GIS
Docker und Azure-Cloud

Neues aus der Entwicklung

Agenda

- Ausgangssituation / Rahmenbedingungen / Entscheidung
- Docker
- Azure
  - Beispiel VM anlegen via Web Interface
- Putting all together
  - Beispiel:
    Azure ← → Docker (Deploy Rollback Tomcat → Gis Service → Tomcat)
- Lessons learned „so far“
- Nächste Schritte

Ausgangssituation

- Gis Services sind kompliziert zu deployen.
- Die zugrundeliegenden Datenmengen sind potentiell groß. Vorausgekachelte Kartenservices können mehrere GB groß sein und aus Millionen von einzelnen Files bestehen.
- Zuerst müssen die Daten auf den Server kopiert werden, dann die Services im Gis Server konfiguriert werden. Man muss das Service starten. Dann muss getestet werden.
- Falls dies Fehlschlägt wieder muss für eine Rollback alles wiederholt werden.

→ Gesucht ist ein Mechanismus, der diesen Vorgang vereinfacht und es auch für nicht GIS Experten es ermöglicht GIS Services zu deployen und zu betreiben.

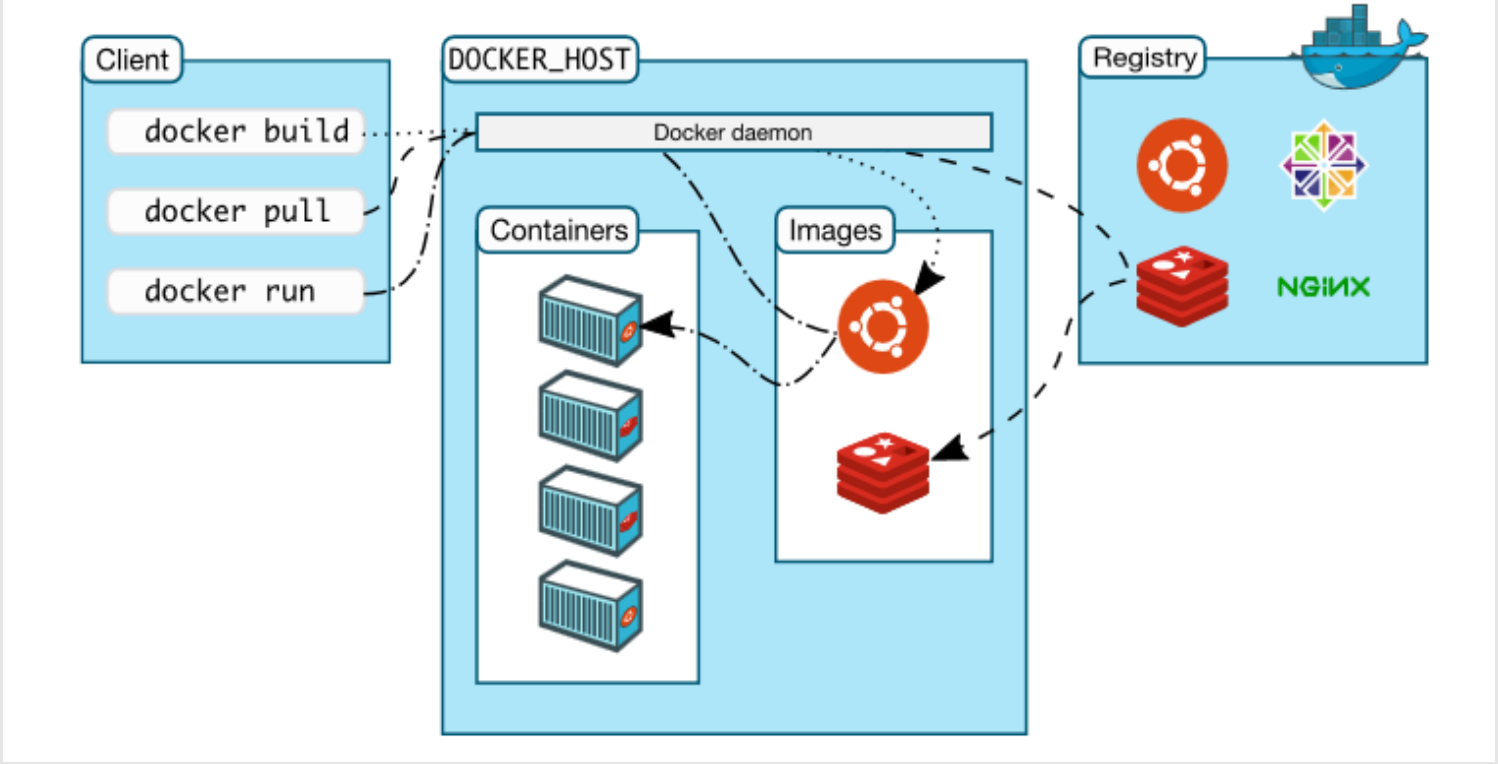
Rahmenbedingungen

- Der Entwicklungs- und vor allem der Deploy-Vorgang muss einfacher werden.
- Rollback muss möglich sein.
- Technologie muss skalierbar sein
- Technologie muss intern (in House) und Extern (Rechenzentrum) laufen können.
- Ein Vendor lock in soll soweit wie möglich vermieden werden.
- Es soll mittelfristig Kosteneinsparungen (Lizenz-, Wartungs-, Betriebs- und Personalkosten) bringen.

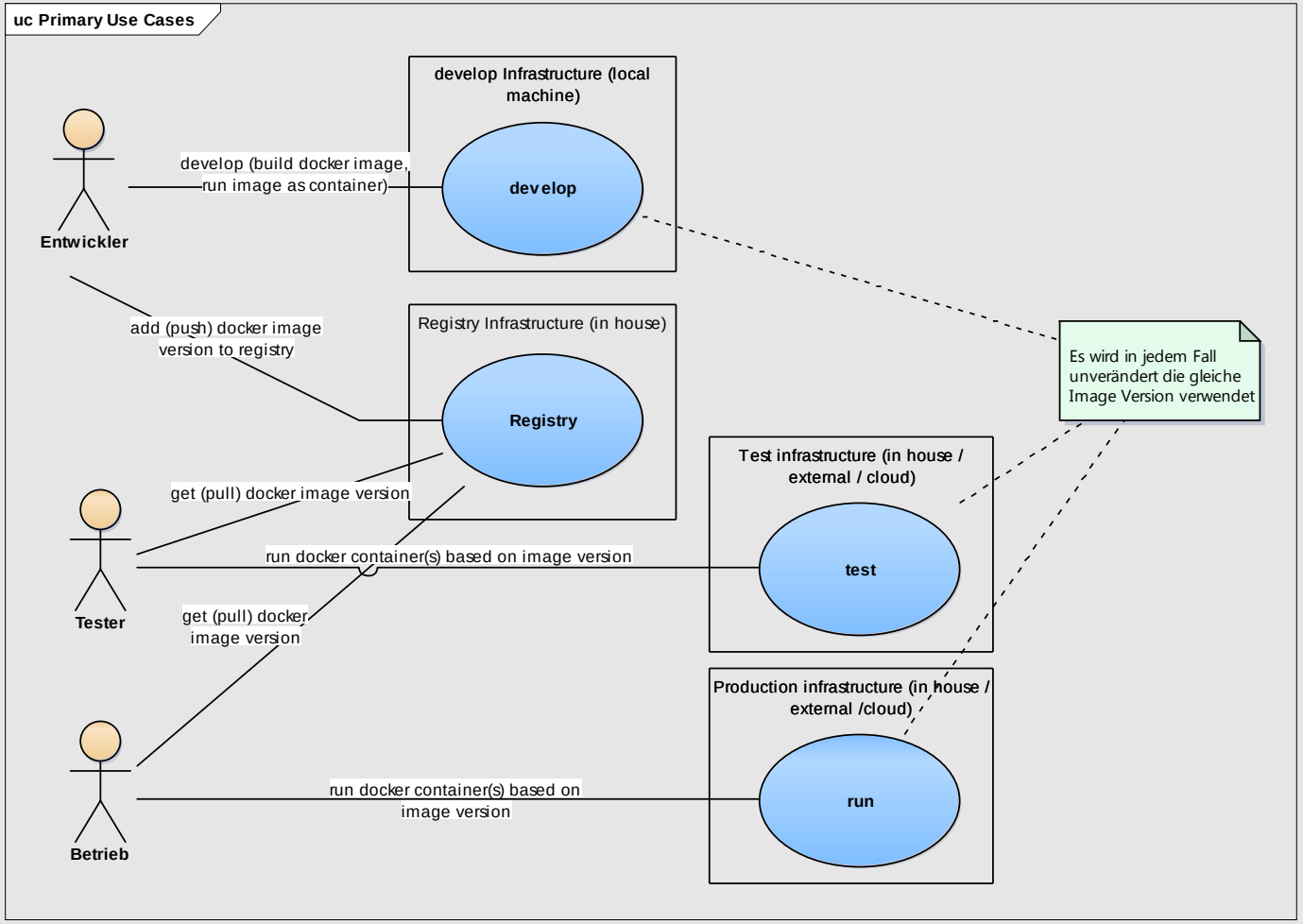
Testszenario

- Der GIS Server der Wahl ist GeoServer (war File, dass innerhalb eines Servlet Containers läuft)
- Als Testdaten sollen drei Szenarien verwendet werden:
  - Die in GeoServer mitgepackten Testdaten (ist für erste testzwecke einfach, da man alle möglichen Services vorkonfiguriert hat und die Größe der Datenmenge überschaubar ist).
  - Corine Landcover (mittelgroßer Datensatz)
  - Gewässernetz (große Datenmenge komplexere Signatur)
- Als Cloud soll Microsoft Azure verwendet werden
- Um realistische Kosten ermitteln zu können sollen die Szenarien 2 und 3 mit einem reproduzierbaren Traffic für eine bestimmte Zeit belegt werden (mittels Apache JMeter Test) um daraus Kosten ableiten zu können.

Docker

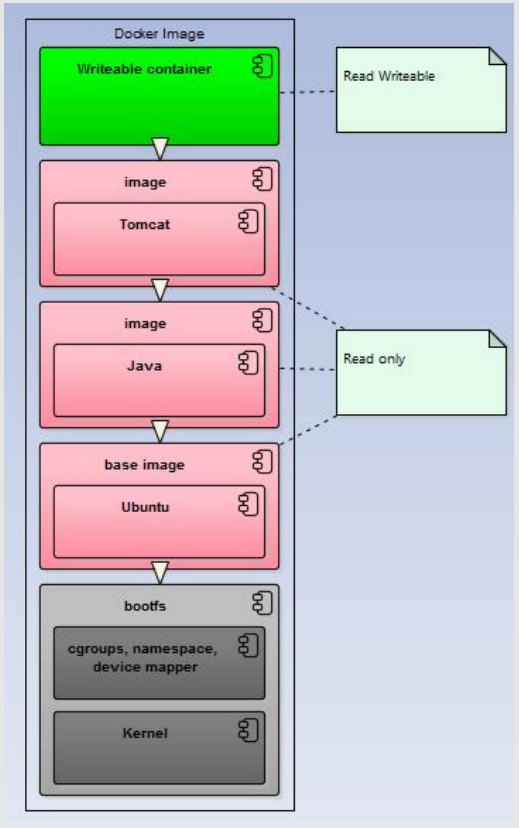

Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

Docker Workflow



Docker File → Docker Image → Docker Container

```
1 FROM java:8-jre
2
3 MAINTAINER hadrbolec <michael.hadrbolec@umweltbundesamt.at>
4
5 ENV TOMCAT_MAJOR 8
6 ENV TOMCAT_VERSION 8.0.23
7 ENV CATALINA_HOME /usr/local/apache-tomcat-${TOMCAT_VERSION}
8 RUN mkdir -p ${CATALINA_HOME}
9 ENV PATH ${CATALINA_HOME}/bin:$PATH
10 # set defaults that make sense (can be overriden on demand)
11 # -XX:MaxPermSize=512 -XX:PermSize=256m Settings make only fpr java 7 sense
12 ENV TOMCAT4U_JAVA_OPTS "-server -Xms1024M -Xmx2048M -XX:SoftRefLRUPolicyMSPerMB=36000 -
13 # The default encoding should be allways UTF-8
14 ENV CATALINA_OPTS ${TOMCAT4U_JAVA_OPTS}
15 WORKDIR ${CATALINA_HOME}
16 # copy the tar file temporary into the conatiner
17 COPY apache-tomcat-8.0.23.tar.gz /usr/local/
18 RUN tar -xf /usr/local/apache-tomcat-8.0.23.tar.gz --strip-components=1
19 # override the tomcat default server.xml file
20 ADD server.xml ${CATALINA_HOME}/conf/
21 # for production switch log level to SEVERE
22 #ADD logging.properties ${CATALINA_HOME}/conf/
23 # cleanup (delete all not necesary files)
24 RUN rm /usr/local/apache-tomcat-8.0.23.tar.gz
25 RUN rm /usr/local/apache-tomcat-8.0.23/bin/*.bat
26 RUN rm -r ./webapps/*
27 # expose the webapps directory as volume (allows to add war files)
28 # expose the logs directory as volume.
29 #VOLUME [${CATALINA_HOME}/webapps, ${CATALINA_HOME}/logs ]
30
31 EXPOSE 8080
32
33 CMD ["/usr/local/apache-tomcat-8.0.23/bin/catalina.sh", "run"]
```



Putting all together

- Info über der remote Docker Container
  - docker info

Welche Images liegen dort?

```
docker  images
```

Welche Services laufen dort?

```
docker  ps
```

Gis Service Deploy / Rollback

- Mit docker pull die Images aus der Registry holen

  (derzeit nicht möglich haben keine von außen zugängliche Firmenregistry)

  docker start tomcat80

  http://192.168.99.100/

- Altes Service stoppen

  docker stop tomcat80

- Neues services deployen

  docker run -p 80:8080 --volumes-from dv4u_clcdata -d --name corine4u geoserver4u

  http://192.168.99.100/geoserver

- Rollback (neues Service stoppen altes starten aufräumen)

  docker stop corine4u

  docker start tomcat80

  http://192.168.99.100/

  docker rm corine4u

  …

  docker stop tomcat80

  docker rm tomcat80

Docker Produkte

https://www.docker.com/products/overview

Docker Machine

Automated Docker provisioning

Docker Swarm

Host clustering and container scheduling

Docker Compose

Define multi-container applications

Docker Registry

Open source Docker image distribution

* Not included in toolbox

Docker Engine

Creates and runs Docker containers

Kitematic

Desktop GUI for Docker

Lessons learned „so far“

- Container Technologie hat großes Potential
  - Einfacher und schneller Deploy & Rollback
    - Ansatz wiederverwenden Standardisiertes Geoserver Image mit unterschiedlichen Konfigurationen & Daten funktioniert.
    - Großes Potential für Automatisierung und automatische Skalierung
  - Hosting auf Azure funktioniert grundsätzlich
    - Lokales Entwickeln des Services und unverändertes verschieben des Images auf Produktion funktioniert.
  - Security: Docker läuft unter Root und daher muss Docker Port und Firmenregistry entsprechend gesichert sein.
- Azure VM einfach über Portal anzulegen
  - Unterstützt Docker

Kontakt & Information

Michael Hadrbolec

michael.hadrbolec@umweltbundesamt.at