

Continuous Delivery with Docker Containers and Java

The Good, the Bad, and the Ugly

Daniel Bryant

@danielbryantuk

Containers: Expectations versus reality

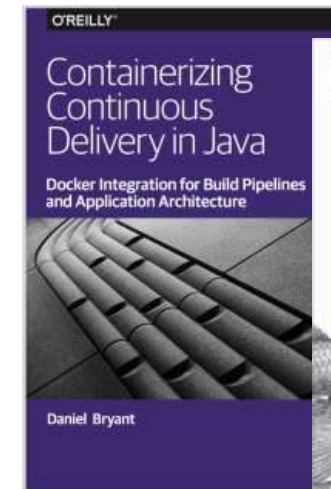


"DevOps"

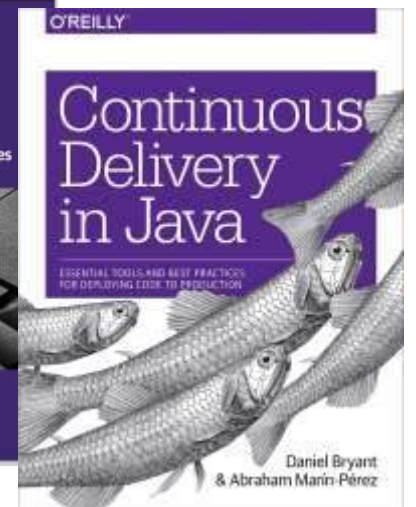
@danielbryantuk



- 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**



bit.ly/2jWDSF7



oreil.ly/2RgU3Pe

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.”

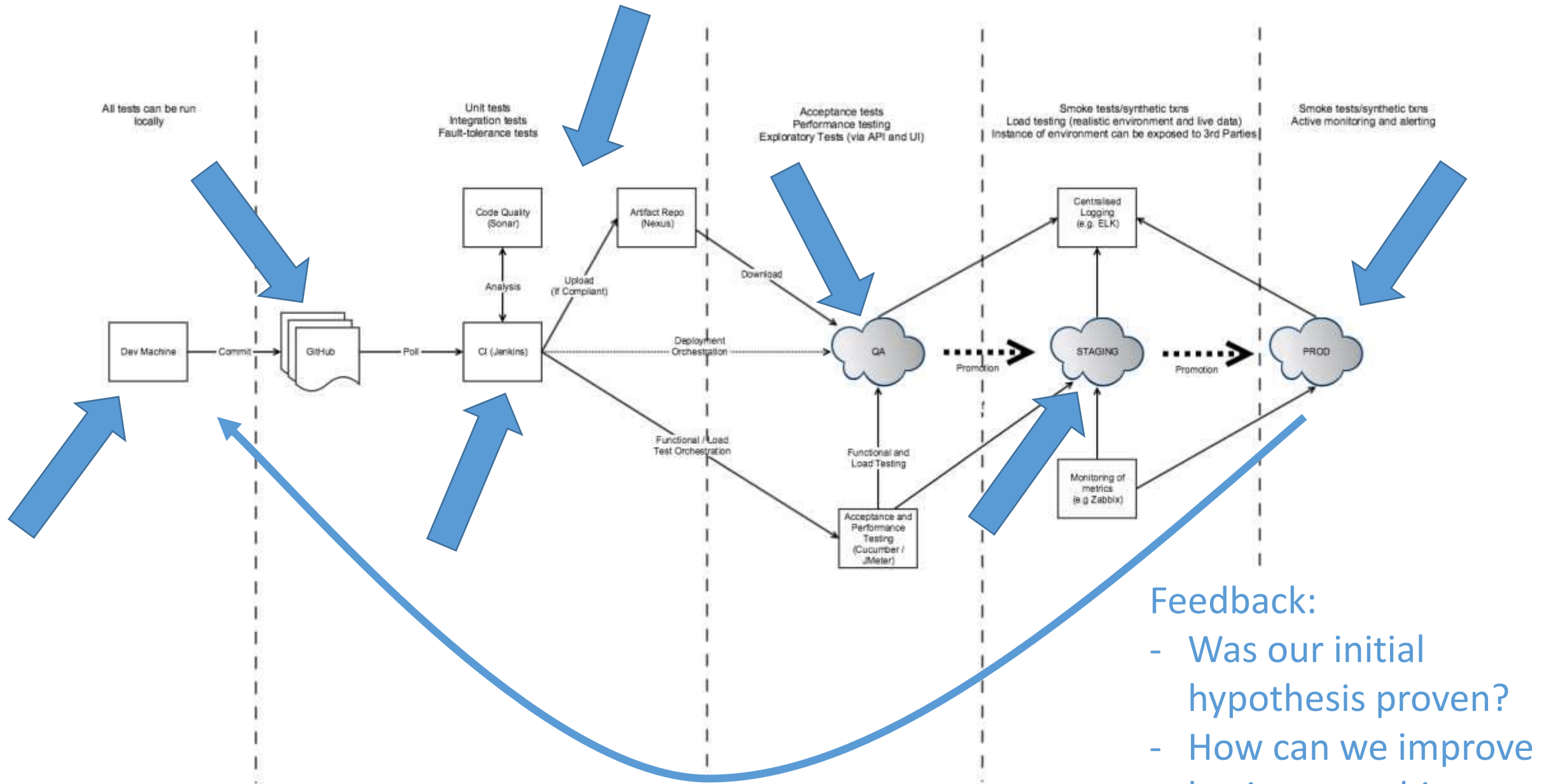
- Steve Smith (@SteveSmithCD)

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)

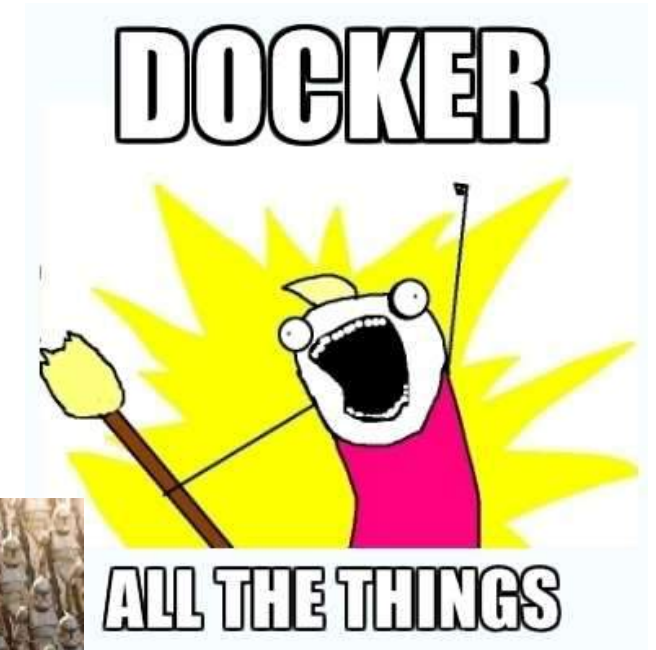


Feedback:

- 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

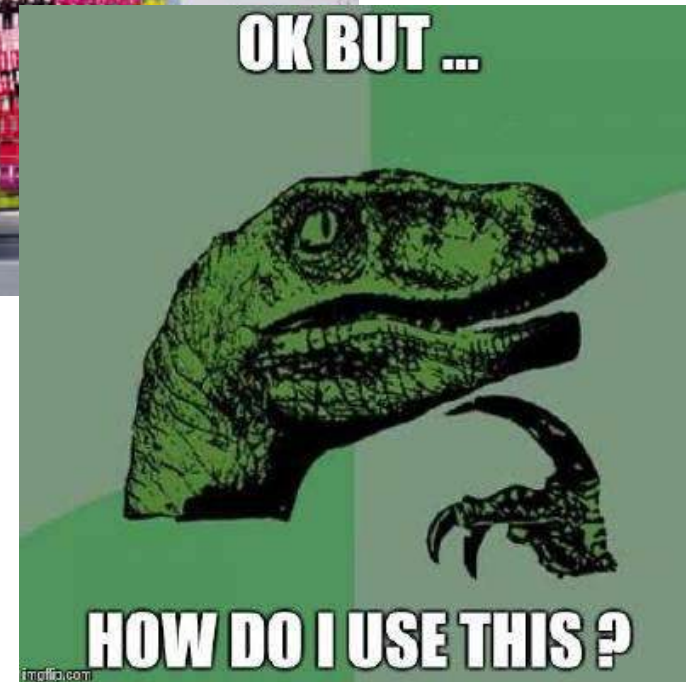


The bad (lessons learned for speed/stability)

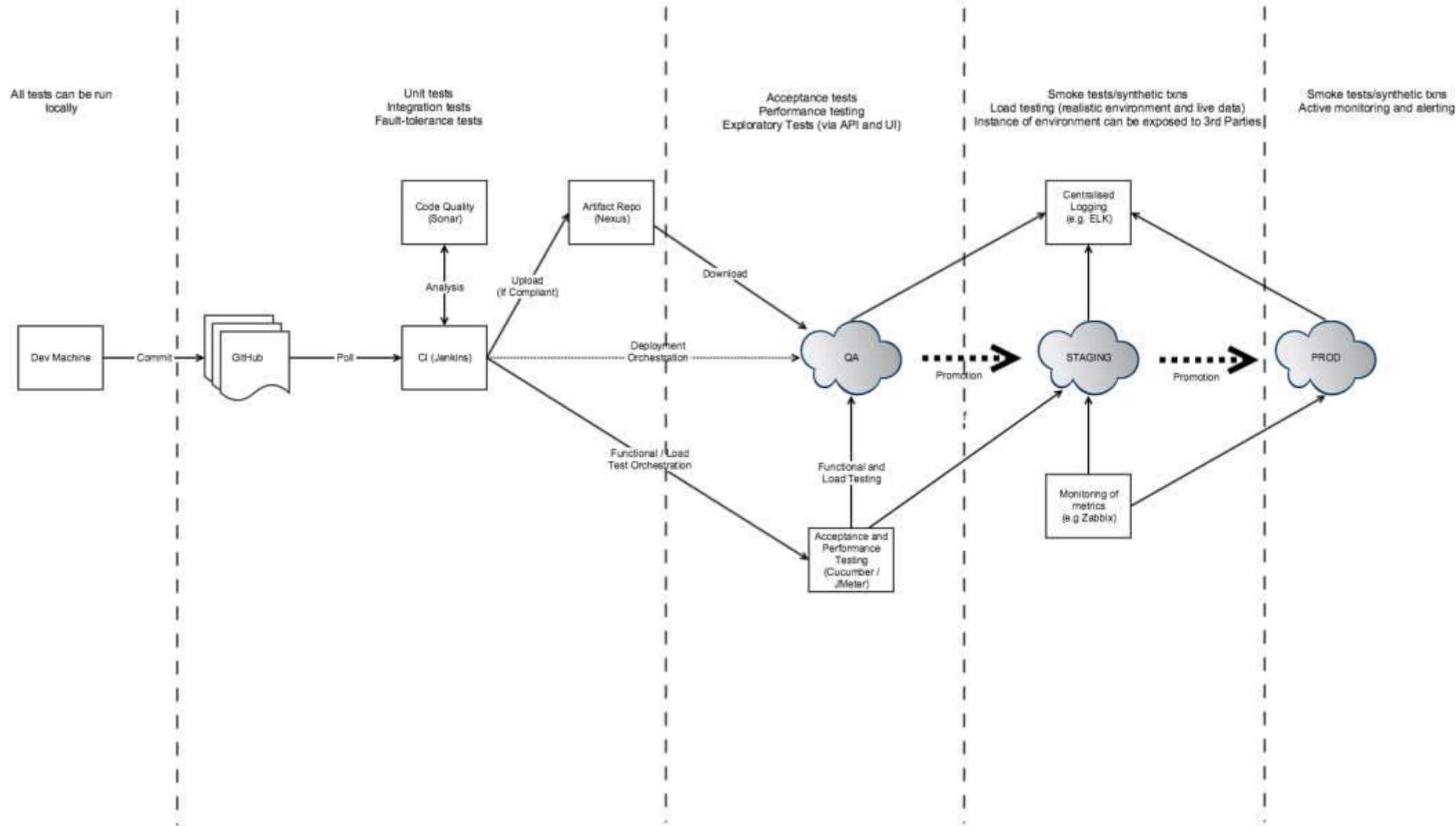
“Why is the container image 1GB?
It’s a helloworld Java app!!!”

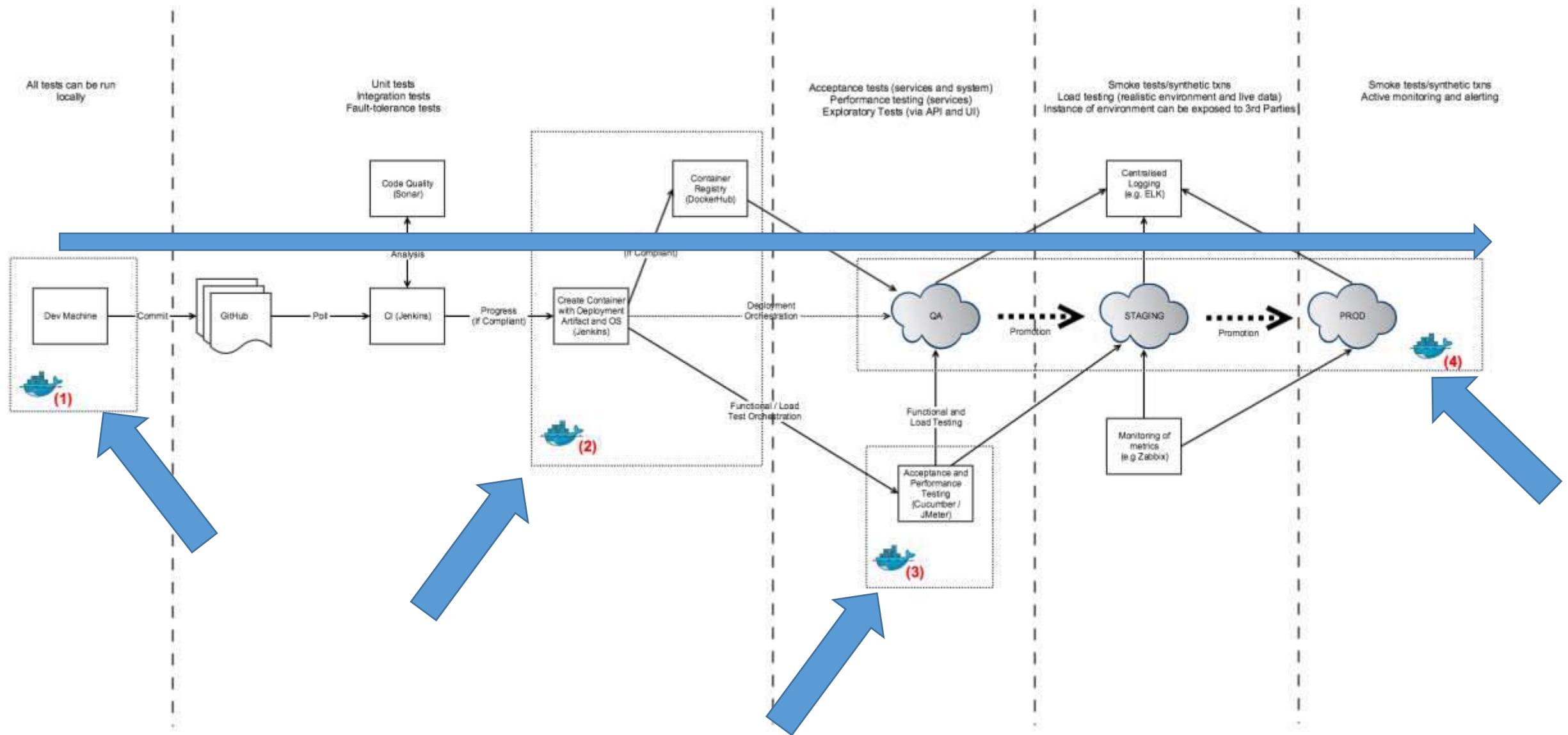
“Dinosaurs must have completed their
dev/test/deploy loop faster than me”

“This Java app runs slow (or freezes)
in Docker”

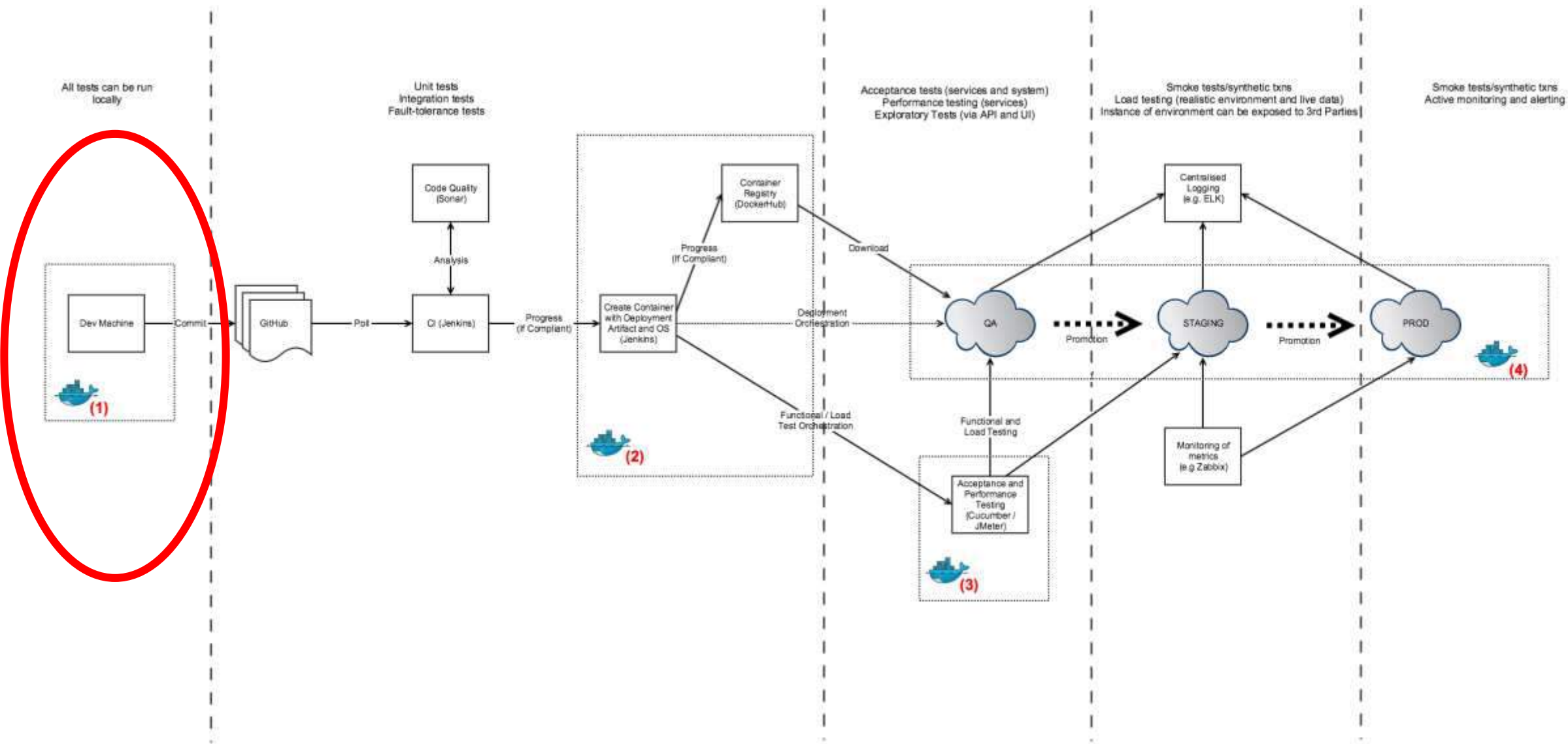


Impact of container tech on CD





Lessons learned



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
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



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

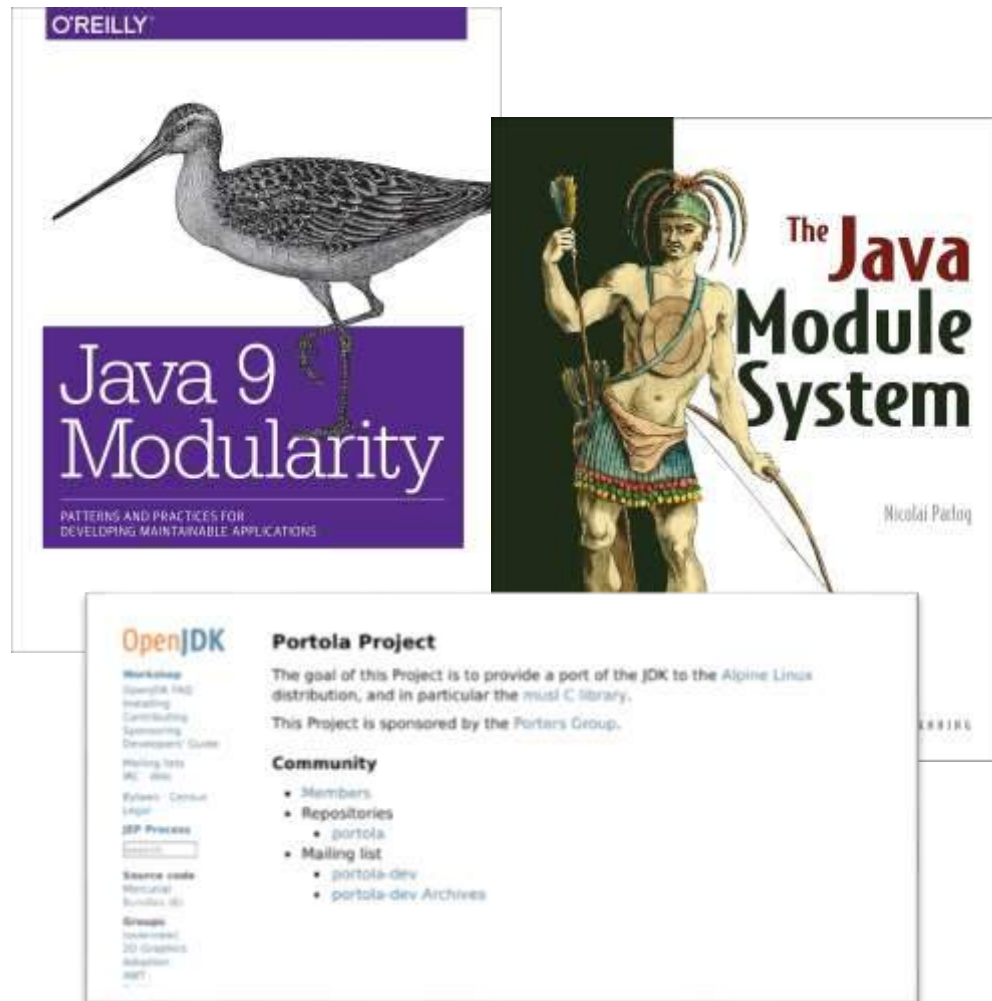
Start from good foundations: base image

```
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
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?


```

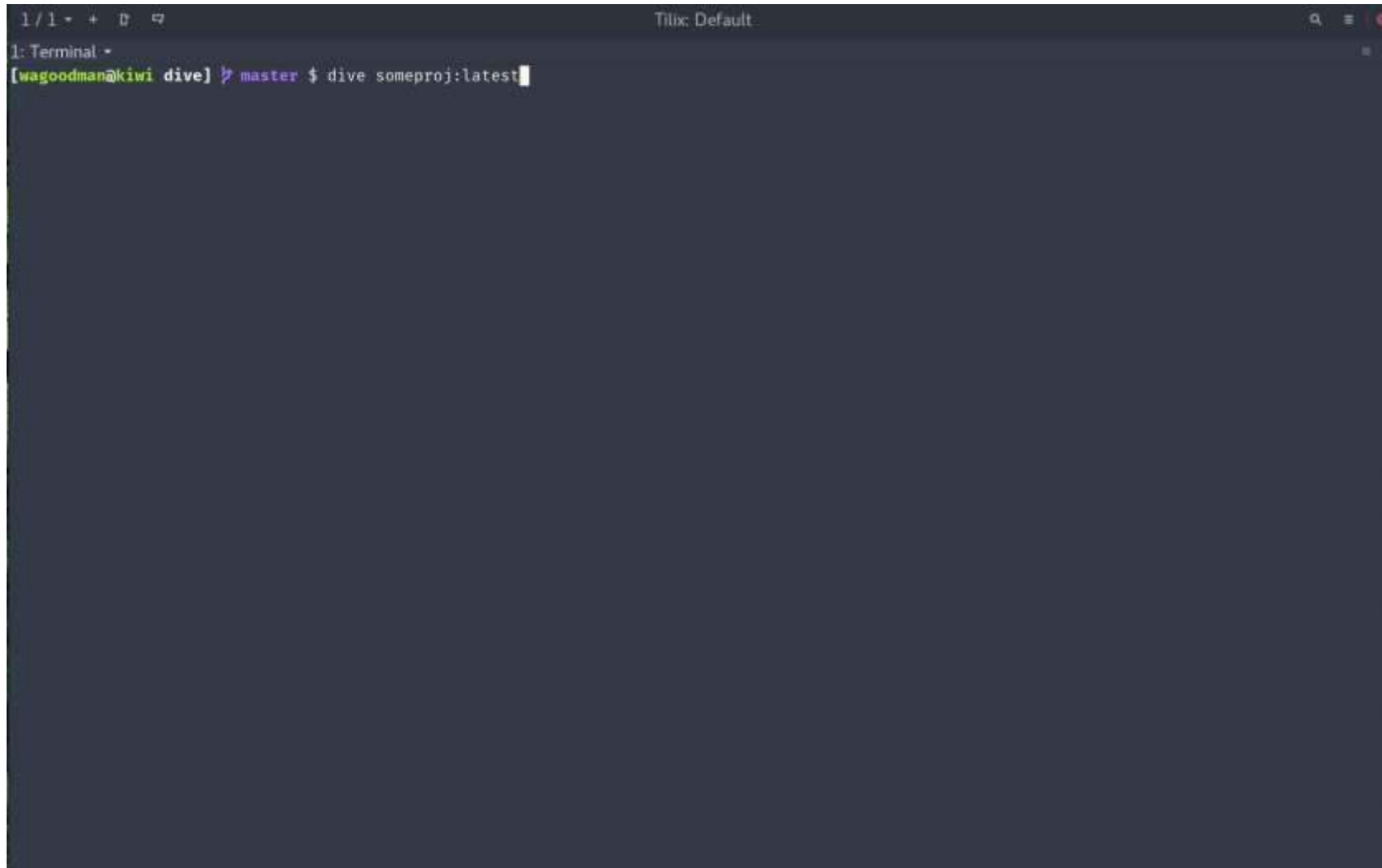
(master *) stockmanager $ mvn dependency:tree
[INFO] Scanning for projects...
[INFO]
[INFO] -----< uk.co.danielbryant.djshopping:stockmanager >-----
[INFO] Building stockmanager 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-dependency-plugin:2.10:tree (default-cli) @ stockmanager ---
[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

A screenshot of a terminal window titled "Tilix: Default". The terminal shows a prompt for a user named "wagoodman" on a host named "kiwi". The user is in a "dive" session. The command "dive someproj:latest" has been entered, and the cursor is at the end of the line. The terminal background is dark blue with light blue text.

```
1/1 + [ ] [ ]  
Tilix: Default  
1: Terminal -  
[wagoodman@kiwi dive] master $ dive someproj:latest
```

<https://github.com/wagoodman/dive>

@danielbryantuk

05/12/2018

BigPictureTech

...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: sha256:0f19a3bf0af3caa2065b4c13a71cfb0c044825750c717ce47291c798ae658a86

Tar ID: ef391ae14085888645bce38ad2725964c9e1f072e0ed1a04cedd153a2e6a4d31

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-journal
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

...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:a6766cf937d8d60df031fbbfb4d97e0ca16fe1324f2e2ce5e640d0eb22342130

Tar ID: 4697d9fb77558f2c820f9b027796ca0924689195496cd1be13c095dd80e89789

Command:
/bin/sh -c set -eux; curl -fL -o /openjdk.tgz "\$JAVA_URL"; echo "\$JAVA_SHA256 */openjdk.tgz" | sha256sum -c -; mkdir -p "\$JAVA_HOME"; tar --extract --file /openjdk.tgz --directory "\$JAVA_HOME" --strip-components 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 -Xshare: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/Shalheader
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/o
-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/o
-rwxrwxrwx	0:0	0 B	jdeps → /usr/java/openjdk
-rwxrwxrwx	0:0	0 B	jhsdb → /usr/java/openjdk

Building in containers (multi-stage FTW)

```
1 FROM maven:3.6.0-jdk-8-alpine AS BUILD
2 COPY src /usr/src/myapp/src
3 COPY pom.xml /usr/src/myapp
4 RUN mvn -f /usr/src/myapp/pom.xml clean package
5
6 FROM openjdk:8u181-jre-alpine3.8
7 COPY --from=BUILD /usr/src/myapp/target/stockmanager-0.0.1-SNAPSHOT.jar app.jar
8 EXPOSE 8030
9 ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/app.jar"]
```


BuildKit

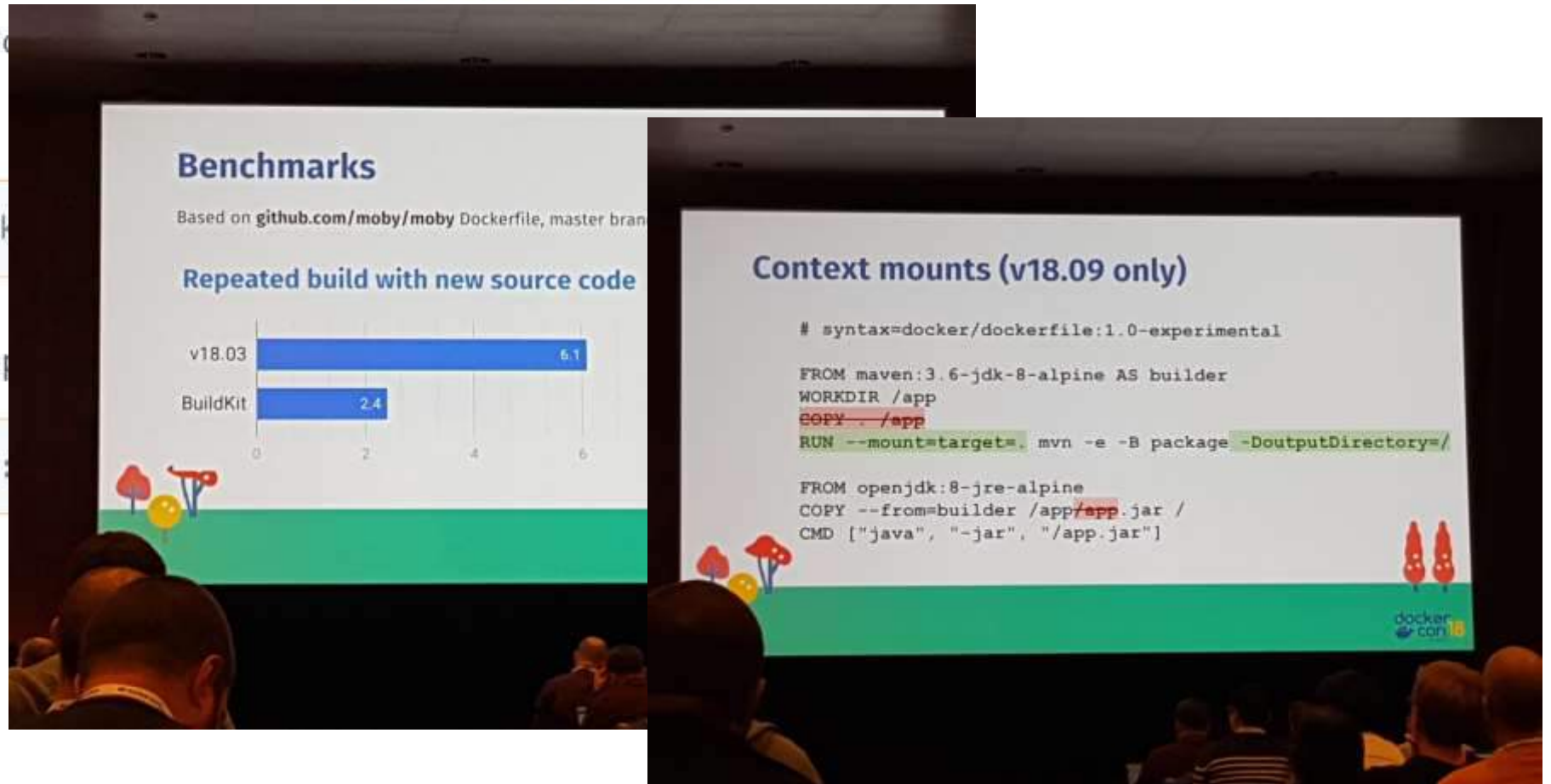
- **BuildKit is now generally available** – Access [improved build performance](#) (see slides 22-26)

and scalability with the option of setting the environment variable, e.g.

```
$ DOCKER_BUILDKIT=1 docker build
```

You can also set the feature option in the Dockerfile:

```
{"features":{"buildkit":true}}
```



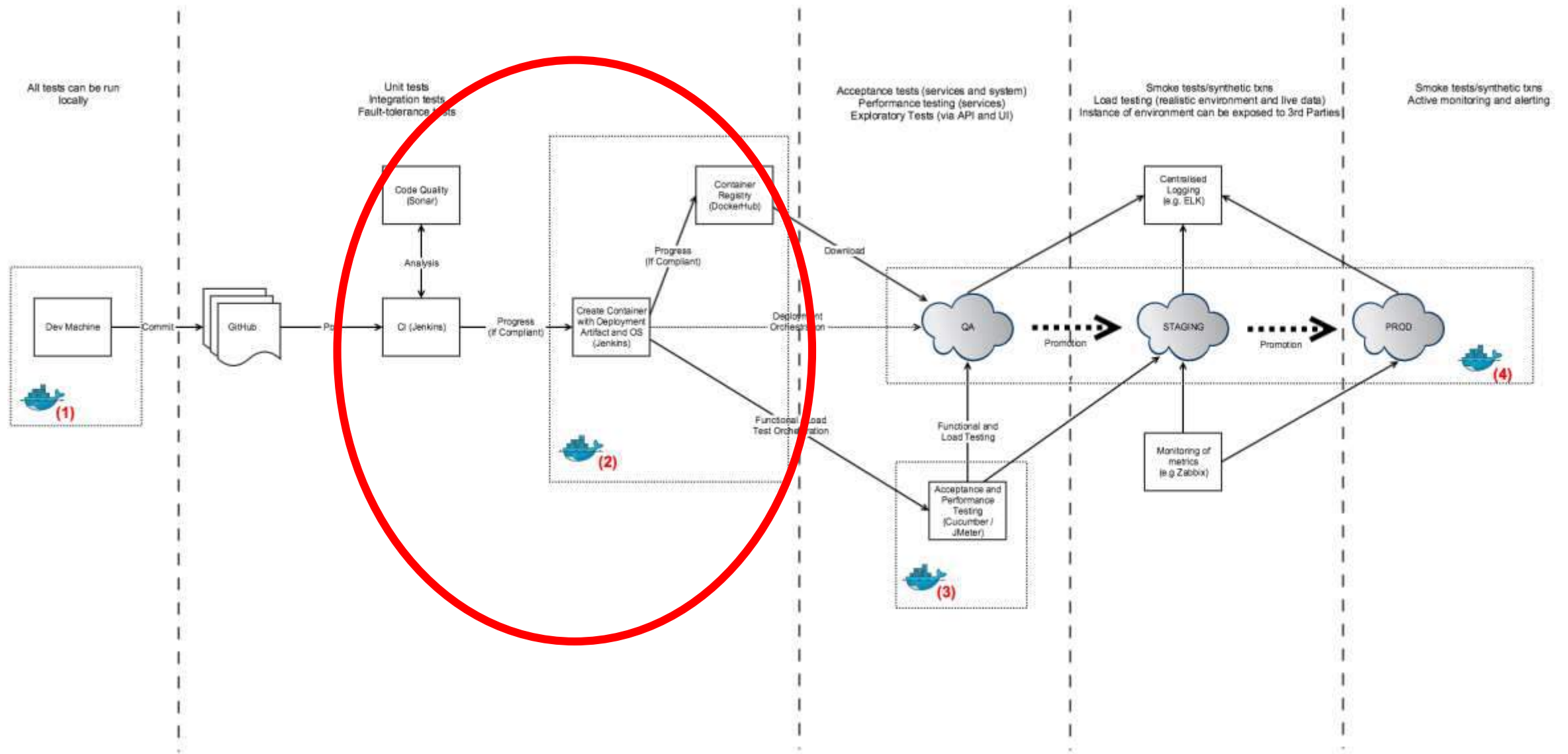
The bad: different test and prod containers?

- 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?

```
1 FROM maven:3.6.0-jdk-8-alpine AS BUILD
2 COPY src /usr/src/myapp/src
3 COPY pom.xml /usr/src/myapp
4 RUN mvn -f /usr/src/myapp/pom.xml clean package
5
6 FROM openjdk:8u181-jre-alpine3.8
7 COPY --from=BUILD /usr/src/myapp/target/stockmanager-0.0.1-SNAPSHOT.jar app.jar
8 # Install test tools here
9 # Run test tools
10
11 FROM openjdk:8u181-jdk-alpine3.8
12 COPY --from=BUILD /usr/src/myapp/target/stockmanager-0.0.1-SNAPSHOT.jar app.jar
13 EXPOSE 8030
14 ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/./urandom", "-jar", "/app.jar"]
```

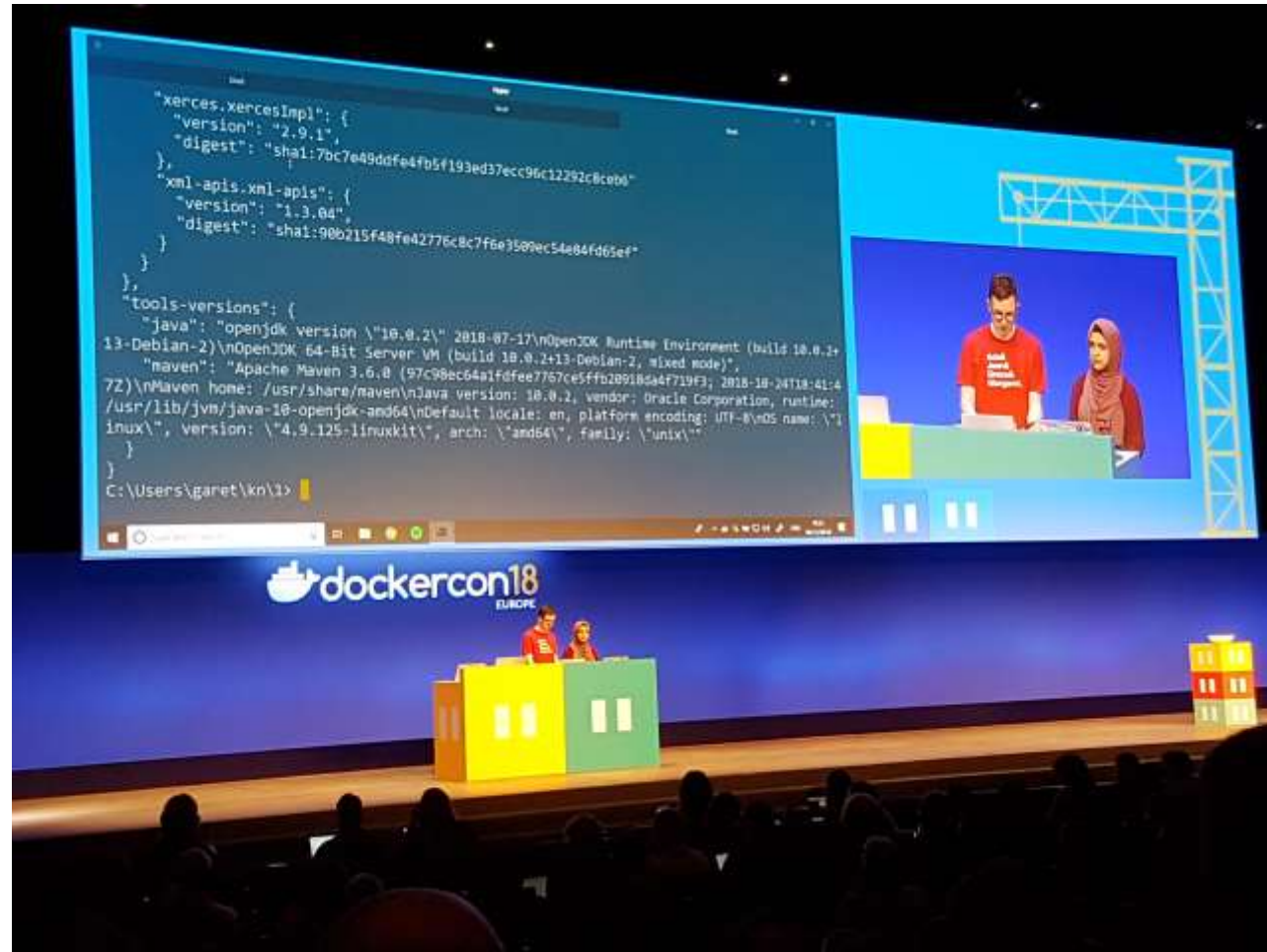


Lesson learned: Metadata is valuable

- 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

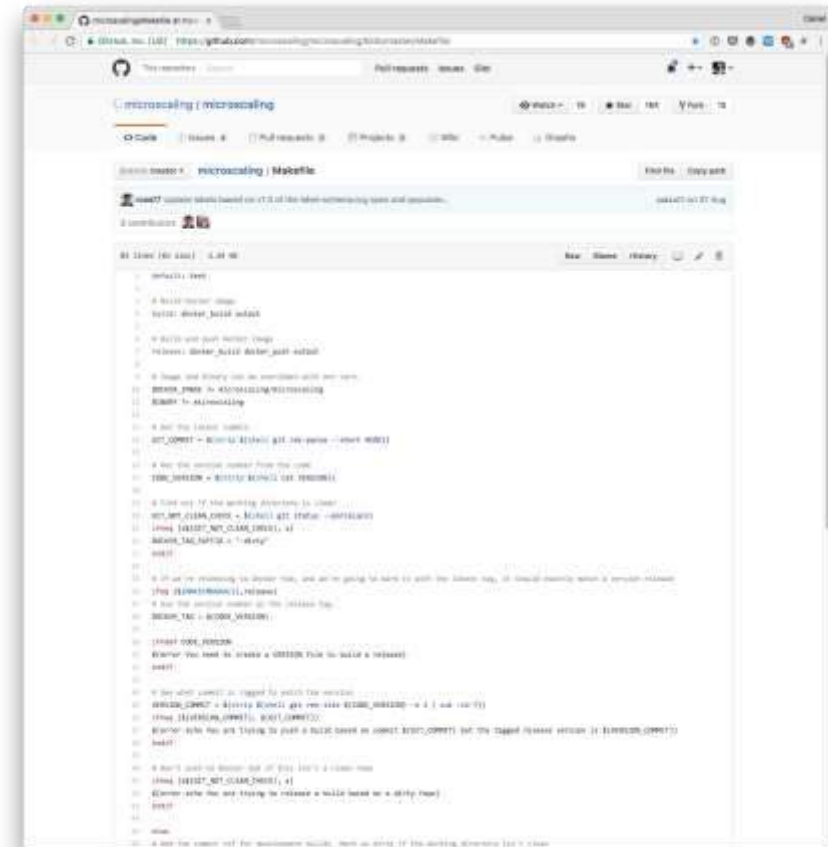


Gareth and Amn totally stole my thunder... :)



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](#)



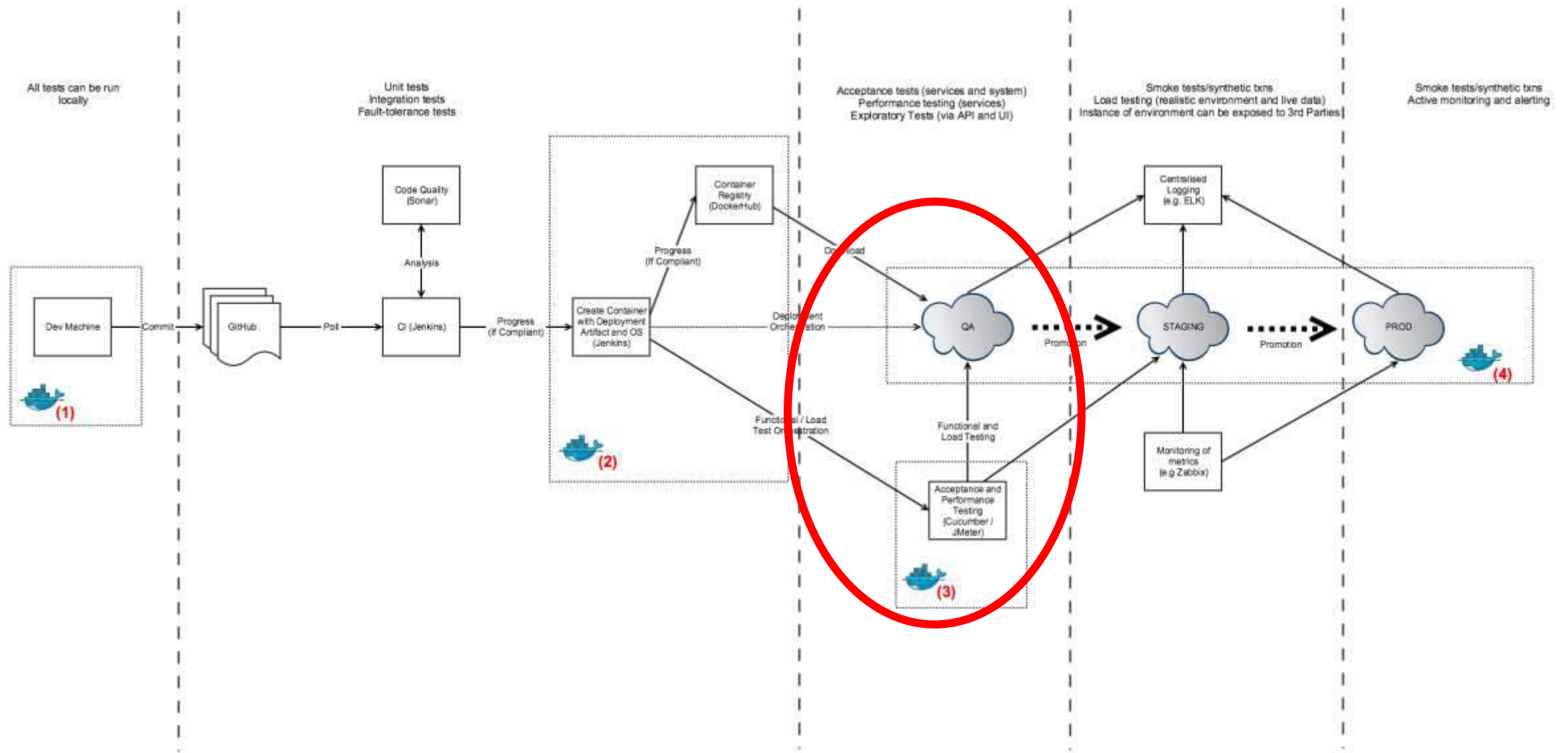
External registry with metadata support

JFrog Artifactory + Docker



 **NexusOSS**
your private docker registry

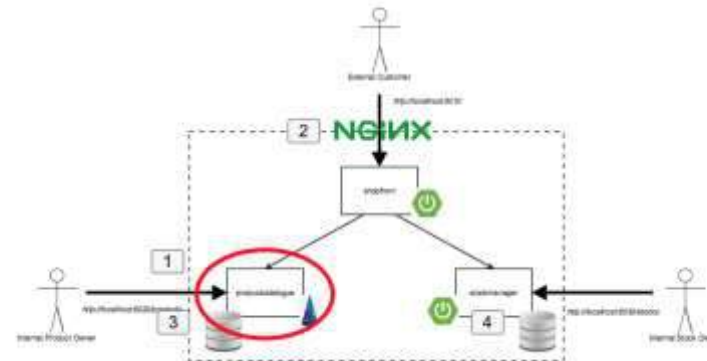




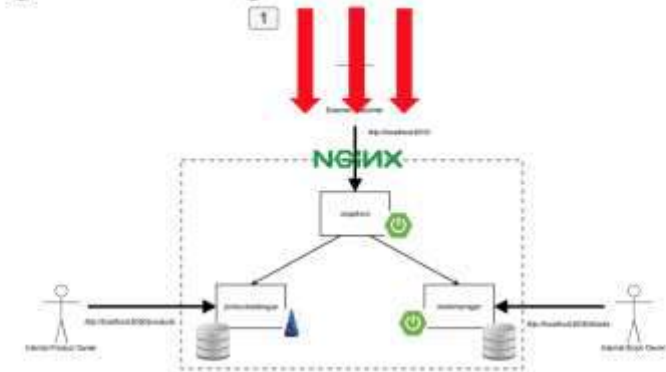
Running tests with containers

```
1 version: '2'
2 services:
3   shopfront:
4     build: shopfront
5     image: danielbryantuk/djshopfront
6     ports:
7       - "8010:8010"
8     links:
9       - productcatalogue
10      - stockmanager
11   productcatalogue:
12     build: productcatalogue
13     image: danielbryantuk/djproductcatalogue
14     ports:
15       - "8020:8020"
16   stockmanager:
17     build: stockmanager
18     image: danielbryantuk/djstockmanager
19     ports:
20       - "8030:8030"
```

Component testing

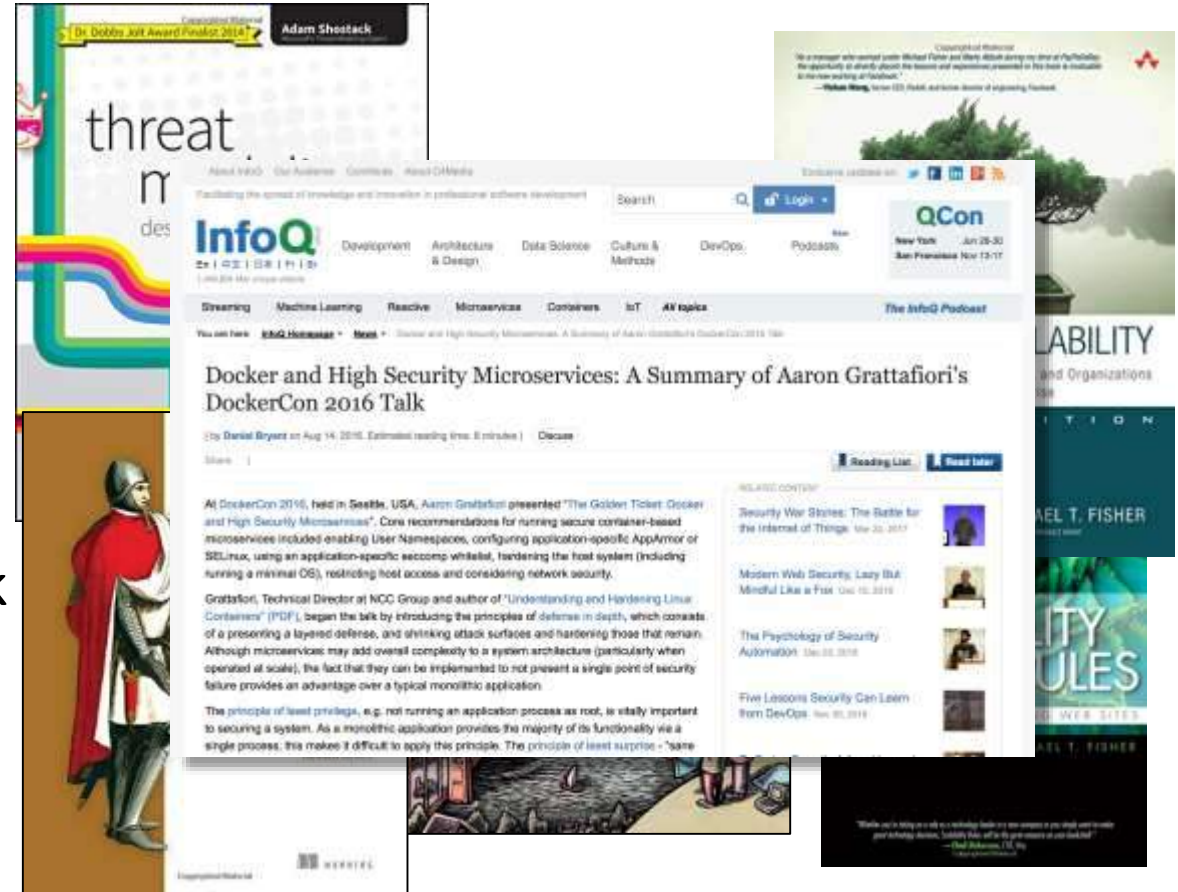


Integration testing



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*

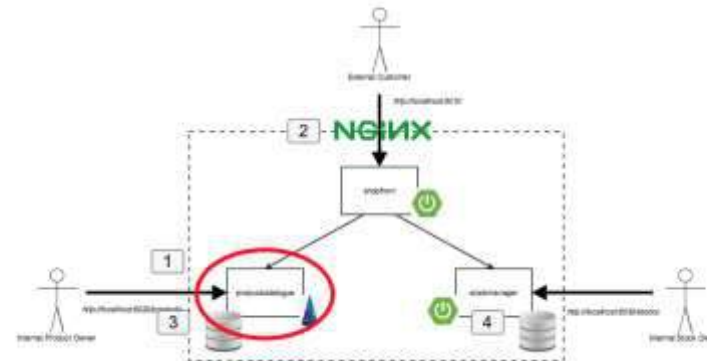
JDK-8197867 Update CPU count algorithm when both cpu shares and quotas are used

etc...

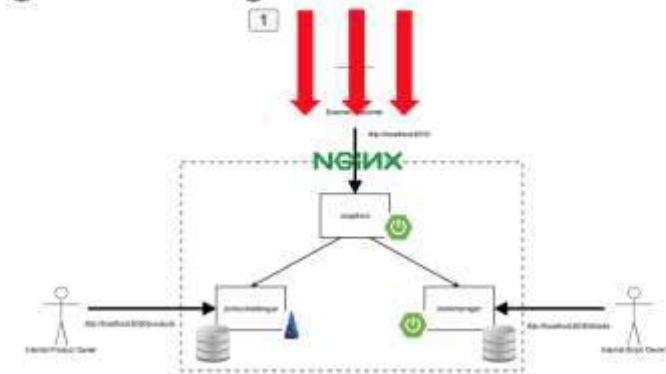
Running tests with containers

```
1 version: '2'
2 services:
3   shopfront:
4     build: shopfront
5     image: danielbryantuk/djshopfront
6     ports:
7       - "8010:8010"
8     links:
9       - productcatalogue
10      - stockmanager
11   productcatalogue:
12     build: productcatalogue
13     image: danielbryantuk/djproductcatalogue
14     ports:
15       - "8020:8020"
16   stockmanager:
17     build: stockmanager
18     image: danielbryantuk/djstockmanager
19     ports:
20       - "8030:8030"
```

Component testing



Integration testing



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);
        }
    }
}
```

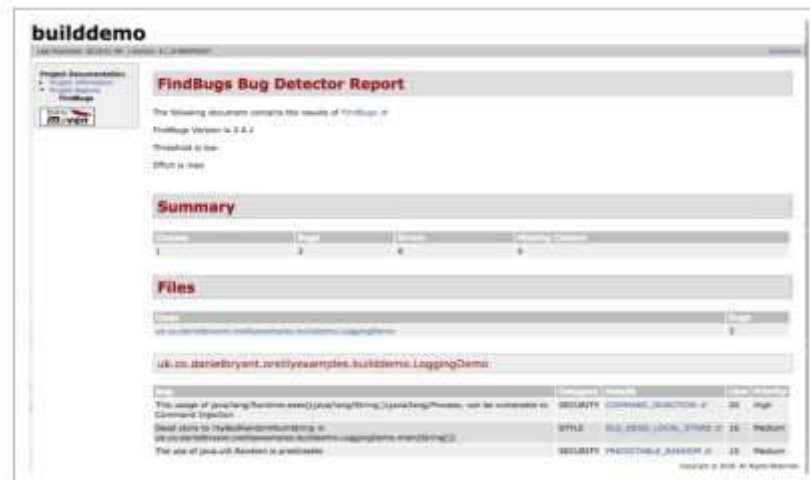
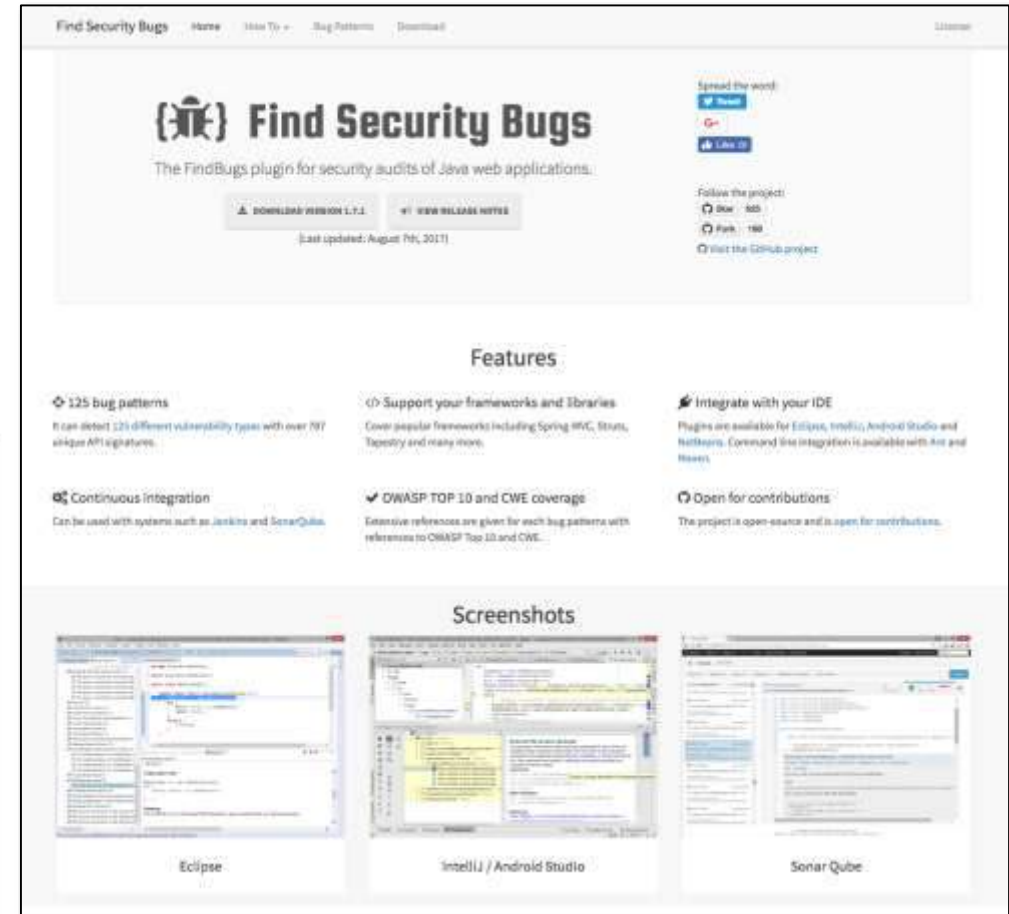


Figure 11-1. Output from the Maven FindBugs plugin with FindSecBugs enabled



Dependency Scanning

www.owasp.org/index.php/OWASP_Dependency_Check

[illegible]

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 experimental state. Please use the tags for production-ready binaries.




Clair is an open source container image scanner powered by [Docker](#) and [Docker Hub](#).

1. In regular intervals, Clair scans the Docker Hub database.
2. Clients use the Clair API to scan their images and store them in the database.
3. Clients use the Clair API to scan their images and compare the results against the database.
4. When updates to the database occur, Clair will re-scan the images.

Our goal is to enable you to scan your images and named clair after the scan.

Getting Started


- Learn the [terminology](#)
- Watch [presentations](#)



Armin C.
@acoralic

@danielbryantuk

talk, a sim with Clair



11:39 AM - 29 Sep

1 Like

Clair scanner

downloaded yes build passing go report A+ coverage 100%

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 on

This straightforward process is not that easy to achieve when using the services like Docker Hub because they work asynchronously which makes it harder to do straightforward CI/CD pipeline.

Clair to the rescue

CoreOS has created an awesome container scan tool called Clair. Clair is also used by Docker Hub. It has a simple tool that scans your image and compares the vulnerabilities against a whitelist of known vulnerabilities.

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 a server, therefore, I have created a way to run this standalone. See here <https://github.com/arminc/clair-scanner>

Credits

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

```
(master) clair-scanner $ docker pull openjdk:8-jre
openjdk:8-jre: Pulling from library/openjdk
3e17c0e0e66c: Pull complete
74d44b20f851: Pull complete
d156217f3f04: Pull complete
8bce9f57693: Pull complete
3f9410abd9cf: Pull complete
c5e2fb2d8cf: Pull complete
c9681f71497b: Pull complete
c20de950e52: Pull complete
Digest: sha256:d2bc505026bddf120e5e459468ef12de447b00c29f9090763c329d00536d
Status: Downloaded newer image for openjdk:8-jre
 clair-scanner $ ./clair-scanner --ip 192.168.0.7 openjdk:8-jre
2017/10/23 10:50:50 [INFO] > Start Clair scanner
2017/10/23 10:51:00 [INFO] > Server is running on port 9279
2017/10/23 10:51:00 [INFO] > Analyzing 69357d9443c362194d1c6640c321c7c57f69e444d0e165d810a7057f6d081
2017/10/23 10:51:03 [INFO] > Analyzing 2f8fcb54a2bd4930d7038f504f08974d460d4e22073837f7c6b386942d038e
2017/10/23 10:51:03 [INFO] > Analyzing 505a276d97295828e2660d7210b645f50f60306d5e32b0d7f91f03042300dd
2017/10/23 10:51:04 [INFO] > Analyzing f4ef0b482008fe0b738332070c03ee445c4008903214b14e30e01600d0
2017/10/23 10:51:04 [INFO] > Analyzing 33a0d0d1f0d1a0701b3a0f17c0533e27c0b9855a0e738748d11e4d0e30e
2017/10/23 10:51:04 [INFO] > Analyzing 1deb9d8d089116680bd4972867432265bca5f77dd68c1e867e74b8f0ef077
2017/10/23 10:51:04 [INFO] > Analyzing 423a0d08070d52de5af63fbc1f43e7704e1453ddfb64400cab0dd02e9
2017/10/23 10:51:14 [INFO] > Unapproved vulnerabilities: [CVE-2017-9937 CVE-2011-3389 CVE-2017-11333 CVE-2017-11735 CVE-2017-9854 CVE-2017-11331 CVE-2017-11332 CVE-2017-12944 CVE-2014-8139 CVE-2017-13727 CVE-2017-4995 CVE-2017-13726 CVE-2017-9117 CVE-2017-11613 CVE-2017-7246 CVE-2017-11614 CVE-2017-11615 CVE-2017-11616 CVE-2017-11617 CVE-2017-11618 CVE-2017-11619 CVE-2017-11620 CVE-2017-11621 CVE-2017-11622 CVE-2017-11623 CVE-2017-11624 CVE-2017-11625 CVE-2017-11626 CVE-2017-11627 CVE-2017-11628 CVE-2017-11629 CVE-2017-11630 CVE-2017-11631 CVE-2017-11632 CVE-2017-11633 CVE-2017-11634 CVE-2017-11635 CVE-2017-11636 CVE-2017-11637 CVE-2017-11638 CVE-2017-11639 CVE-2017-11640 CVE-2017-11641 CVE-2017-11642 CVE-2017-11643 CVE-2017-11644 CVE-2017-11645 CVE-2017-11646 CVE-2017-11647 CVE-2017-11648 CVE-2017-11649 CVE-2017-11650 CVE-2017-11651 CVE-2017-11652 CVE-2017-11653 CVE-2017-11654 CVE-2017-11655 CVE-2017-11656 CVE-2017-11657 CVE-2017-11658 CVE-2017-11659 CVE-2017-11660 CVE-2017-11661 CVE-2017-11662 CVE-2017-11663 CVE-2017-11664 CVE-2017-11665 CVE-2017-11666 CVE-2017-11667 CVE-2017-11668 CVE-2017-11669 CVE-2017-11670 CVE-2017-11671 CVE-2017-11672 CVE-2017-11673 CVE-2017-11674 CVE-2017-11675 CVE-2017-11676 CVE-2017-11677 CVE-2017-11678 CVE-2017-11679 CVE-2017-11680 CVE-2017-11681 CVE-2017-11682 CVE-2017-11683 CVE-2017-11684 CVE-2017-11685 CVE-2017-11686 CVE-2017-11687 CVE-2017-11688 CVE-2017-11689 CVE-2017-11690 CVE-2017-11691 CVE-2017-11692 CVE-2017-11693 CVE-2017-11694 CVE-2017-11695 CVE-2017-11696 CVE-2017-11697 CVE-2017-11698 CVE-2017-11699 CVE-2017-11700 CVE-2017-11701 CVE-2017-11702 CVE-2017-11703 CVE-2017-11704 CVE-2017-11705 CVE-2017-11706 CVE-2017-11707 CVE-2017-11708 CVE-2017-11709 CVE-2017-11710 CVE-2017-11711 CVE-2017-11712 CVE-2017-11713 CVE-2017-11714 CVE-2017-11715 CVE-2017-11716 CVE-2017-11717 CVE-2017-11718 CVE-2017-11719 CVE-2017-11720 CVE-2017-11721 CVE-2017-11722 CVE-2017-11723 CVE-2017-11724 CVE-2017-11725 CVE-2017-11726 CVE-2017-11727 CVE-2017-11728 CVE-2017-11729 CVE-2017-11730 CVE-2017-11731 CVE-2017-11732 CVE-2017-11733 CVE-2017-11734 CVE-2017-11735 CVE-2017-11736 CVE-2017-11737 CVE-2017-11738 CVE-2017-11739 CVE-2017-11740 CVE-2017-11741 CVE-2017-11742 CVE-2017-11743 CVE-2017-11744 CVE-2017-11745 CVE-2017-11746 CVE-2017-11747 CVE-2017-11748 CVE-2017-11749 CVE-2017-11750 CVE-2017-11751 CVE-2017-11752 CVE-2017-11753 CVE-2017-11754 CVE-2017-11755 CVE-2017-11756 CVE-2017-11757 CVE-2017-11758 CVE-2017-11759 CVE-2017-11760 CVE-2017-11761 CVE-2017-11762 CVE-2017-11763 CVE-2017-11764 CVE-2017-11765 CVE-2017-11766 CVE-2017-11767 CVE-2017-11768 CVE-2017-11769 CVE-2017-11770 CVE-2017-11771 CVE-2017-11772 CVE-2017-11773 CVE-2017-11774 CVE-2017-11775 CVE-2017-11776 CVE-2017-11777 CVE-2017-11778 CVE-2017-11779 CVE-2017-11780 CVE-2017-11781 CVE-2017-11782 CVE-2017-11783 CVE-2017-11784 CVE-2017-11785 CVE-2017-11786 CVE-2017-11787 CVE-2017-11788 CVE-2017-11789 CVE-2017-11790 CVE-2017-11791 CVE-2017-11792 CVE-2017-11793 CVE-2017-11794 CVE-2017-11795 CVE-2017-11796 CVE-2017-11797 CVE-2017-11798 CVE-2017-11799 CVE-2017-11800 CVE-2017-11801 CVE-2017-11802 CVE-2017-11803 CVE-2017-11804 CVE-2017-11805 CVE-2017-11806 CVE-2017-11807 CVE-2017-11808 CVE-2017-11809 CVE-2017-11810 CVE-2017-11811 CVE-2017-11812 CVE-2017-11813 CVE-2017-11814 CVE-2017-11815 CVE-2017-11816 CVE-2017-11817 CVE-2017-11818 CVE-2017-11819 CVE-2017-11820 CVE-2017-11821 CVE-2017-11822 CVE-2017-11823 CVE-2017-11824 CVE-2017-11825 CVE-2017-11826 CVE-2017-11827 CVE-2017-11828 CVE-2017-11829 CVE-2017-11830 CVE-2017-11831 CVE-2017-11832 CVE-2017-11833 CVE-2017-11834 CVE-2017-11835 CVE-2017-11836 CVE-2017-11837 CVE-2017-11838 CVE-2017-11839 CVE-2017-11840 CVE-2017-11841 CVE-2017-11842 CVE-2017-11843 CVE-2017-11844 CVE-2017-11845 CVE-2017-11846 CVE-2017-11847 CVE-2017-11848 CVE-2017-11849 CVE-2017-11850 CVE-2017-11851 CVE-2017-11852 CVE-2017-11853 CVE-2017-11854 CVE-2017-11855 CVE-2017-11856 CVE-2017-11857 CVE-2017-11858 CVE-2017-11859 CVE-2017-11860 CVE-2017-11861 CVE-2017-11862 CVE-2017-11863 CVE-2017-11864 CVE-2017-11865 CVE-2017-11866 CVE-2017-11867 CVE-2017-11868 CVE-2017-11869 CVE-2017-11870 CVE-2017-11871 CVE-2017-11872 CVE-2017-11873 CVE-2017-11874 CVE-2017-11875 CVE-2017-11876 CVE-2017-11877 CVE-2017-11878 CVE-2017-11879 CVE-2017-11880 CVE-2017-11881 CVE-2017-11882 CVE-2017-11883 CVE-2017-11884 CVE-2017-11885 CVE-2017-11886 CVE-2017-11887 CVE-2017-11888 CVE-2017-11889 CVE-2017-11890 CVE-2017-11891 CVE-2017-11892 CVE-2017-11893 CVE-2017-11894 CVE-2017-11895 CVE-2017-11896 CVE-2017-11897 CVE-2017-11898 CVE-2017-11899 CVE-2017-11900 CVE-2017-11901 CVE-2017-11902 CVE-2017-11903 CVE-2017-11904 CVE-2017-11905 CVE-2017-11906 CVE-2017-11907 CVE-2017-11908 CVE-2017-11909 CVE-2017-11910 CVE-2017-11911 CVE-2017-11912 CVE-2017-11913 CVE-2017-11914 CVE-2017-11915 CVE-2017-11916 CVE-2017-11917 CVE-2017-11918 CVE-2017-11919 CVE-2017-11920 CVE-2017-11921 CVE-2017-11922 CVE-2017-11923 CVE-2017-11924 CVE-2017-11925 CVE-2017-11926 CVE-2017-11927 CVE-2017-11928 CVE-2017-11929 CVE-2017-11930 CVE-2017-11931 CVE-2017-11932 CVE-2017-11933 CVE-2017-11934 CVE-2017-11935 CVE-2017-11936 CVE-2017-11937 CVE-2017-11938 CVE-2017-11939 CVE-2017-11940 CVE-2017-11941 CVE-2017-11942 CVE-2017-11943 CVE-2017-11944 CVE-2017-11945 CVE-2017-11946 CVE-2017-11947 CVE-2017-11948 CVE-2017-11949 CVE-2017-11950 CVE-2017-11951 CVE-2017-11952 CVE-2017-11953 CVE-2017-11954 CVE-2017-11955 CVE-2017-11956 CVE-2017-11957 CVE-2017-11958 CVE-2017-11959 CVE-2017-11960 CVE-2017-11961 CVE-2017-11962 CVE-2017-11963 CVE-2017-11964 CVE-2017-11965 CVE-2017-11966 CVE-2017-11967 CVE-2017-11968 CVE-2017-11969 CVE-2017-11970 CVE-2017-11971 CVE-2017-11972 CVE-2017-11973 CVE-2017-11974 CVE-2017-11975 CVE-2017-11976 CVE-2017-11977 CVE-2017-11978 CVE-2017-11979 CVE-2017-11980 CVE-2017-11981 CVE-2017-11982 CVE-2017-11983 CVE-2017-11984 CVE-2017-11985 CVE-2017-11986 CVE-2017-11987 CVE-2017-11988 CVE-2017-11989 CVE-2017-11990 CVE-2017-11991 CVE-2017-11992 CVE-2017-11993 CVE-2017-11994 CVE-2017-11995 CVE-2017-11996 CVE-2017-11997 CVE-2017-11998 CVE-2017-11999 CVE-2017-12000 CVE-2017-12001 CVE-2017-12002 CVE-2017-12003 CVE-2017-12004 CVE-2017-12005 CVE-2017-12006 CVE-2017-12007 CVE-2017-12008 CVE-2017-12009 CVE-2017-12010 CVE-2017-12011 CVE-2017-12012 CVE-2017-12013 CVE-2017-12014 CVE-2017-12015 CVE-2017-12016 CVE-2017-12017 CVE-2017-12018 CVE-2017-12019 CVE-2017-12020 CVE-2017-12021 CVE-2017-12022 CVE-2017-12023 CVE-2017-12024 CVE-2017-12025 CVE-2017-12026 CVE-2017-12027 CVE-2017-12028 CVE-2017-12029 CVE-2017-12030 CVE-2017-12031 CVE-2017-12032 CVE-2017-12033 CVE-2017-12034 CVE-2017-12035 CVE-2017-12036 CVE-2017-12037 CVE-2017-12038 CVE-2017-12039 CVE-2017-12040 CVE-2017-12041 CVE-2017-12042 CVE-2017-12043 CVE-2017-12044 CVE-2017-12045 CVE-2017-12046 CVE-2017-12047 CVE-2017-12048 CVE-2017-12049 CVE-2017-12050 CVE-2017-12051 CVE-2017-12052 CVE-2017-12053 CVE-2017-12054 CVE-2017-12055 CVE-2017-12056 CVE-2017-12057 CVE-2017-12058 CVE-2017-12059 CVE-2017-12060 CVE-2017-12061 CVE-2017-12062 CVE-2017-12063 CVE-2017-12064 CVE-2017-12065 CVE-2017-12066 CVE-2017-12067 CVE-2017-12068 CVE-2017-12069 CVE-2017-12070 CVE-2017-12071 CVE-2017-12072 CVE-2017-12073 CVE-2017-12074 CVE-2017-12075 CVE-2017-12076 CVE-2017-12077 CVE-2017-12078 CVE-2017-12079 CVE-2017-12080 CVE-2017-12081 CVE-2017-12082 CVE-2017-12083 CVE-2017-12084 CVE-2017-12085 CVE-2017-12086 CVE-2017-12087 CVE-2017-12088 CVE-2017-12089 CVE-2017-12090 CVE-2017-12091 CVE-2017-12092 CVE-2017-12093 CVE-2017-12094 CVE-2017-12095 CVE-2017-12096 CVE-2017-12097 CVE-2017-12098 CVE-2017-12099 CVE-2017-12100 CVE-2017-12101 CVE-2017-12102 CVE-2017-12103 CVE-2017-12104 CVE-2017-12105 CVE-2017-12106 CVE-2017-12107 CVE-2017-12108 CVE-2017-12109 CVE-2017-12110 CVE-2017-12111 CVE-2017-12112 CVE-2017-12113 CVE-2017-12114 CVE-2017-12115 CVE-2017-12116 CVE-2017-12117 CVE-2017-12118 CVE-2017-12119 CVE-2017-12120 CVE-2017-12121 CVE-2017-12122 CVE-2017-12123 CVE-2017-12124 CVE-2017-12125 CVE-2017-12126 CVE-2017-12127 CVE-2017-12128 CVE-2017-12129 CVE-2017-12130 CVE-2017-12131 CVE-2017-12132 CVE-2017-12133 CVE-2017-12134 CVE-2017-12135 CVE-2017-12136 CVE-2017-12137 CVE-2017-12138 CVE-2017-12139 CVE-2017-12140 CVE-2017-12141 CVE-2017-12142 CVE-2017-12143 CVE-2017-12144 CVE-2017-12145 CVE-2017-12146 CVE-2017-12147 CVE-2017-12148 CVE-2017-12149 CVE-2017-12150 CVE-2017-12151 CVE-2017-12152 CVE-2017-12153 CVE-2017-12154 CVE-2017-12155 CVE-2017-12156 CVE-2017-12157 CVE-2017-12158 CVE-2017-12159 CVE-2017-12160 CVE-2017-12161 CVE-2017-12162 CVE-2017-12163 CVE-2017-12164 CVE-2017-12165 CVE-2017-12166 CVE-2017-12167 CVE-2017-12168 CVE-2017-12169 CVE-2017-12170 CVE-2017-12171 CVE-2017-12172 CVE-2017-12173 CVE-2017-12174 CVE-2017-12175 CVE-2017-12176 CVE-2017-12177 CVE-2017-12178 CVE-2017-12179 CVE-2017-12180 CVE-2017-12181 CVE-2017-12182 CVE-2017-12183 CVE-2017-12184 CVE-2017-12185 CVE-2017-12186 CVE-2017-12187 CVE-2017-12188 CVE-2017-12189 CVE-2017-12190 CVE-2017-12191 CVE-2017-12192 CVE-2017-12193 CVE-2017-12194 CVE-2017-12195 CVE-2017-12196 CVE-2017-12197 CVE-2017-12198 CVE-2017-12199 CVE-2017-12200 CVE-2017-12201 CVE-2017-12202 CVE-2017-12203 CVE-2017-12204 CVE-2017-12205 CVE-2017-12206 CVE-2017-12207 CVE-2017-12208 CVE-2017-12209 CVE-2017-12210 CVE-2017-12211 CVE-2017-12212 CVE-2017-12213 CVE-2017-12214 CVE-2017-12215 CVE-2017-12216 CVE-2017-12217 CVE-2017-12218 CVE-2017-12219 CVE-2017-12220 CVE-2017-12221 CVE-2017-12222 CVE-2017-12223 CVE-2017-12224 CVE-2017-12225 CVE-2017-12226 CVE-2017-12227 CVE-2017-12228 CVE-2017-12229 CVE-2017-12230 CVE-2017-12231 CVE-2017-12232 CVE-2017-12233 CVE-2017-12234 CVE-2017-12235 CVE-2017-12236 CVE-2017-12237 CVE-2017-12238 CVE-2017-12239 CVE-2017-12240 CVE-2017-12241 CVE-2017-12242 CVE-2017-12243 CVE-2017-12244 CVE-2017-12245 CVE-2017-12246 CVE-2017-12247 CVE-2017-12248 CVE-2017-12249 CVE-2017-12250 CVE-2017-12251 CVE-2017-12252 CVE-2017-12253 CVE-2017-12254 CVE-2017-12255 CVE-2017-12256 CVE-2017-12257 CVE-2017-12258 CVE-2017-12259 CVE-2017-12260 CVE-2017-12261 CVE-2017-12262 CVE-2017-12263 CVE-2017-12264 CVE-2017-12265 CVE-2017-12266 CVE-2017-12267 CVE-2017-12268 CVE-2017-12269 CVE-2017-12270 CVE-2017-12271 CVE-2017-12272 CVE-2017-12273 CVE-2017-12274 CVE-2017-12275 CVE-2017-12276 CVE-2017-12277 CVE-2017-12278 CVE-2017-12279 CVE-2017-12280 CVE-2017-12281 CVE-2017-12282 CVE-2017-12283 CVE-2017-12284 CVE-2017-12285 CVE-2017-12286 CVE-2017-12287 CVE-2017-12288 CVE-2017-12289 CVE-2017-12290 CVE-2017-12291 CVE-2017-12292 CVE-2017-12293 CVE-2017-12294 CVE-2017-12295 CVE-2017-12296 CVE-2017-12297 CVE-2017-12298 CVE-2017-12299 CVE-2017-12300 CVE-2017-12301 CVE-2017-12302 CVE-2017-12303 CVE-2017-12304 CVE-2017-12305 CVE-2017-12306 CVE-2017-12307 CVE-2017-12308 CVE-2017-12309 CVE-2017-12310 CVE-2017-12311 CVE-2017-12312 CVE-2017-12313 CVE-2017-12314 CVE-2017-12315 CVE-2017-12316 CVE-2017-12317 CVE-2017-12318 CVE-2017-12319 CVE-2017-12320 CVE-2017-12321 CVE-2017-12322 CVE-2017-12323 CVE-2017-12324 CVE-2017-12325 CVE-2017-12326 CVE-2017-12327 CVE-2017-12328 CVE-2017-12329 CVE-2017-12330 CVE-2017-12331 CVE-2017-12332 CVE-2017-12333 CVE-2017-12334 CVE-2017-12335 CVE-2017-12336 CVE-2017-12337 CVE-2017-12338 CVE-2017-12339 CVE-2017-12340 CVE-2017-12341 CVE-2017-12342 CVE-2017-12343 CVE-2017-12344 CVE-2017-12345 CVE-2017-12346 CVE-2017-12347 CVE-2017-12348 CVE-2017-12349 CVE-2017-12350 CVE-2017-12351 CVE-2017-12352 CVE-2017-12353 CVE-2017-12354 CVE-2017-12355 CVE-2017-12356 CVE-2017-12357 CVE-2017-12358 CVE-2017-12359 CVE-2017-12360 CVE-2017-12361 CVE-2017-12362 CVE-2017-12363 CVE-2017-12364 CVE-2017-12365 CVE-2017-12366 CVE-2017-12367 CVE-2017-12368 CVE-2017-12369 CVE-2017-12370 CVE-2017-12371 CVE-2017-12372 CVE-2017-12373 CVE-2017-12374 CVE-2017-12375 CVE-2017-12376 CVE-2017-12377 CVE-2017-12378 CVE-2017-12379 CVE-2017-12380 CVE-2017-12381 CVE-2017-12382 CVE-2017-12383 CVE-2017-12384 CVE-2017-12385 CVE-2017-12386 CVE-2017-12387 CVE-2017-12388 CVE-2017-12389 CVE-2017-12390 CVE-2017-12391 CVE-2017-12392 CVE-2017-12393 CVE-2017-12394 CVE-2017-12395 CVE-2017-12396 CVE-2017-12397 CVE-2017-12398 CVE-2017-12399 CVE-2017-12400 CVE-2017-12401 CVE-2017-12402 CVE-2017-12403 CVE-2017-12404 CVE-2017-12405 CVE-2017-12406 CVE-2017-12407 CVE-2017-12408 CVE-2017-12409 CVE-2017-12410 CVE-2017-12411 CVE-2017-12412 CVE-2017-12413 CVE-2017-12414 CVE-2017-12415 CVE-2017-12416 CVE-2017-12417 CVE-2017-12418 CVE-2017-12419 CVE-2017-12420 CVE-2017-12421 CVE-2017-12422 CVE-2017-12423 CVE-2017-12424 CVE-2017-12425 CVE-2017-12426 CVE-2017-12427 CVE-2017-12428 CVE-2017-12429 CVE-2017-12430 CVE-2017-12431 CVE-2017-12432 CVE-2017-12433 CVE-2017-12434 CVE-2017-12435 CVE-2017-12436 CVE-2017-12437 CVE-2017-12438 CVE-2017-12439 CVE-2017-12440 CVE-2017-12441 CVE-2017-12442 CVE-2017-12443 CVE-2017-12444 CVE-2017-12445 CVE-2017-12446 CVE-2017-12447 CVE-2017-12448 CVE-2017-12449 CVE-2017-12450 CVE-2017-12451 CVE-2017-12452 CVE-2017-12453 CVE-2017-12454 CVE-2017-12455 CVE-2017-12456 CVE-2017-12457 CVE-2017-12458 CVE-2017-12459 CVE-2017-12460 CVE-2017-12461 CVE-2017-12462 CVE-
```

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
- Provenance (metadata) and validation (testing NFR) of builds is vital
 - Not all developers are operationally aware

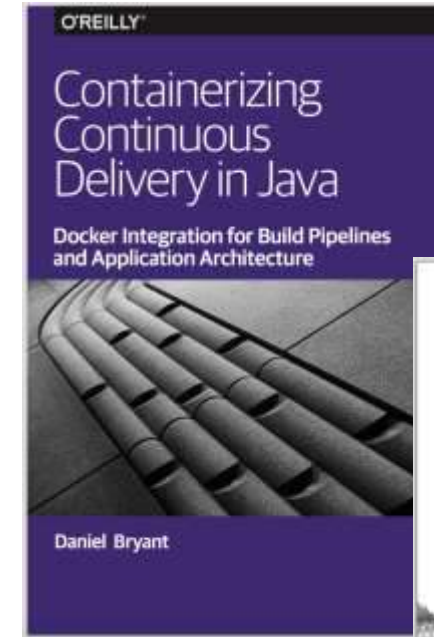
Thanks for listening...

Twitter: [@danielbryantuk](https://twitter.com/danielbryantuk)

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

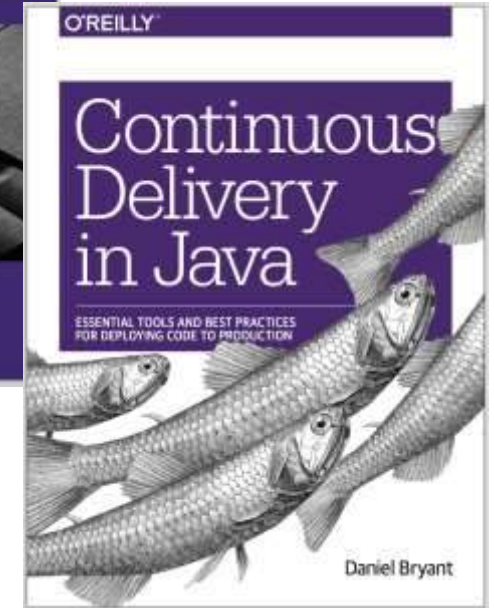
Writing: <https://www.infoq.com/profile/Daniel-Bryant>

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



oreil.ly/2RgU3Pe

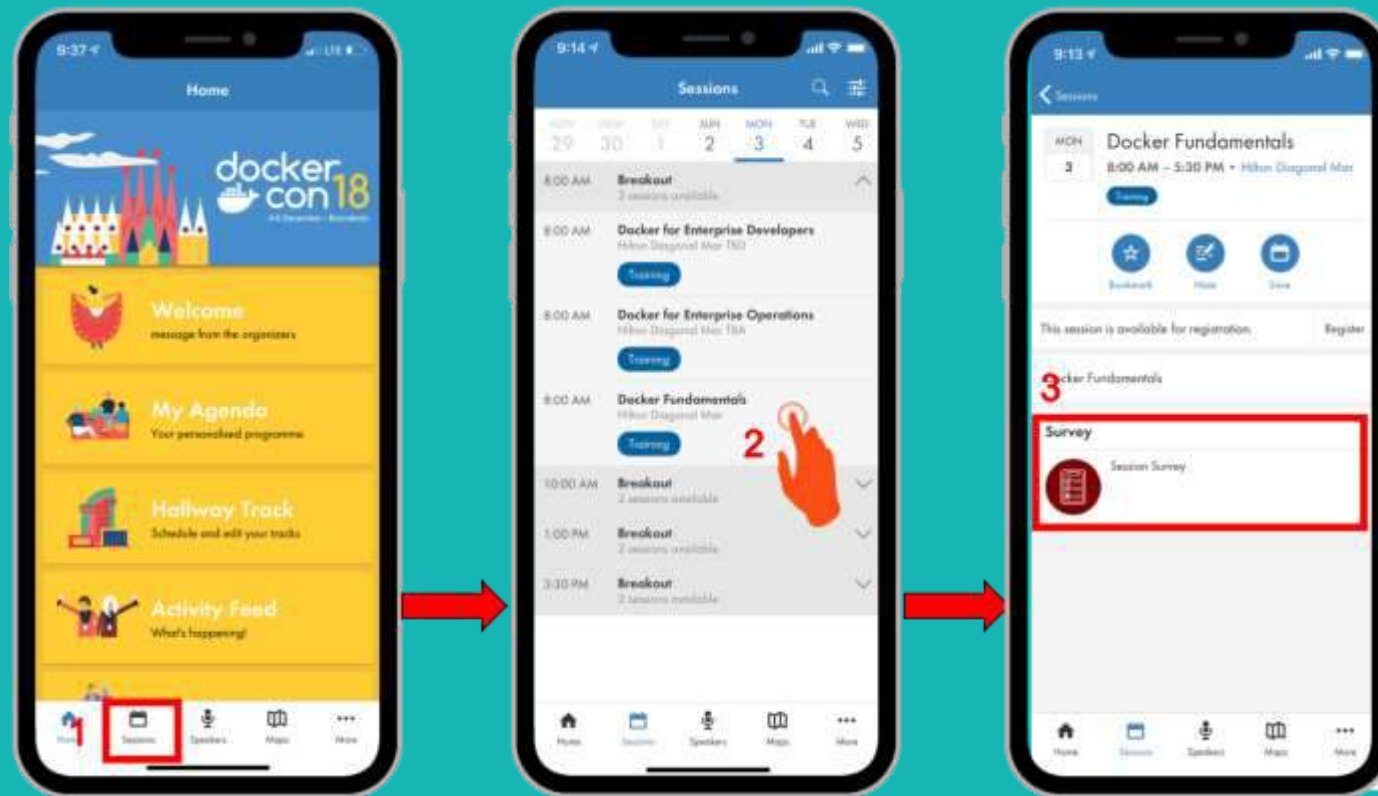
bit.ly/2jWDSF7



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.



Bedtime reading



05/12/2018

@danielbryantuk

BigPictureTech