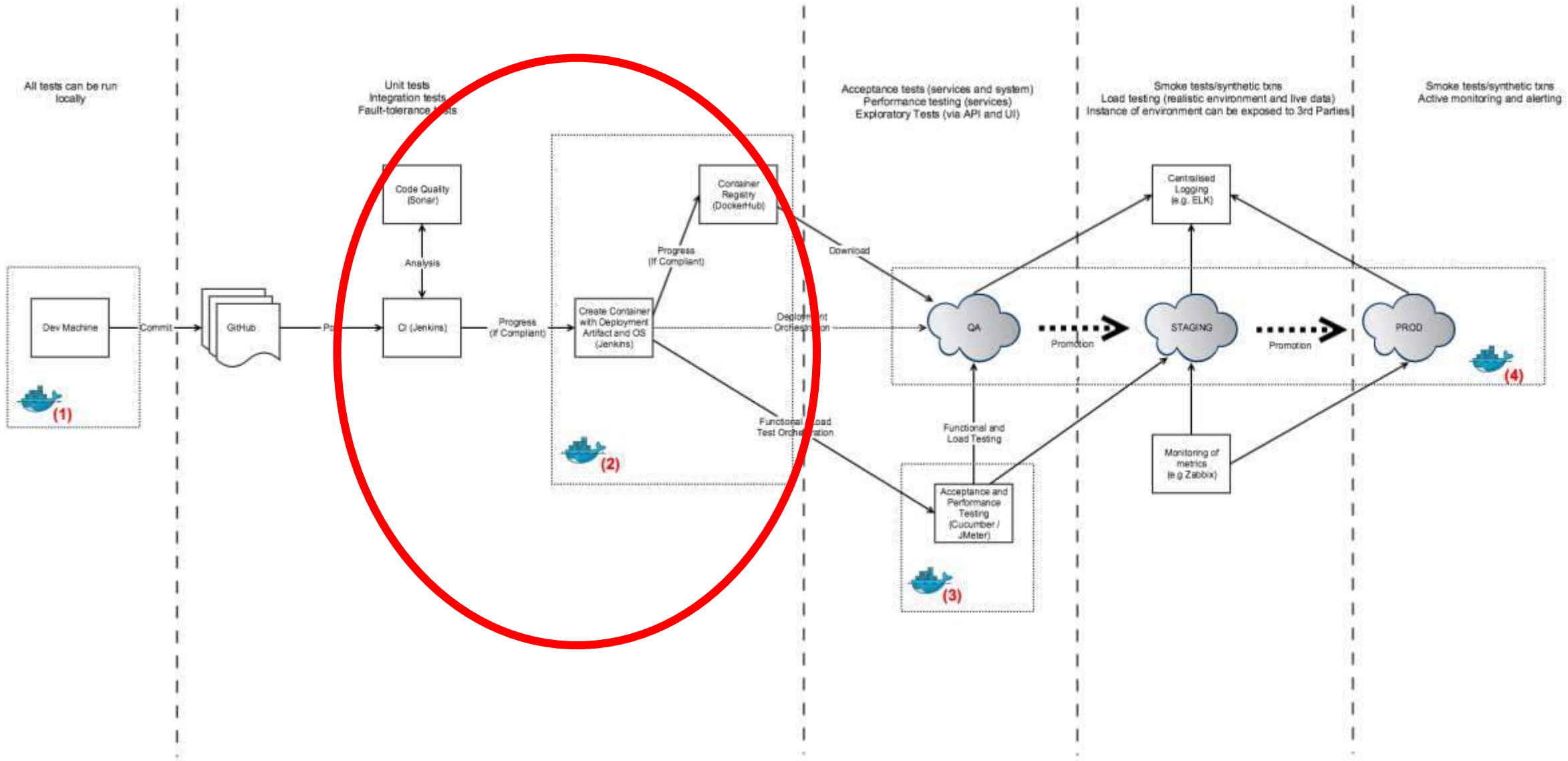


The bad: different test and prod containers?

- Create “test” version of container
  - Full OS (e.g. Ubuntu), JDK
  - Test tools and data
- Create “prod” version of the container
  - Minimal OS
  - JRE only
- Easy to see **app/configuration drift**

The bad: different test and prod containers?

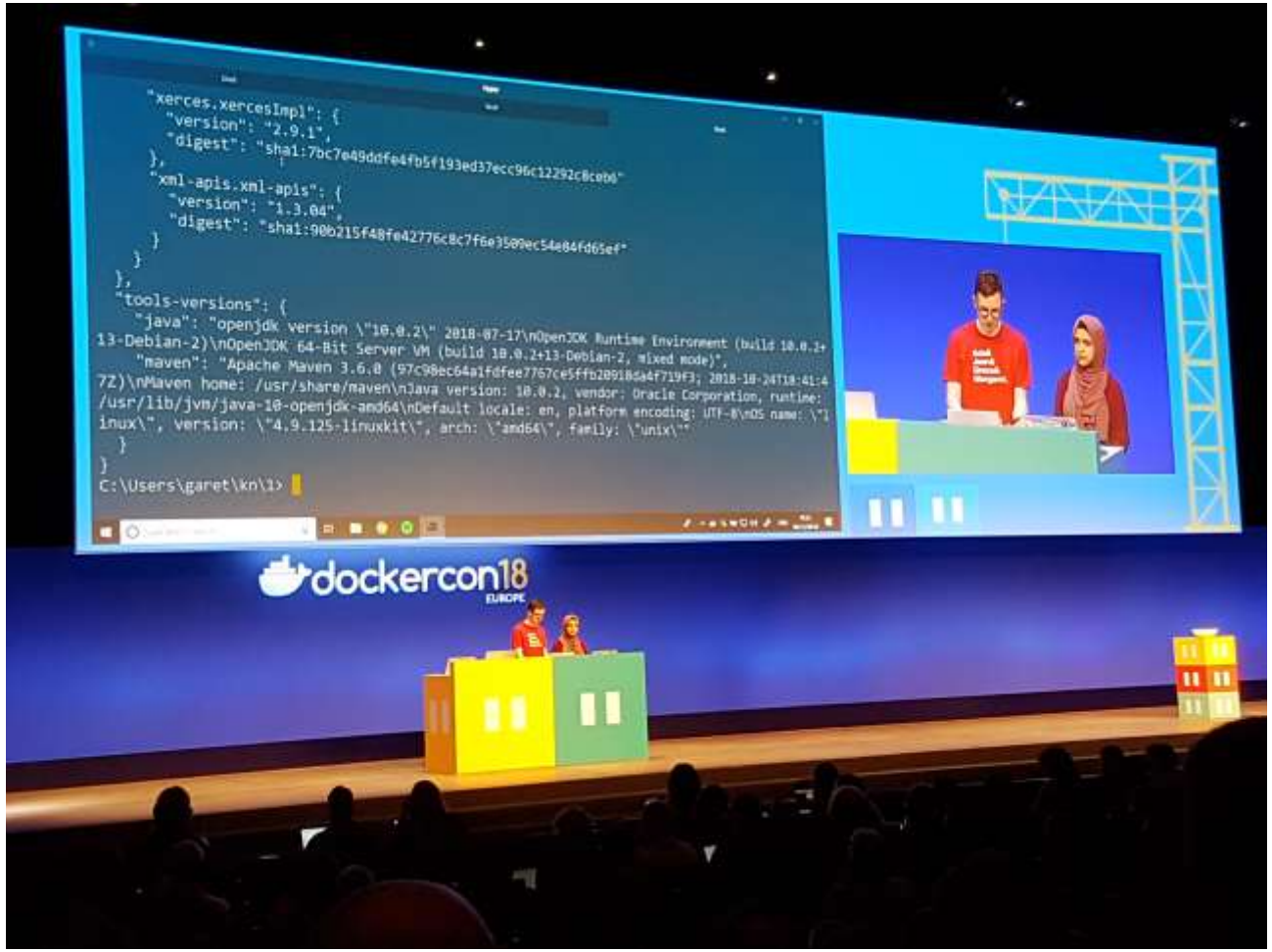
```
FROM maven:3.6.0-jdk-8-alpine AS BUILD
COPY src /usr/src/myapp/src
COPY pom.xml /usr/src/myapp
RUN mvn -f /usr/src/myapp/pom.xml clean package

FROM openjdk:8u181-jre-alpine3.8
COPY --from=BUILD /usr/src/myapp/target/stockmanager-0.0.1-SNAPSHOT.jar app.jar
# Install test tools here
# Run test tools

FROM openjdk:8u181-jdk-alpine3.8
COPY --from=BUILD /usr/src/myapp/target/stockmanager-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8030
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```


All tests can be run locally
Unit tests
Integration tests
Fault-tolerance tests
Acceptance tests (services and system)
Performance testing (services)
Exploratory Tests (via API and UI)
Smoke tests/synthetic txns
Load testing (realistic environment and live data)
Instance of environment can be exposed to 3rd Parties
Smoke tests/synthetic txns
Active monitoring and alerting
Dev Machine
Commit
GitHub
Code Quality (Sonar)
Analysis
CI (Jenkins)
Progress (if Compliant)
Create Container with Deployment Artifact and OS (Jenkins)
Progress (if Compliant)
Container Registry (Dockerhub)
Download
Deployment Orchestration
Functional Load Test Orchestration
QA
Promotion
STAGING
Promotion
PROD
Centralised Logging (e.g. ELK)
Monitoring of metrics (e.g Zabbix)
Functional and Load Testing
Acceptance and Performance Testing (Cucumber / JMeter)
(1)
(2)
(3)
(4)

Lesson learned: Metadata is valuable

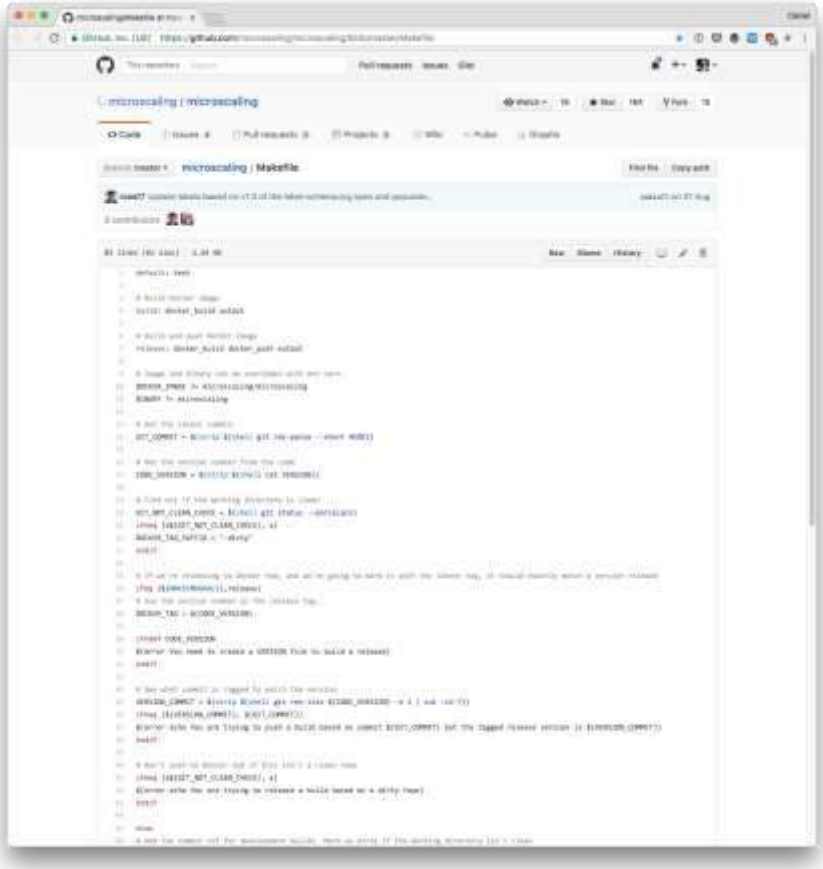
- Application metadata
  - Version / GIT SHA
- Build metadata
  - Build date
  - Image name
  - Vendor
- Quality metadata
  - QA control, signed binaries, ephemeral support
  - Security profiles (AppArmor), Security audited etc

Gareth and Amn totally stole my thunder… :)



Metadata - Adding Labels at build time

- Microscaling Systems' Makefile
- Labelling automated builds on DockerHub (h/t Ross Fairbanks)
  - Create file '/hooks/build'
- label-schema.org
- microbadger.com

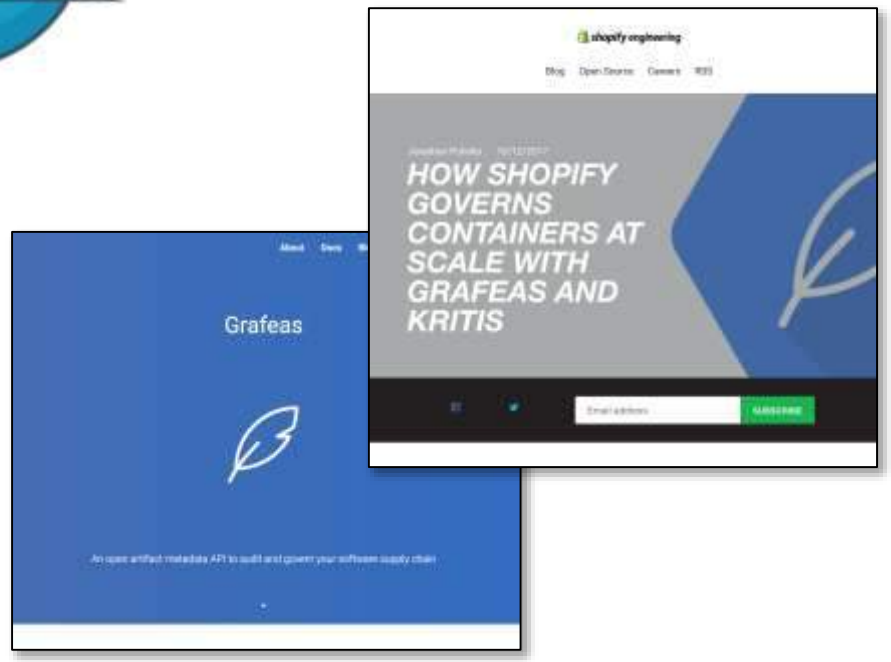


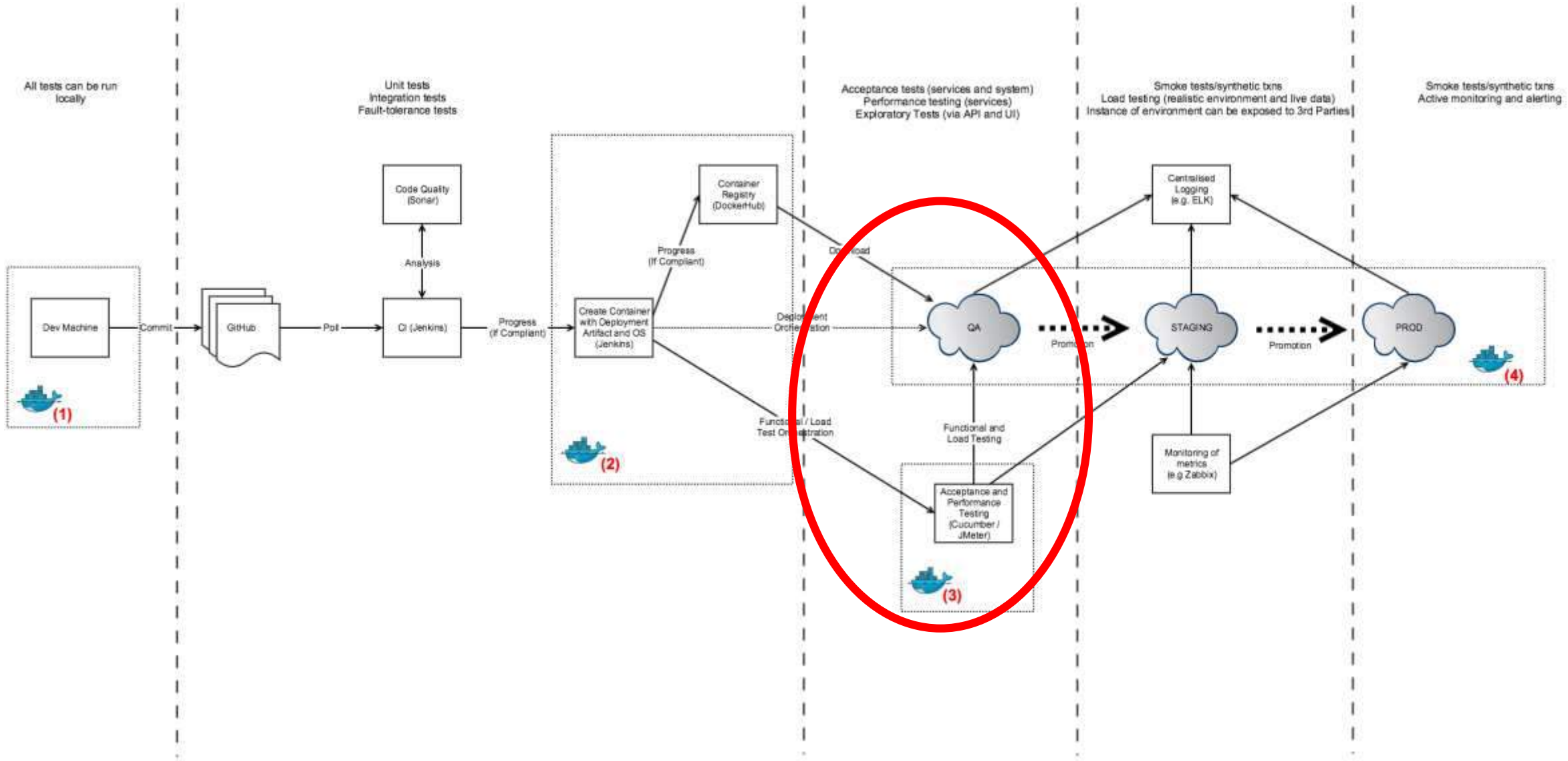
External registry with metadata support

JFrog Artifactory + Docker

NexusOSS

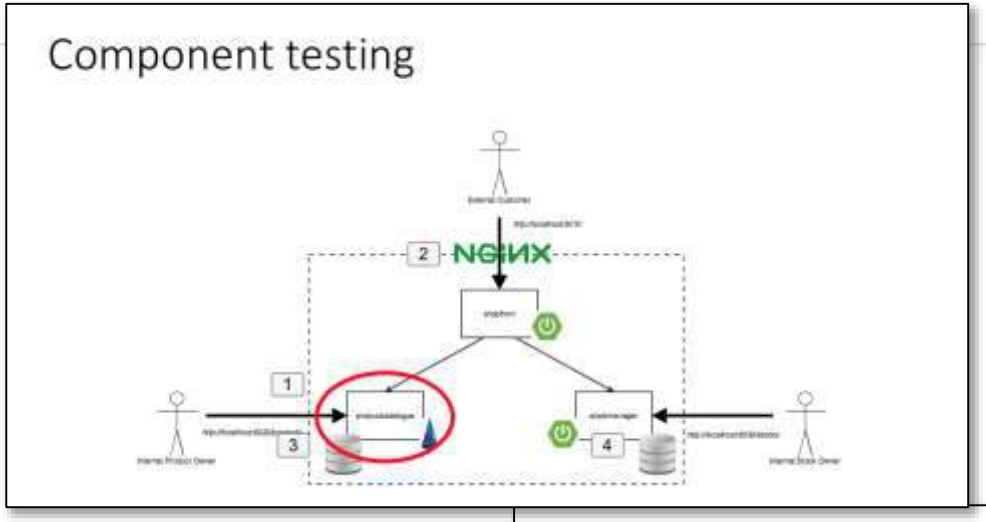
your private docker registry

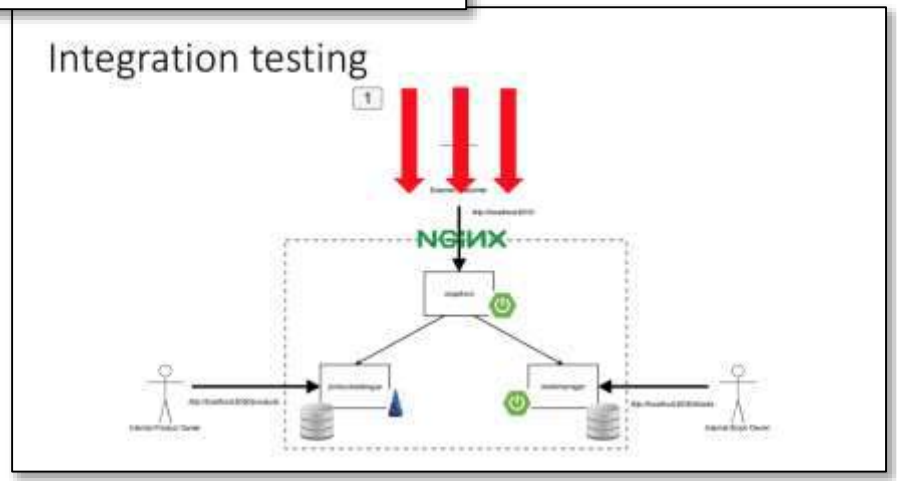



All tests can be run locally
Unit tests
Integration tests
Fault-tolerance tests
Acceptance tests (services and system)
Performance testing (services)
Exploratory Tests (via API and UI)
Smoke tests/synthetic txns
Load testing (realistic environment and live data)
Instance of environment can be exposed to 3rd Parties
Smoke tests/synthetic txns
Active monitoring and alerting
Dev Machine
Commit
GitHub
Poll
Code Quality (Sonar)
Analysis
CI (Jenkins)
Progress (if Compliant)
Create Container with Deployment Artifact and OS (Jenkins)
Progress (if Compliant)
Container Registry (DockerHub)
Download
Deployment Orchestration
Functional / Load Test Orchestration
QA
Promotion
STAGING
Promotion
PROD
Functional and Load Testing
Acceptance and Performance Testing (Cucumber / JMeter)
Centralised Logging (e.g. ELK)
Monitoring of metrics (e.g Zabbix)
(1)
(2)
(3)
(4)

Running tests with containers

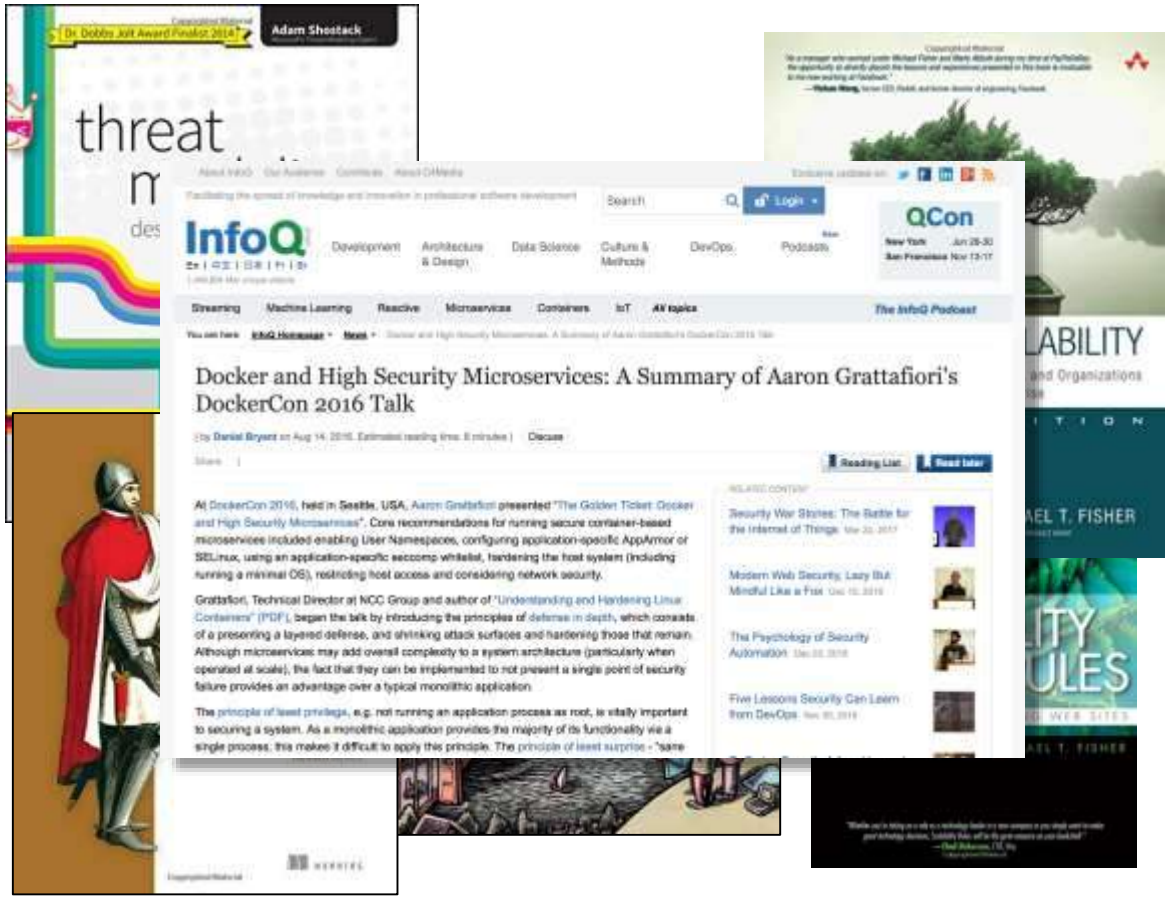
```yaml
version: '2'
services:
  shopfront:
    build: shopfront
    image: danielbryantuk/djshopfront
    ports:
     - "8010:8010"
    links:
      - productcatalogue
      - stockmanager
  productcatalogue:
    build: productcatalogue
    image: danielbryantuk/djproductcatalogue
    ports:
     - "8020:8020"
  stockmanager:
    build: stockmanager
    image: danielbryantuk/djstockmanager
    ports:
     - "8030:8030"
```





Testing NFRs in the build pipeline

- Performance and Load testing
  - Gatling / jmeter / Flood.io
- Security testing
  - Findsecbugs / OWASP Dependency check
  - Bdd-security (OWASP ZAP) / Arachni
  - Gauntlt / Serverspec
  - Docker Bench for Security / CoreOS Clair



Stability: Docker and Java

- Watch for JVM cgroup/taskset awareness (with JDK <= 8u131)
  - getAvailableProcessors() may incorrectly report the number of cpus in Docker (JDK-8140793)
  - Runtime.availableProcessors() ignores Linux taskset command (JDK-6515172)
  - *GC threads, default fork/join thread pool sizes (and others) is based from host CPU count*

- Set container memory appropriately
  - JVM requirements = Heap size (Xmx) + Metaspace + JVM overhead
  - Account for native thread requirements e.g. thread stack size (Xss)

- Entropy
  - Host entropy can soon be exhausted by crypto operations and `/dev/random` blocks
  - `-Djava.security.egd=file:/dev/./urandom` (notes on this)

Since JDK8 this has been improving with every release

JDK-8170888 Use cgroup memory limit when determining heap size *JDK 8,9*

JDK-8146115 Improve container detection and resource configuration usage *JDK 10*

JDK-8179498 attach in linux should be relative to /proc/pid/root and namespace aware

JDK-8186248 Allow more flexibility in selecting Heap % of available RAM *JDK 10*

JDK-8193710 jcmd -l and jps commands do not list JVMs in Docker containers *JDK 11*

JDK-8203357 Container Metrics *JDK 11*

JDK-8197867 Update CPU count algorithm when both cpu shares and quotas are used

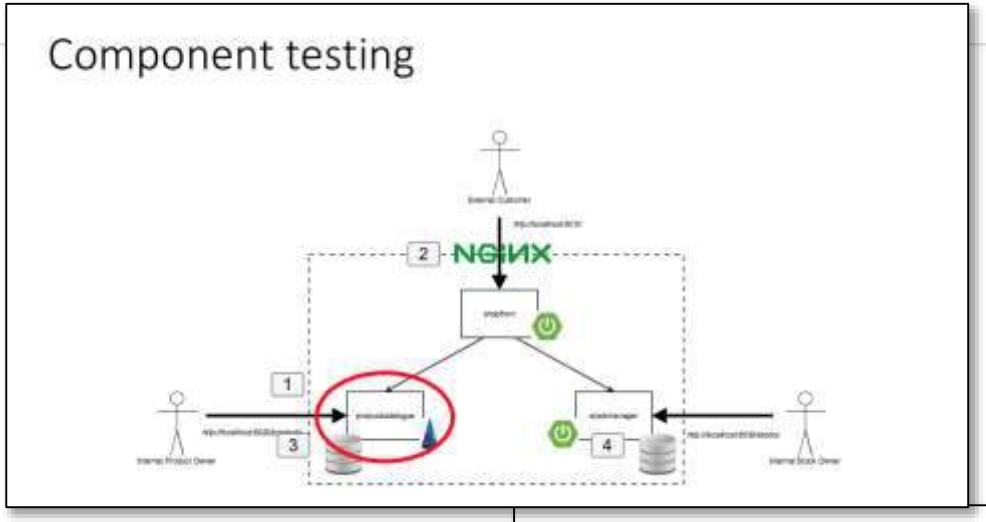
etc...

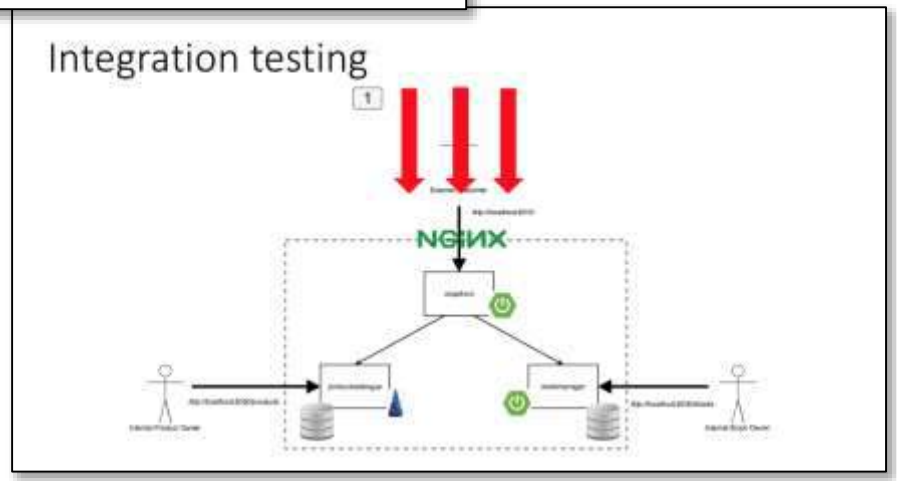
Java in a World of Containers @MaximumGilliard JavaZone 2018

https://docs.google.com/presentation/d/11VjOwW8MjDqXX9uRx0BEGYrIQtGGcXJJWMxS2q-02nA/edit#slide=id.g3c0528a66b_1_162

Running tests with containers

```yaml
version: '2'
services:
  shopfront:
    build: shopfront
    image: danielbryantuk/djshopfront
    ports:
     - "8010:8010"
    links:
      - productcatalogue
      - stockmanager
  productcatalogue:
    build: productcatalogue
    image: danielbryantuk/djproductcatalogue
    ports:
     - "8020:8020"
  stockmanager:
    build: stockmanager
    image: danielbryantuk/djstockmanager
    ports:
     - "8030:8030"
```





Security Visibility: Basic (Java) Code Scanning

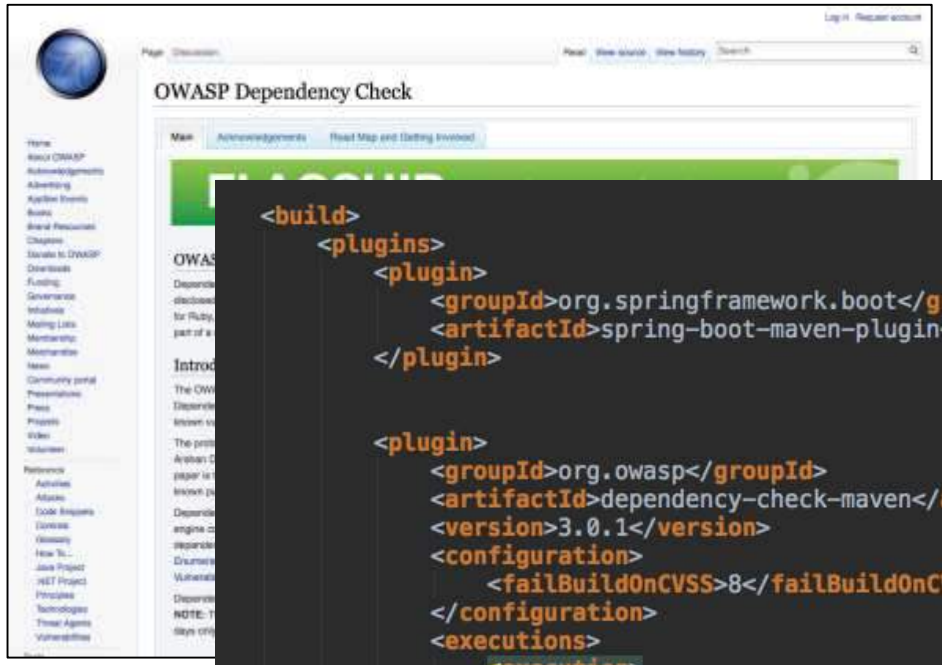
Example 11-8. Example simple Java application with obvious security issues

```
print "Hello World"package uk.co.danielbryant.oreillyexamples.builddemo;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.IOException;
import java.util.Random;

public class LoggingDemo {

    public static final Logger LOGGER = LoggerFactory.getLogger(LoggingDemo.class);

    public static void main(String[] args) {
        LOGGER.info("Hello, (Logging) World!");
        Random random = new Random();
        String myBadRandomNumString = Long.toHexString(random.nextLong());

        Runtime runtime = Runtime.getRuntime();
        try {
            runtime.exec("/bin/sh -c some_tool" + args[1]);
        } catch (IOException iox) {
            LOGGER.error("Caught IOException with command", iox);
        }
    }
}
```

Figure 11-1. Output from the Maven FindBugs plugin with FindSecBugs enabled

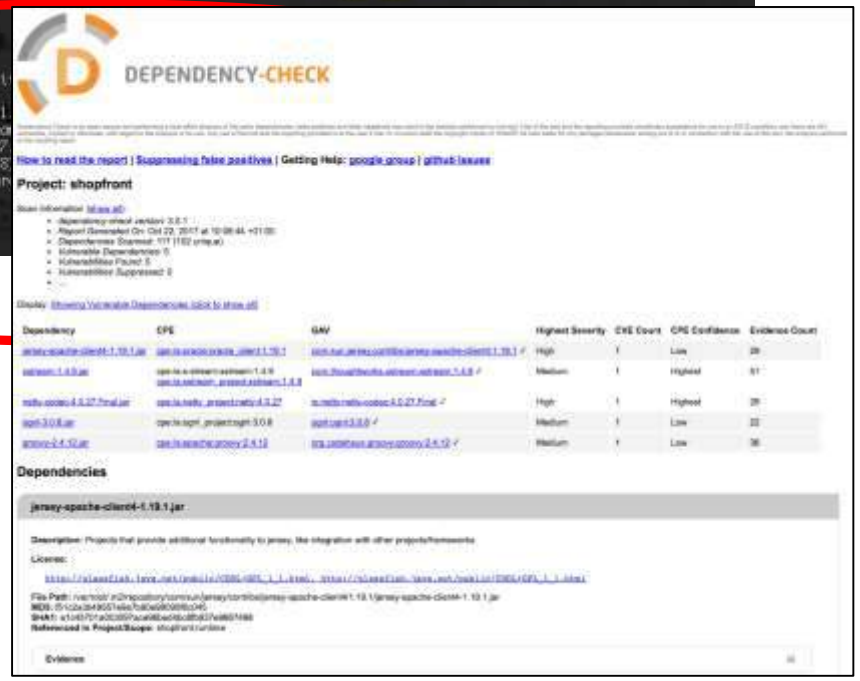
Dependency Scanning

www.owasp.org/index.php/OWASP_Dependency_Check




```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactI
        </plugin>


        <plugin>
            <groupId>org.owasp</groupId>
            <artifactId>dependency-check-maven</artifactId>
            <version>3.0.1</version>
            <configuration>
                <failBuildOnCVSS>8</failBuildOnCVSS>
            </configuration>
            <executions>
                <execution>
                    <goals>
                        <goal>check</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>

</build>
```

```
[INFO] Processing Complete for NVD CVE - 2017 (43421 ms)
[INFO] Processing Complete for NVD CVE - 2008 (44885 ms)
[INFO] Processing Complete for NVD CVE - 2010 (40114 ms)
[INFO] Processing Complete for NVD CVE - 2011 (39776 ms)
[INFO] Processing Complete for NVD CVE - 2016 (45011 ms)
[INFO] Processing Complete for NVD CVE - 2014 (41809 ms)
[INFO] Begin database maintenance.
[INFO] End database maintenance.
[INFO] Check for updates complete (91508 ms)
[INFO] Analysis Started
[INFO] Finished Archive Analyzer (0 seconds)
[INFO] Finished File Name Analyzer (0 seconds)
[INFO] Finished Jar Analyzer (1 seconds)
[INFO] Finished Central Analyzer (7 seconds)
[INFO] Finished Dependency Merging Analyzer (0 seconds)
[INFO] Finished Version Filter Analyzer (0 seconds)
[INFO] Finished Hint Analyzer (0 seconds)
[INFO] Created CPE Index (1 seconds)
[INFO] Finished CPE Analyzer (2 seconds)
[INFO] Finished False Positive Analyzer (0 seconds)
[INFO] Finished Cpe Suppression Analyzer (0 seconds)
[INFO] Finished NVD CVE Analyzer (0 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer
[INFO] Analysis Complete
[WARNING]

4 or more dependencies were identified with known vulnerabili

jersey-apache-client4-1.19.1.jar (cpe:/a:oracle:oracle_client:1
xstream-1.4.9.jar (cpe:/a:x-stream:xstream:1.4.9, cpe:/a:xstrea
netty-codec-4.0.27.Final.jar (cpe:/a:netty_project:netty:4.0.27
ognl-3.0.8.jar (ognl:ognl:3.0.8, cpe:/a:ognl_project:ognl:3.0.8
groovy-2.4.12.jar (cpe:/a:apache:groovy:2.4.12, org.codehaus.gr

See the dependency-check report for more details.
```



Static Image Scanning

github.com/arminc/clair-scanner

README.md

Clair

build passing | container ready | go report A+ | godoc reference | freenode #clair

Note: The `master` branch may be in an *unstable or* of the `master` branch in order to get stable binaries.

Clair is an open sour and docker).

1. In regular interva database.
2. Clients use the C stores them in th
3. Clients use the C features is done
4. When updates t occured.

Our goal is to enable named `Clair` after t

Getting Starte

- Learn the termin
- Watch presenta

Armin C
@acoralic

@danielbr
talk, a sim
with Clair

11:39 AM - 29 Sep

1 Like

Clair scanner

maintained yes | build passing | go report A+ | coverage 82%

Docker containers vulnerability scan

When you work with containers (Docker) you are not only packaging your application but also part of the OS. It is crucial to know what kind of libraries might be vulnerable in your container. One way to find this information is to look at the Docker registry (Hub or Quay.io) security scan. This means your vulnerable image is already on the Docker registry.

What you want is a scan as a part of CI/CD pipeline that stops the Docker image push on vulnerabilities:

1. Build and test your application
2. Build the container
3. Test the container for vulnerabilities
4. Check the vulnerabilities against allowed ones, if everything is allowed then pass ot

This straightforward process is not that easy to achieve when using the services like Do because they work asynchronously which makes it harder to do straightforwa CI/CD p

Clair to the rescue

CoreOS has created an awesome container scan tool called Clair. Clair is also used by C have is a simple tool that scans your image and compares the vulnerabilities against a w approved or not.

This is where clair-scanner comes into place. The clair-scanner does the following:

- Scans an image against Clair server
- Compares the vulnerabilities against a whitelist
- Tells you if there are vulnerabilities that are not in the whitelist and fails
- If everything is fine it completes correctly

Clair server or standalone

For the clair-scanner to work, you need a clair server. It is not always convenient to have therefore, I have created a way to run this standalone. See here https://github.com/arm

Credits

The clair-scanner is a copy of the Clair 'analyze-local-images' https://github.com/coreos/analyze-local-images with changes/improvements and addition that checks the vulnerabilities against a whitelist.

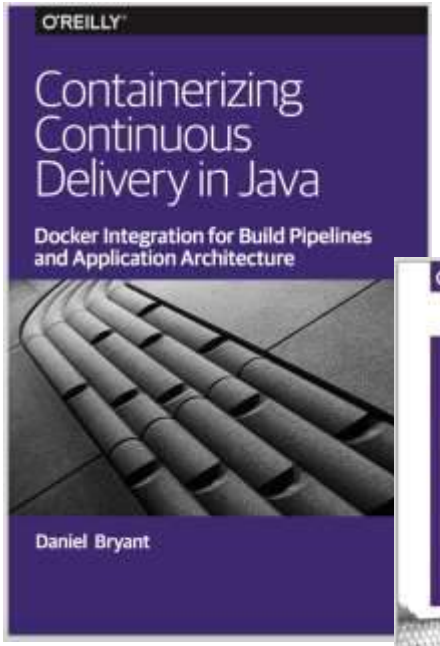
```
(master) clair-scanner $ docker pull openjdk:8-jre
8-jre: Pulling from library/openjdk
3e17c6eae66c: Pull complete
74d44b20f851: Pull complete
d156217f3fa4: Pull complete
8bce09f57693: Pull complete
268410abd9cf: Pull complete
c5e2fb2d87cf: Pull complete
09681f714a7b: Pull complete
c20de950ce52: Pull complete
Digest: sha256:d2bec505b26bddf120e5e0459468ef12de447bf0[illegible]297f9a90763c329d80536a
Status: Downloaded newer image for openjdk:8-jre
(master) clair-scanner $ ./clair-scanner --ip 192.[illegible]8.0.7 openjdk:8-jre
2017/10/23 10:50:50 [INFO] ▶ Start clair-scanner
2017/10/23 10:51:00 [INFO] ▶ Server [illegible]ing on port 9279
2017/10/23 10:51:00 [INFO] ▶ Analyzing 69357d9443c362104c1cb648c321c7c67f69e44d[illegible]
2017/10/23 10:51:03 [INFO] ▶ Analyzing 2f3fcbb54a2bd4930d7088f9a4f08374d4e0dc[illegible]
2017/10/23 10:51:03 [INFO] ▶ Analyzing 6b5a2f6d972958e8e2664df21dcb645f5a0fa[illegible]
2017/10/23 10:51:04 [INFO] ▶ Analyzing 74efab482098fe7ab7383320fb3ee4456c4a[illegible]
2017/10/23 10:51:04 [INFO] ▶ Analyzing 33d80ad21fd01d0781b3e0f17cd5333e27ca[illegible]
2017/10/23 10:51:04 [INFO] ▶ Analyzing 1deb9d8dc0891b6680bd249728b743265bca[illegible]
2017/10/23 10:51:04 [INFO] ▶ Analyzing 4234[illegible]
2017/10/23 10:51:14 [INFO] ▶ [illegible]
2017/10/23 10:51[illegible] [INFO] ▶ Unapproved vulnerabilities [[CVE-2017-9937 CVE-2011-3389 CVE-2017-11333 CVE-2017-11735 CVE-2017-2834 [illegible]
CVE-2017-12944 CVE-2014-8130 CVE-2017-13727 CVE-2017-9935 CVE-2017-13728 CVE-2017-9117 CVE-2017-11613 CVE-2017-2[illegible]
CVE-2015-3276 CVE-2017-14159 CVE-2011-3374 CVE-2005-2541 CVE-2017-5715 CVE-2017-10790 CVE-2017-11464 CVE-2013-4392 CVE-2017-1000082 CVE-2016-3177 [illegible]
CVE-2007-6755 CVE-2010-0928 CVE-2017-3735 CVE-2017-1000100 CVE-2017-1000101 CVE-2017-6313 CVE-2017-2862 CVE-2017-6311 CVE-2017-2870 CVE-2017-6314 [illegible]
CVE-2017-6312 CVE-2017-10140 CVE-2012-0039 CVE-2010-4052 CVE-2017-8804 CVE-2017-12132 CVE-2010-4756 CVE-2010-4051 CVE-2016-10228 CVE-2015-8985 CVE-2017-12133 [illegible]
CVE-2017-14062 CVE-2017-11698 CVE-2017-11696 CVE-2017-11695 CVE-2017-11697 CVE-2017-7475 CVE-2017-9814 CVE-2017-12424 CVE-2013-4235 CVE-2007-5686 CVE-2017-14062
CVE-2017-6519 CVE-2017-12562 CVE-2017-6892 CVE-2017-8872 CVE-2017-5969 CVE-2016-9318 CVE-2017-13685 CVE-2017-10989 CVE-2014-8166 CVE-2017-12883 CVE-2011-411[illegible]
CVE-2012-3378 CVE-2017-12837 CVE-2016-2779 CVE-2017-11368 CVE-2017-11462 CVE-2004-0971 CVE-2017-11113 CVE-2017-13731 CVE-2017-13729 CVE-2017-10685 CVE-2017[illegible]
CVE-2017-10684 CVE-2017-13734 CVE-2017-13730 CVE-2017-13728 CVE-2017-13731 CVE-2017-13732 CVE-2013-0340]]
```

Summary

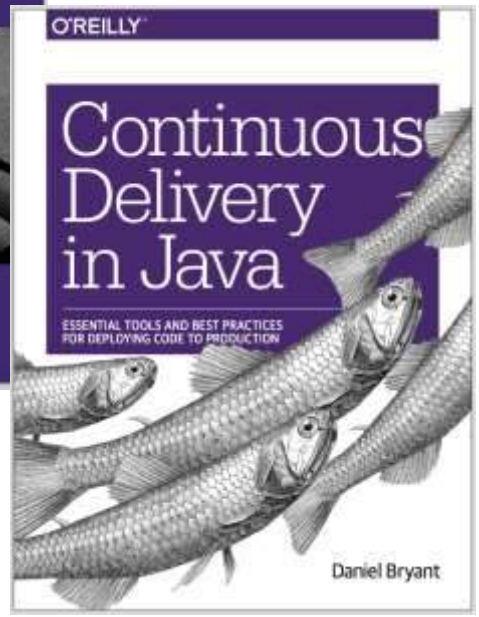
In summary

- Docker and Java are a great combination
  - But make sure you understand the technology and challenges
- Continuous delivery is essential with modern architecture/tech
  - Container images must be the (single) source of truth within pipeline
- Provence (metadata) and validation (testing NFR) of builds is vital
  - Not all developers are operationally aware

Thanks for listening…



oreil.ly/2RgU3Pe



bit.ly/2jWDSF7

Twitter: @danielbryantuk

Email: daniel.bryant@tai-dev.co.uk

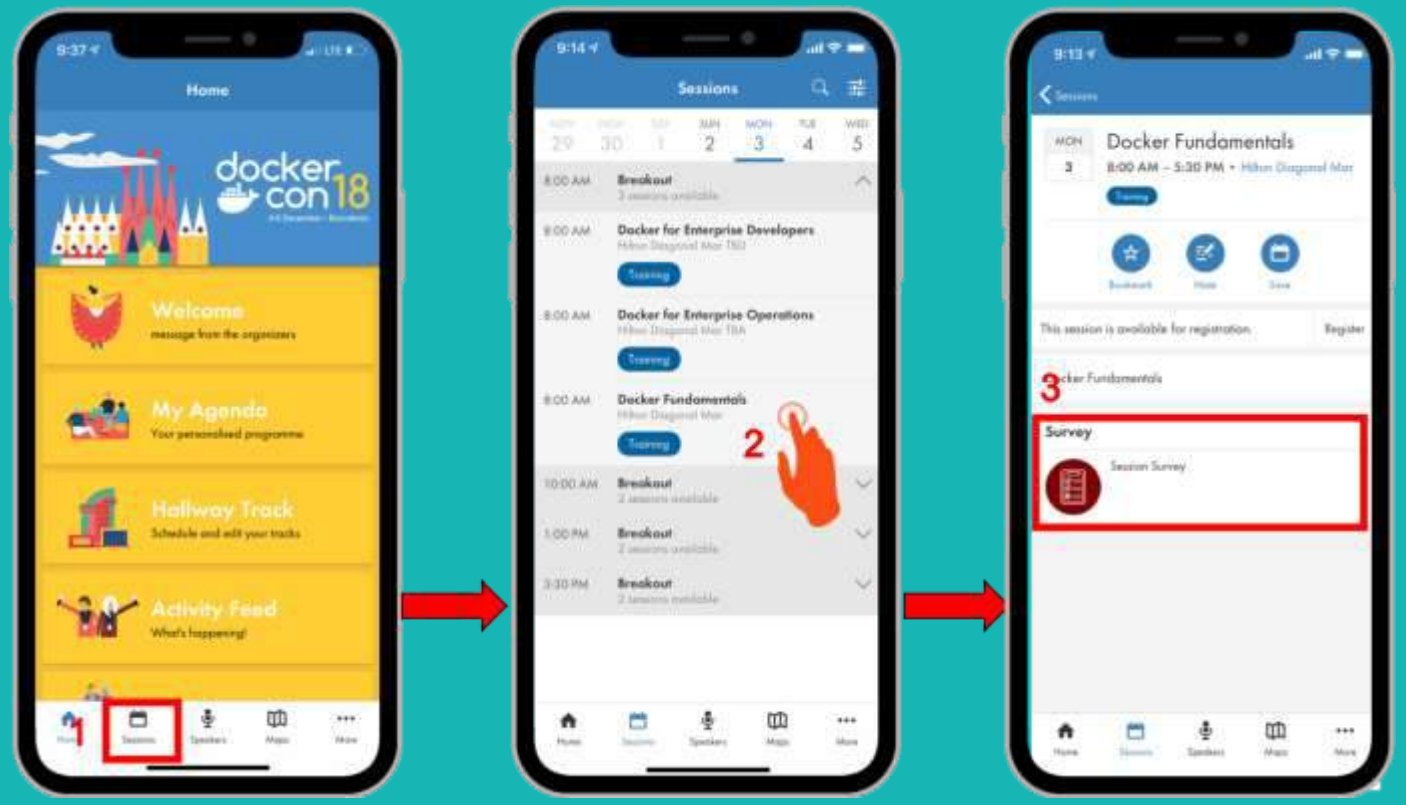
Writing: https://www.infoq.com/profile/Daniel-Bryant

Talks: https://www.youtube.com/playlist?list=PLoVYf_0qOYNeBmrpjuBOOAqJnQb3QAEtM

Take A Breakout Survey

Access your session and/or workshop surveys for the conference at any time by tapping the Sessions link on the navigation menu or block on the home screen.

Find the session/workshop you attended and tap on it to view the session details. On this page, you will find a link to the survey.



BigPictureTech

Bedtime reading