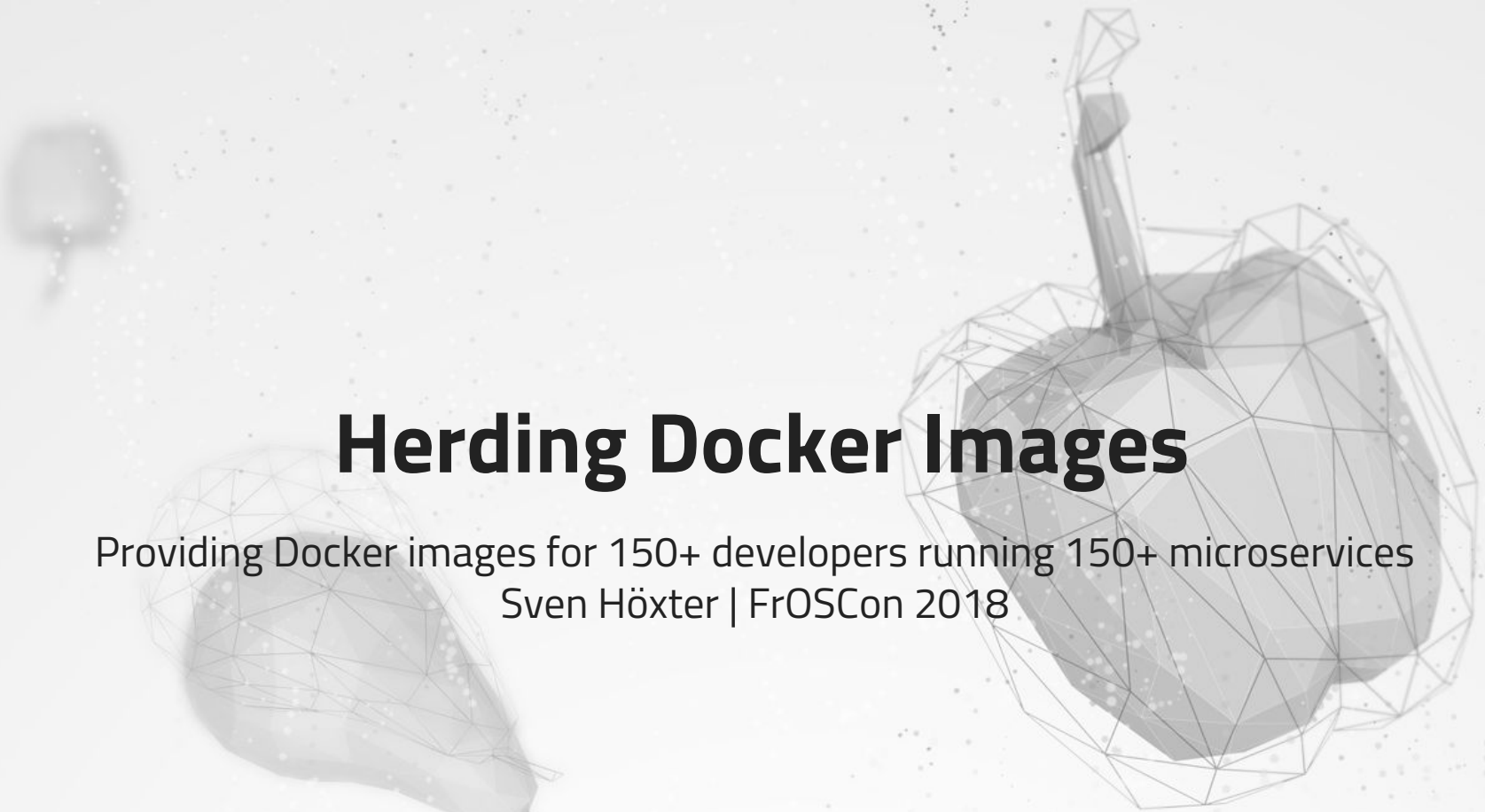
Herding Docker Images

Providing Docker images for 150+ developers running 150+ microservices

Sven Höxter | FrOSCon 2018



REWE digital

Warum stehe ich eigentlich hier?

Warum will ich einheitliche base images?

Updates, primär die sicherheitsrelevanten

Einheitlicher Build-Prozess, vereinfacht Implementierung von Änderungswünschen.

Probleme müssen nur einmal debugged werden.

Compliance

- Niemand möchte mutwillig gegen Lizenzen verstoßen.
- Niemand möchte die Flexibilität in agilen Unternehmen einschränken.
- Der Inhalt von Docker Hub Containern ist häufig unklar, siehe z.B. auch Dirk Hohndel, Legal and Licensing Workshop 2018, https://lwn.net/Articles/752982/
- Versuch eine hilfreiche Basis für alle Nutzer im Unternehmen zu schaffen.

Vertrauenswürdige Inhalte

Vertrauenswürdige Inhalte

Geschichte aus dem Leben eines Admins
Ein Unternehmen machte erste Schritte mit Puppet. In allen Installationen fand sich aber ein cfengine Paket. Die Frage warum das so ist führte zu "**Hm, vielleicht ist das in dem Baseimage drinne das ich für unser FAI Setup von einem FTP Server geladen habe.**"

Situation bei der REWE Digital

~150 Entwickler

~150 microservices

~150 microservices

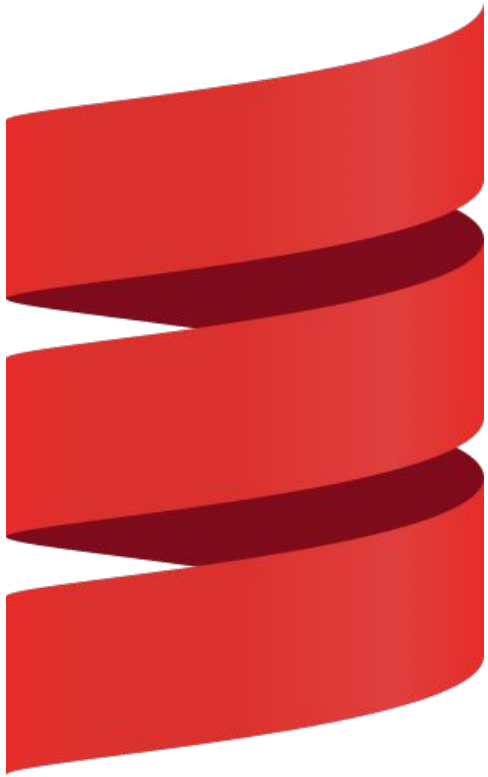
From Monolith to Microservices
Paul Puschmann | OSDC 2018

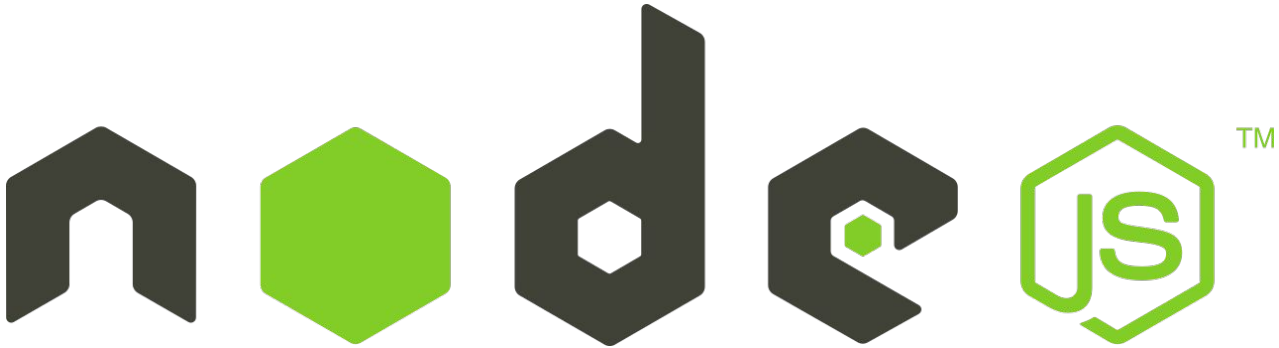
https://www.youtube.com/watch?v=DmMvthLegJo

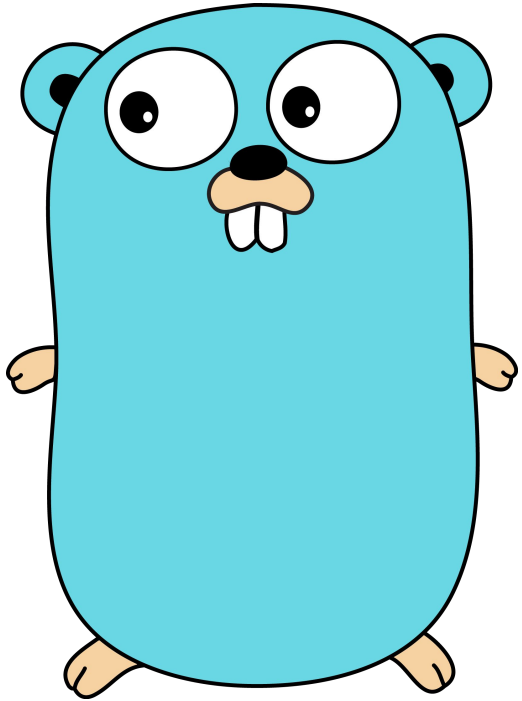
Deployment Format: Container

Unabhängige Teams, viele Götter


Java

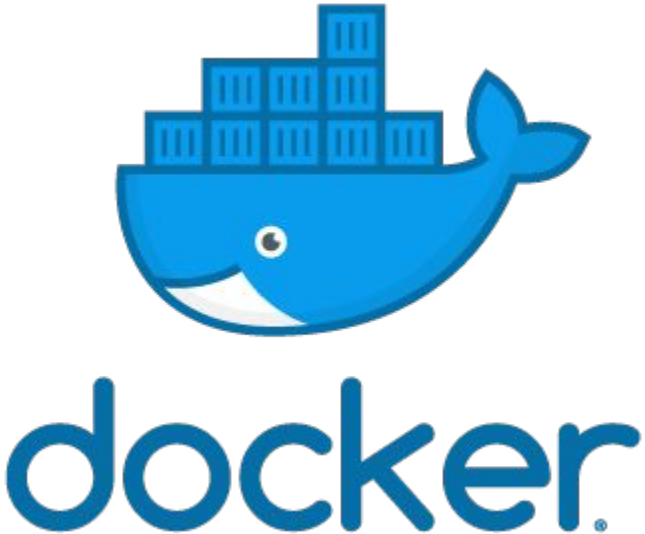



node
JS
TM



Hatten wir auch mal ...

Verschiedene Betriebsplattformen


docker

kubernetes

Das Resultat nach dem los laufen

rewe-base

rewe-java-generic

rewe-java

rewe-java-ubuntu

rewe-java-root

rewe-go

debian-java

usw.

Warum jetzt anders?
Warum jetzt Debian?

Drei Distributionen sind zwei zu viel

(Oracle) Java und die glibc

```
FROM registry.rewe-digital.com/rewe-base
[...]
# Install cURL, certificates, tar and glibc-2
RUN apk --update add curl tar ca-certificates libgcc
RUN curl -Ls https://github.com/sgerrand/alpine-pkg-glibc/releases/download/2.23-r4/glibc-2.23-r4.apk >
/tmp/glibc.apk
RUN curl -Ls https://github.com/sgerrand/alpine-pkg-glibc/releases/download/2.23-r4/glibc-bin-2.23-r4.apk
> /tmp/glibc-bin.apk
RUN apk add --allow-untrusted /tmp/glibc*.apk
[...]
```

Und auch andere - hier rewe-go

```
FROM registry.rewe-digital.com/rewe-base
[...]
RUN apk --update add tzdata curl ca-certificates tar && \
        curl -Ls https://github.com/andyshinn/alpine-pkg-glibc/releases/download/2.23-r1/glibc-2.23-r1.apk > /tmp/glibc.apk && \
        apk add --allow-untrusted /tmp/glibc.apk && \
        rm -rf /var/cache/apk/*
[...]
```

Kein Alpine Linux blaming, aber für uns zu viele Probleme.

Selbst mit glibc Probleme mit Java 9 + 10.

Niemand im Unternehmen verwendet es außerhalb von Containern, schwierige QA, busybox update filterte ENV variablen mit “.” im Namen.

Folge: Viel Zeit mit debugging von Alpine Spezifika verbracht.

Kein Fehler von Alpine, es passt nur nicht für uns.

Tradeoffs Debian vs Alpine

Pros

1. **Glibc als Standard**
2. **Einheitliche Plattform**
   Sonstige VMs verwenden auch Debian.
   The Universal Operating System. ;)
3. **Java 9+**

Contras

1. **5.5MB vs 55MB (Debian slim)**
   Nicht so wichtig, liegt nur einmal auf jedem Docker Hosts pro release. Relativiert sich sobald ein JRE an Bord ist.
2. **Memory overhead**
   Schwer zu quantifizieren, aber wir laden die glibc auch in Alpine Systemen.

Verworfene Alternative: Google Jib

- Glibc + OpenJDK
  https://cloudplatform.googleblog.com/2018/07/introducing-jib-build-java-docker-images-better.html
- Leider nicht nur mit Java unterwegs. :(
- Derzeit noch Oracle JRE in Verwendung, möglicherweise ab Java 11 wieder eine Option.
- Verfügbarkeit weiterer Tools im Image teilweise gewünscht.

Unsere Lösung

Jenkins

+ Debian-slim docker image
+ Docker build
+ Google ContainerDiff

Komponenten

1. Jenkinsfile
2. Dockerfile
3. Deckschrubber policy
4. Interne CAs
5. (Jenkins seedjob)

Jenkinsfile 1/4

```
node {
        def releaseVersion = '9'
        def buildDate = sh(script: 'date -u +"%Y%m%d%H%M"', returnStdout: true).trim()
        def registry = 'registry.rewe-digital.com'
        def imageName = 'rewe-debian'
        // filter list must contain files in double quotes separated by spaces
        // See also https://golang.org/pkg/text/template/ multi-way equality tests
        def modifiedFilter = '""'

[...]
```

Jenkinsfile 2/4

```
[...]
        stage("build image") {
        checkout scm
        sh "docker build \
        --force-rm=true --pull=true --no-cache=true \
        -t ${registry}/${imageName}:${releaseVersion}.${buildDate} \
        -t ${registry}/${imageName}:${releaseVersion}.latest \
        -t ${registry}/${imageName}:latest \
        -t ${registry}/${imageName}:stable ."
        }
[...]
```

Jenkinsfile 3/4

```
[...]
        stage("push image") {
          def diffResult = sh(script: "container-diff diff -q --type=file --format ' \
            {{if .Diff.Adds}}{{range .Diff.Adds}}{{.Name}}{{\" a\\n\"}}{{end}}{{end}} \
            {{if .Diff.Dels}}{{range .Diff.Dels}}{{.Name}}{{\" d\\n\"}}{{end}}{{end}} \
            {{if .Diff.Mods}}{{range .Diff.Mods}}{{if eq .Name ${modifiedFilter}}}{{else}}{{.Name}}{{\"
m\\n\"}}{{end}}{{end}}{{end}} \
        ' daemon://${registry}/${imageName}:${releaseVersion}.latest
remote://${registry}/${imageName}:${releaseVersion}.latest 2>&1 | \
        grep -E '^/.+[adm]\$' || echo 'unchanged'", returnStdout: true).trim()

        echo "diffResult: $diffResult"
        if (diffResult != 'unchanged') {
          sh "docker push ${registry}/${imageName}:${releaseVersion}.${buildDate}"
          sh "docker push ${registry}/${imageName}:${releaseVersion}.latest"
          sh "docker push ${registry}/${imageName}:latest"
          sh "docker push ${registry}/${imageName}:stable"
        }
        }
[...]
```

Jenkinsfile 4/4

```
[...]
        stage("push deckschrubber policy") {
        sh "curl -s -H 'Content-Type: application/json' -X PUT -d @dspolicy.json
https://${registry}/config/${imageName}/prod"
        }

        stage("cleanup local build") {
        sh "docker rmi -f \$(docker images -q '*/${imageName}' | uniq) || exit 0"
        }
}
```

Dockerfile rewe-debian 1/4

```
# stretch-slim ist für uns gut genug
FROM debian:stretch-slim

MAINTAINER Platform Squad <some@email-address>

# generische Umgebung für alle Applikationen
RUN useradd -d /opt/rewe -U service -e 2038-01-18 && \
 mkdir -m 0755 /opt/rewe /var/log/rewe && \
 chown service:service /opt/rewe /var/log/rewe

VOLUME /var/log/rewe

[...]
```

Dockerfile rewe-debian 2/4

```
[...]

# security updates, zusätzliche Tools und cleanup um den diff sauber zu halten
RUN apt-get update; \
 apt-get install -y netcat wget ca-certificates; \
 apt-get -y upgrade && apt-get clean; \
 find /var/lib/apt/lists -type f -delete; \
 rm -f /var/log/apt/* \
  /var/log/*.log \
  /etc/machine-id \
  /var/cache/ldconfig/aux-cache

[...]
```

Dockerfile rewe-debian 3/4

```
[...]

# import interne CA
ADD cert /usr/share/ca-certificates/rewe
RUN echo "rewe/RootCA.pem" >> /etc/ca-certificates.conf && \
 echo "rewe/RSDeviceCA02.pem" >> /etc/ca-certificates.conf && \
 update-ca-certificates

[...]
```

Dockerfile rewe-debian 4/4

```
[...]

# “cache breaker” für container-diff
COPY Dockerfile /root/Dockerfile.rewe-debian
```

Dockerfile rewe-java 1/3

```
#multi-stage build Schritt I
FROM registry.rewe-digital.com/rewe-debian:stable as builder
MAINTAINER Platform Squad <some@email-address>

ENV JAVA_HOME="/opt/jre"
ENV PATH="${PATH}:${JAVA_HOME}/bin"
ENV JCE_POLICY="${JAVA_HOME}/conf/security/java.security"

RUN wget -O jre.tar.gz --no-verbose https://storage.rewe-digital.com/java/serverjre-10-latest.tar.gz
RUN tar xzf jre.tar.gz
RUN mv /jdk-10* ${JAVA_HOME}

[...]
```

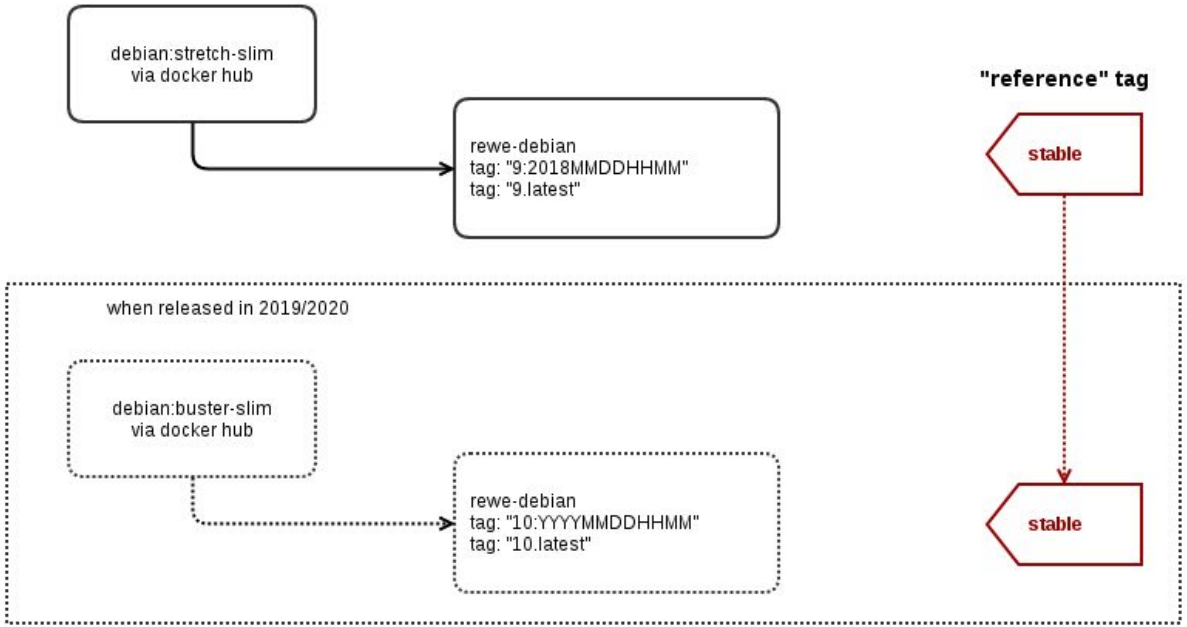
Dockerfile rewe-java 2/3

```
[...]

# JVM DNS cache settings
RUN sed -i -e 's/networkaddress\.cache\.negative\.ttl=10/networkaddress.cache.negative.ttl=5/' ${JCE_POLICY}
RUN echo 'networkaddress.cache.ttl=5' >> ${JCE_POLICY}

# import interne CA
ADD cert/RootCA.cer /tmp
ADD cert/RSDeviceCA02.cer /tmp
RUN echo yes | keytool -cacerts -storepass changeit -trustcacerts -importcert -alias rootca -file /tmp/RootCA.cer
RUN echo yes | keytool -cacerts -storepass changeit -trustcacerts -importcert -alias rsdeviceca02 -file
/tmp/RSDeviceCA02.cer

[...]
```

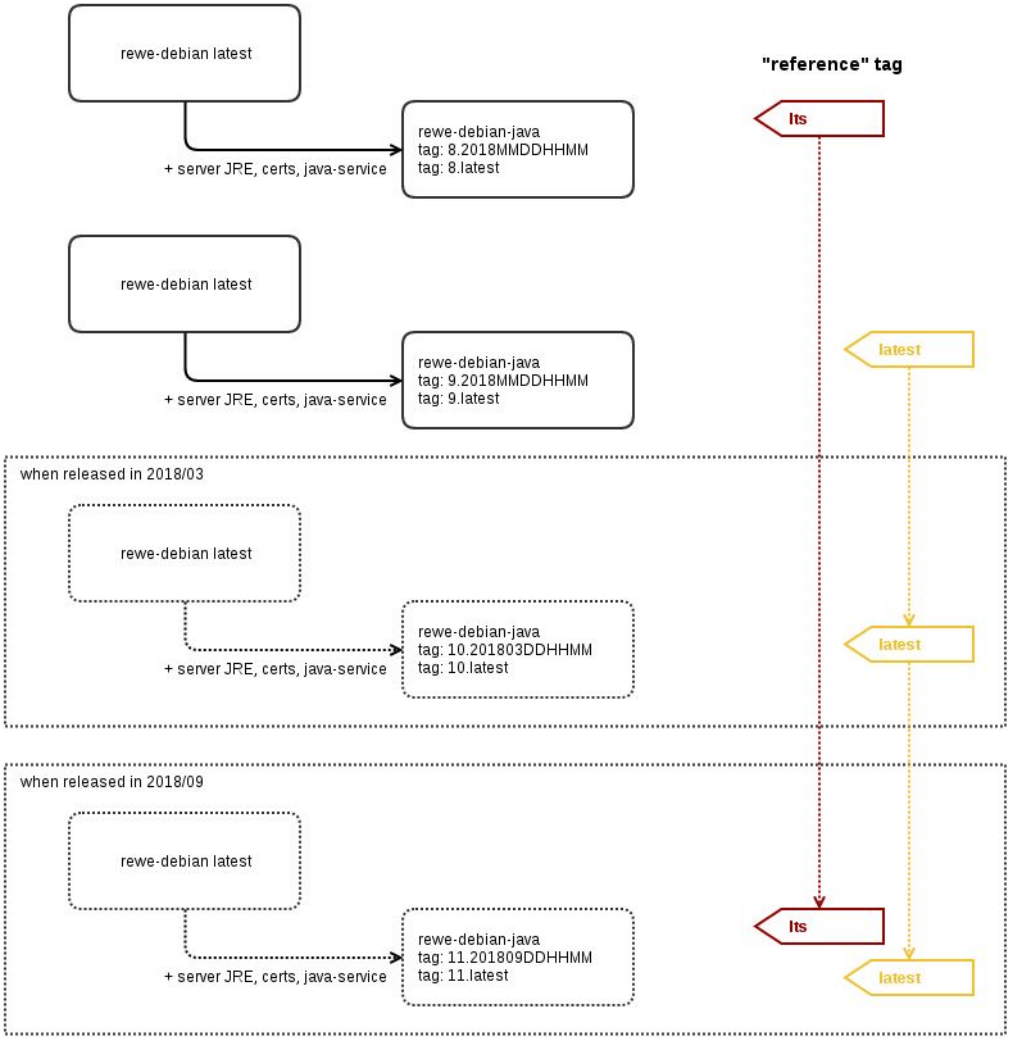
Dockerfile rewe-java 3/3

```
[...]
#multi-stage build Schritt II - finaler Container
FROM registry.rewe-digital.com/rewe-debian:stable

COPY Dockerfile /root/Dockerfile.rewe-debian-java

# expose ports
ENV SERVICE_PORT 8080
EXPOSE ${SERVICE_PORT}

#Umgebungsvariablen und Import des JRE
ENV JAVA_HOME="/opt/jre"
ENV PATH="${PATH}:${JAVA_HOME}/bin"
COPY --from=builder ${JAVA_HOME} ${JAVA_HOME}

#Startscript und Userwechsel
COPY java-service /usr/sbin/java-service
RUN chmod a+x /usr/sbin/java-service
USER service
ENTRYPOINT ["/usr/sbin/java-service", "start"]
```

Lifecycle rewe-debian



Lifecycle rewe-debian-java



REWE digital

Seedjob 1/2

```
 pipelineJobs: [
   [name: 'docker_image_rewe_debian', repo: 'PI/rewe-debian', branch: '*/stretch', jenkinsFilename: 'Jenkinsfile',
cron: '@daily'],
   [name: 'docker_image_rewe_debian_java8', repo: 'PI/rewe-debian-java', branch: '*/java8', jenkinsFilename:
'Jenkinsfile', cron: '@daily'],
   [name: 'docker_image_rewe_debian_java10', repo: 'PI/rewe-debian-java', branch: '*/java10', jenkinsFilename:
'Jenkinsfile', cron: '@daily'],
          ]
```

Seedjob 2/2

```
params.pipelineJobs.each { job ->
pipelineJob(job.name) {
        definition {
          cpsScm {
            scm {
              git("ssh://git@git.rewe-digital.com/${job.repo}.git", job.branch)
            }
            scriptPath(job.jenkinsFilename)
            }
          }
        logRotator() {
          artifactNumToKeep(300)
          numToKeep(10)
        }
        triggers {
          if(job.triggers) scm("${job.triggers}")
          if(job.cron) cron("${job.cron}")
        }
    }
}
```

Verbesserungspotenzial

- rewe-debian-dev bereitstellen als Basis für multi-stage builds.
- Build-Prozess für debian-slim komplett selber ausführen.

  Verlassen uns derzeit auf die Bereitstellung durch Mitentwickler im Debian Projekt.

  Verlassen uns auf die garantierten Freiheiten durch die DFSG.
- Kein nc und wget im (Produktions-) Container.
- container-diff für Debian/main paketieren.

Fails

- Mit java-package erstellte .deb Pakete des JREs lassen sich nicht installieren weil manpages aus debian-slim entfernt wurden.
- Der Versuch unseren Build-Prozess auf einem Jenkins in einem GKE Container auszuführen scheitert weil die Docker Version 17.03 von Google kein multi-stage build kann.

Fragen?

Sven Hoexter <sven@timegate.de>
GPG 7C0717F9FA2B2B9D788B141BA6DC24D9DA2493D1

Thank You!

Would like to join us?

www.rewe-digital.com/jobs.html

REWE digital